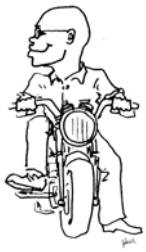
เริ่มต้นกับ Java

ฉบับเริ่มต้น (จริง ๆ)

ลั่นฟ้า ชุมสาย

ภาควิชาคอมพิวเตอร์ธุรกิจ

วิทยาลัยฟาร์อีสเทอร์น



คำนำ

หนังสือเล่มนี้เขียนขึ้นมาเพื่อเป็นแนวทางให้นักศึกษา วิทยาลัยฟาร์อีสเทอร์น ใช้เป็นคู่มือในการเรียนวิชาหลักการเขียนโปรแกรมคอมพิวเตอร์ (เบื้องต้น) เนื่องจากว่าในระยะเวลา สองถึงสามปีที่ผ่านมา หนังสือที่ทางภาควิชาคอมพิวเตอร์ธุรกิจใช้เป็นหนังสืออ้างอิงในการเรียนการสอนนั้น โดยส่วนใหญ่แล้วจะเป็นหนังสือที่แต่งโดยชาวต่างชาติ ทำให้การใช้หนังสือเหล่านั้นในการประกอบการเรียนของนักศึกษาเป็นไปด้วยความลำบาก เพราะต้องใช้เวลานานในการแปล ถึงแม้ว่าได้รับความช่วยเหลือจาก อาจารย์หลาย ๆ ท่านในภาควิชา ในการช่วยชี้แนะ ช่วยแปล แต่ก็เป็นไปได้เพียงน้อยนิดเท่านั้น ผู้เขียนได้เล็งเห็นถึงความสำคัญของการมีหนังสือไว้อ่านเอง ในเวลาที่ไม่สามารถเข้าหาอาจารย์ได้ จึงได้พยายามเขียนหนังสือเล่มนี้ขึ้น เพื่อให้เป็นคู่มือที่นักศึกษาสามารถนำไปใช้ในการค้นคว้าหาความรู้ในเรื่องของการเขียนโปรแกรมอีกทางหนึ่ง

หนังสือเล่มนี้ เป็นได้เพียงคู่มืออ้างอิงเบื้องต้น ในการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษา Java เท่านั้น เนื่องจากว่า Java เป็นภาษาที่ ถ้าผู้อ่านต้องการที่จะรู้ถึงการใช้อย่างละเอียด จะต้องใช้เวลาในการศึกษาค้นคว้ามากพอสมควร ยังมีหนังสืออีกหลายเล่มที่สามารถนำมาใช้เป็นแบบอย่างในการเขียนโปรแกรมได้ ผู้เขียนหลายท่านได้นำเอาบทความ ตำรา และข้อชี้แนะในการใช้ภาษา Java ขึ้นไปเก็บไว้ใน web site หลายแห่งผู้อ่านควรใช้ web site เหล่านี้ช่วยในการศึกษา ค้นคว้าเพื่อให้การเขียนโปรแกรมด้วยภาษา Java เป็นไปได้อย่างรวดเร็ว และแม่นยำมากยิ่งขึ้น

ถ้อยคำและสำนวนในหนังสือเล่มนี้อาจดูแปลก ๆ สำหรับหลาย ๆ ท่าน สาเหตุก็เนื่องจากว่าผู้เขียนใช้สำนวนที่ตัวเองถนัด และเป็นสำนวนที่ผู้เขียนเองได้สัมผัสมาจากต่างประเทศ ซึ่งอาจไม่เป็นที่ถูกใจของผู้อ่านหลาย ๆ ท่าน แต่ผู้เขียนขอสัญญาว่าจะพยายามทำให้สำนวนเหล่านี้เป็นสำนวนไทย ๆ ให้ได้ในอนาคต อีกสิ่งหนึ่งที่ผู้เขียนไม่สามารถหลีกเลี่ยงได้ในการเขียนหนังสือเล่มนี้ ก็คือ การใช้ภาษาอังกฤษควบคู่ไปกับภาษาไทยทั้งนี้ก็เพราะว่า ศัพท์ทั้งหลายที่เกี่ยวข้องกับคอมพิวเตอร์นั้น เราได้มาจากภาษาอังกฤษ และผู้เขียนเองเห็นว่าควรที่จะใช้ศัพท์เหล่านี้โดยตรง โดยไม่ต้องแปล หรือใช้ศัพท์ที่นักภาษาศาสตร์ชาวไทยได้แปลออกมาใช้ เพราะจะทำให้เสียเวลาเพราะอย่างไรเราก็ทับศัพท์เหล่านี้อยู่แล้ว

ผู้เขียนหวังว่าหนังสือเล่มนี้ จะช่วยให้การเรียนเขียนโปรแกรมภาษา Java เป็นไปอย่างราบรื่นและประสพผลสำเร็จสำหรับทุก ๆ ท่าน

ผู้เขียนขอขอบคุณอาจารย์ในภาควิชาทุกท่าน ที่ได้ช่วยเป็นกำลังใจและช่วยชี้แนะในเรื่องของเนื้อหาในหนังสือเล่มนี้ ว่าควรจะครอบคลุมเรื่องต่าง ๆ มากน้อยแค่ไหน ขอบคุณอาจารย์ ชัยรัตน์ ขันแก้ว ที่ช่วยอ่านและแก้ไขคำผิด และขอขอบคุณอาจารย์อีกหลาย ๆ ท่านที่ได้ช่วยเหลือและสนับสนุนให้ผู้เขียน ได้มีโอกาสเขียนหนังสือเล่มนี้

ลั่นฟ้า ชุมสาย

เมษายน 2546

สารบัญ

บทที่ 1 ก่อนจะถึง Java 1

ทำไมถึงเลือก Java เป็นภาษาในการเขียน code 1
เตรียมความพร้อม 2
ติดตั้ง JDK
การกำหนด path
เขียน code แบบ application หรือ applet 4
การสร้างโปรแกรม Java ที่เป็น application 5
การ execute โปรแกรม HelloWorld 6
การสร้างโปรแกรม Java ที่เป็น applet 7
การเขียนโปรแกรม Java ด้วย EditPlus 8
ที่มาของ Java โดยสังเขป 12

บทที่ 2 ข้อมูล ตัวแปร และ การประมวลผล 13

ข้อมูล และ ตัวแปร 13
ชนิดของข้อมูล 14
Keyword
Primitive datatype
ตัวอย่างการประกาศตัวแปร 15
การกำหนดค่าให้กับตัวแปร
การกำหนดค่าให้กับตัวแปรภายในโปรแกรม 16
การใช้ final ในการกำหนดค่าให้กับตัวแปร 18
อายุ และ ขอบเขตการใช้งาน ของตัวแปร 19
การสร้างประโยค 23
การประมวลผล 26
การประมวลผลข้อมูลที่เป็น integer 26
การใช้ operator ต่าง ๆ 29
การประมวลผล short และ byte 32
การเปลี่ยนแปลงข้อมูลด้วยการ cast 33
การประมวลด้วยตัวแปรที่มีจุดทศนิยม 35
การประมวลผลข้อมูลต่างชนิดกัน 37
การใช้ Mathematical Functions ต่าง ๆ 39
การใช้ข้อมูลที่เป็น Character 43
การใช้ตัวแปรชนิด boolean 45
Logical operator
Relational operator
การประมวลระดับ bit (shift และ bitwise) 49
การกำหนดค่าให้ตัวแปรผ่านสื่อนำเข้ามาตรฐาน 51

บทที่ 3 การประมวลผลแบบวน (Repetition) 58

การตัดสินใจ และ การเปรียบเทียบ
ประโยคที่ใช้ if 58
การใช้ if ในประโยคที่มีมากกว่าหนึ่งประโยค 60
if – else
การใช้ Conditional operator ? : 68
การใช้ switch 69
การทำงานแบบวน 72
การใช้ for loop 73
การใช้ StreamTokenizer 77
การใช้ while loop 80
การใช้ do … while 83

การใช้ break และ continue 86
การใช้ Nested loop 88
การใช้ labeled continue และ labeled break 90

บทที่ 4 การใช้ Array และ String 96

การใช้ Array
การกำหนดค่าเบื้องต้นให้กับ Array 97
การอ้างถึง array ด้วยการ clone และการ copy array 101
การใช้ array แบบยืดหยุ่น 104
การค้นหาข้อมูลใน array 106
การค้นหาแบบตามลำดับ 106
การค้นหาแบบหารสอง 108
การเรียงลำดับข้อมูล 109
การสร้าง array ที่มีข้อมูลเป็น array 111
การใช้ array ที่มีข้อมูลในแต่ละแถวไม่เท่ากัน 113
การใช้ String 115
Array of characters
String ใน Java 116
การเปรียบเทียบ String 118
มากกว่า น้อยกว่า 120
การเข้าหาตัวอักษรที่อยู่ใน String 121
การใช้ StringBuffer 125
การกำหนดให้ข้อมูลของ array เป็น String 129

บทที่ 5 Objects และ Classes 133

Class และ การสร้าง class
ตัวแปรที่เป็นสมาชิกของ class 134
Method 136
การเข้าหาตัวแปร และ method ของ class 136
การสร้าง method 138
กระบวนการที่เกิดขึ้นเมื่อมีการใช้ parameter list 140
การส่งค่าแบบ pass by value 141
การส่งค่าแบบอ้างอิง 143
การกำหนดนโยบายการเข้าหาสมาชิกของ class 149
Constructor อีกครั้ง 150
การเรียก constructor จาก constructor 153
การ overload method 155
การสร้าง class ภายใน class 156
Package 162
การสร้าง package

บทที่ 6 การสร้าง class ใหม่จาก class เดิม และ การถ่ายทอดคุณสมบัติ 168

การใช้ this และ super() 168
การใช้ protected
การ override method 171
Polymorphism 175
การส่ง และ รับ object 179
การสร้าง interface 181

บทที่ 7 การตรวจสอบและดักจับ Error (Exceptions) 188

การ throw exception 188
การใช้ try และ catch 190
การใช้ try และ catch ใน loop 192
การใช้ printStacktrace() 192
การใช้ throw กับ try 195
การสร้าง exception ใช้เอง 198

บทที่ 8 Streams I/O 204

การใช้ File 205
การใช้ FileNameFilter 208
Input และ Output streams 210
การใช้ FileReader
การใช้ FileInputStream และ read() 211
การใช้ StreamTokenizer กับ FileReader 213
ข้อดีของการใช้ BufferedReader 217
การใช้ delimiter กับ text file 218
การใช้ FileWriter และ PrintWriter 218
Binary file 221
การใช้ DataInputStream และ FileInputStream
การใช้ DataOutputStream และ FileOutputStream
การใช้ writeUTF() และ readUTF()
การใช้ PrintWriter เพื่อผลลัพธ์ที่สวยงาม 224
การใช้ writeChars() 228
การสร้างและใช้ Random-access file 229
การใช้ seek() 233
การ update record 234

ตารางต่าง ๆ 240

ตาราง ASCII และ UNICODE (บางส่วน)
ตาราง Constant ที่ใช้บ่อย ๆ ในโปรแกรม
ตาราง ชนิดของข้อมูล
ตาราง Escape sequences
ตาราง Format ต่าง ๆ สำหรับข้อมูลที่เป็นตัวเลข

1
ก่อนจะถึง Java

ในบทที่หนึ่งนี้เราจะมาดูการติดตั้งเครื่องมือที่ใช้ ในการพัฒนาโปรแกรมด้วยภาษา Java ตัวอย่างการเขียนโปรแกรมในรูปแบบของ application และ การเขียนโปรแกรมในรูปแบบของ applet

หลังจากจบบทนี้แล้ว ผู้อ่านจะได้ทราบถึง

- การติดตั้งเครื่องมือที่ใช้ในการพัฒนาโปรแกรมในภาษา Java
- การเขียนโปรแกรมในรูปแบบของ application
- การเขียนโปรแกรมในรูปแบบของ applet
- การเขียน HTML ไฟล์ที่ใช้ในการ execute applet
- การเขียนโปรแกรม Java ด้วย EditPlus
- ประวัติคร่าว ๆ ของภาษา Java

ทำไมถึงเลือก Java เป็นภาษาในการเขียน code

Java เป็นภาษาที่นักพัฒนาโปรแกรมสามารถที่จะเขียน code เพียงครั้งเดียว แล้วนำไปใช้ได้กับเครื่อง computer ชนิด (platform) ต่าง ๆ ได้ โดยไม่ต้อง compile ใหม่ เช่น ถ้าเราเขียนโปรแกรมบนเครื่อง pc เราสามารถที่จะนำเอา code ที่ได้ compile เรียบร้อยแล้วไป execute ในเครื่องอื่น ๆ ที่ไม่ใช่ pc ได้ เช่น apple, linux, unix, หรือ อื่น ๆ ที่มี JVM หรือ Java Virtual Machine อยู่

Java เป็นภาษาที่ได้ถูกพัฒนาขึ้นเพื่อสนองต่อการพัฒนาโปรแกรมในรูปแบบของการเขียนโปรแกรมที่เรียกว่า OOP หรือ Object-Oriented Programming ซึ่งเป็นภาษาที่ช่วยให้การพัฒนาโปรแกรมมีความยืดหยุ่นสูง ผู้เขียน หรือ ผู้พัฒนาสามารถที่จะสนองตอบความต้องการของผู้ใช้ (user) ได้มากกว่าการพัฒนาในลักษณะของการเขียนโปรแกรมแบบเดิม (procedural language programming) และที่สำคัญอีกอย่างหนึ่งก็คือ Java เป็นภาษาที่ไม่ต้องเสียค่าใช้จ่ายใด ๆ เลย เราสามารถที่จะ download โปรแกรมสำหรับการพัฒนาได้จากบริษัท Sun Micro System หรือที่อื่น ๆ ที่ทาง Sun ได้อนุญาตให้มีการ download ได้

ก่อนที่เราจะเขียนโปรแกรมด้วย Java เราต้องมีอะไรบ้าง

1. เช่นเดียวกันกับการเขียนโปรแกรมในภาษาอื่น ๆ เราต้องมี Text Editor ที่ใช้สร้างและเก็บ source code

2. JDK (Java Development Kit) ซึ่งมีหลาย version ล่าสุดในขณะที่เขียนตำราเล่มนี้ คือ J2SE 1.4 ซึ่ง download ได้ที่ http://sun.java.com เราต้องใช้ JDK ในการ compile โปรแกรมที่เราได้เขียนขึ้น

3. Java VM หรือที่เรียกว่า Java Virtual Machine ซึ่งเป็นตัวกลางที่เปลี่ยน code ที่ได้จากการ compile เป็น code ที่สามารถ execute บนเครื่องต่าง ๆ (code ที่เครื่องนั้น ๆ รู้จัก – machine code) โดยปกติ VM จะถูกติดตั้งพร้อมกับ JDK

ถ้าเรามีอุปกรณ์ทั้งหมดที่ได้กล่าวมาแล้ว ขั้นตอนต่อไปที่เราต้องทำเพื่อให้การพัฒนาโปรแกรมของเราเป็นไปได้ด้วยความเรียบร้อย (ไม่มีอุปสรรคที่คาดไม่ถึง) เราต้องเตรียมความพร้อมของเครื่องก่อน

เตรียมความพร้อม (สำหรับ pc)

1. ติดตั้ง JDK

2. กำหนด path ภายในเครื่องเพื่อให้การ compile และ execute (run) โปรแกรมเป็นไปได้ – ถ้าเราไม่กำหนด path เครื่องของเราจะไม่รู้จักคำสั่งต่าง ๆ ที่ Java ใช้

ติดตั้ง JDK

เมื่อ download JDK มาแล้วการติดตั้งก็ทำได้ง่าย เนื่องจาก JDK เป็นไฟล์ที่เราเรียกว่า self-extracting file คือ ไฟล์ที่จะติดตั้งโดยอัตโนมัติเมื่อเรา execute (doubling click หรือ ด้วยวิธีการอื่น ๆ เช่น click บน icon ของ file แล้วกด enter) สำหรับไฟล์ j2se1.4.1 นั้นมีขนาดประมาณ 35.9 MB ซึ่งเป็นไฟล์ที่ใหญ่พอสมควร และใช้เวลานานในการ download (ถ้า speed ในการ download ไม่ดีพอ) ซึ่งเมื่อขยายออกจะมีขนาดประมาณ 64.6 MB เพราะฉะนั้นเราต้องเตรียมเนื้อที่ไว้ให้เหมาะสมในการ install JDK ถ้าหากว่าเนื้อที่ในการจัดเก็บมีน้อย เราก็ควรใช้ JDK version อื่น ๆ ที่มีขนาดเล็กกว่า

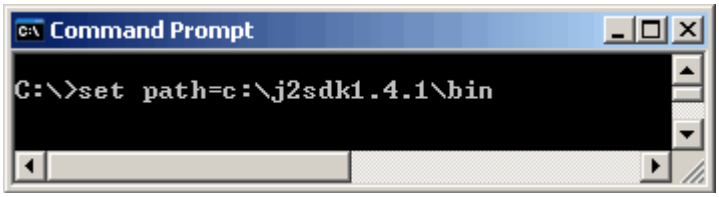
การกำหนด path

ถ้าการพัฒนาโปรแกรมเป็นการพัฒนาบนเครื่อง Windows 9x การกำหนด path ก็สามารถที่จะทำได้ด้วยวิธีการต่าง ๆ ดังนี้

1. เปิดไฟล์ autoexec.bat แล้วเพิ่ม path ที่ได้ติดตั้ง JDK ไว้ ซึ่งถ้าเราติดตั้ง J2SDK 1.4.1 ไว้ใน drive c การกำหนด path ในไฟล์ autoexec.bat ก็ทำได้ดังนี้

   ใส่คำสั่ง set path=c:\j2sdk1.4.1\bin ในบรรทัดใหม่ถ้ายังไม่มีมีคำสั่ง set path อยู่เลย แต่ถ้ามีการใช้ คำสั่ง set path อยู่แล้วก็ให้ใส่ ; (semicolon) หลังตัวสุดท้ายของ path แล้วจึงใส่ c:\j2sdk1.4.1\bin

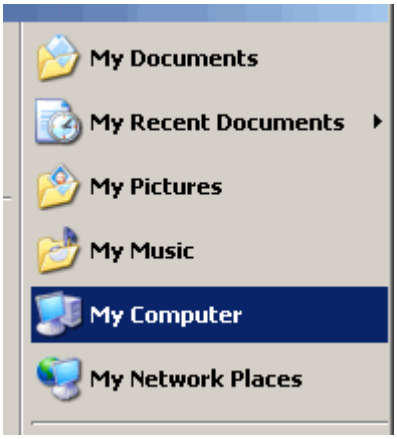
2. หรือเราอาจกำหนด path ทุกครั้งก่อนการเขียนโปรแกรม (ถ้าเราไม่ได้กำหนดในไฟล์ autoexec.bat) โดยไปที่ Command Prompt (หรือที่รู้จักกันในคนรุ่นเก่าว่า dos window) แล้วก็ set path บน (dos) Command Prompt



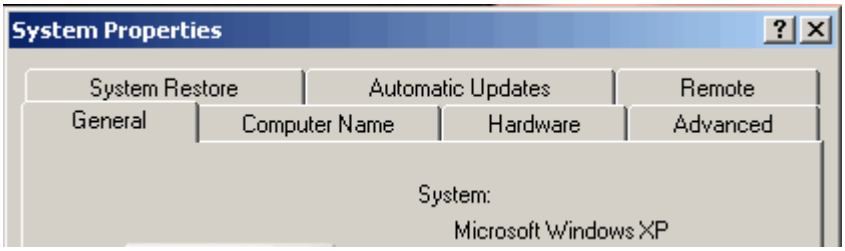
3. ทำการพัฒนาโปรแกรมใน directory ที่มี JDK อยู่ เช่นเปลี่ยน drive ไปที่ c:\j2sdk1.4.1\bin แล้วเริ่มทำงานใน directory นี้ ไฟล์ทุกตัวที่เกี่ยวข้องกับการเขียนโปรแกรมจะต้องเก็บไว้ที่นี่ – วิธีนี้ ไม่แนะนำ เพราะจะทำให้ directory ของ java เปรอะไปด้วยไฟล์ที่เราเขียนขึ้นเอง

ถ้าเราใช้ Windows XP หรือ Windows 2000 การกำหนด path ทำได้ด้วยขั้นตอนต่อไปนี้ (Windows XP และ Windows 2000 ใช้คำสั่งที่แตกต่างกันนิดหน่อย ตัวอย่างที่ทำให้ดูนี้ เป็นการกำหนด path บนเครื่องที่ใช้ Windows XP)

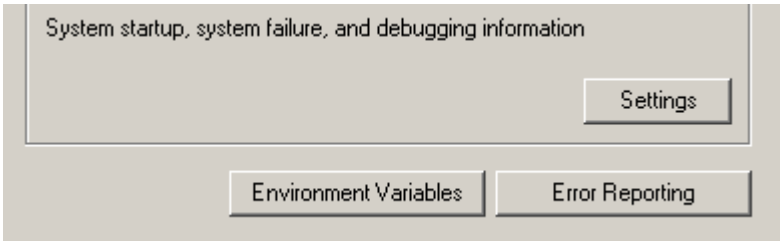
1. กดปุ่มขวาของ mouse บน icon My Computer (หรือ ไปที่ปุ่ม start ที่มุมล่างซ้ายของจอแล้วเลือก My Computer) แล้วเลือก Properties



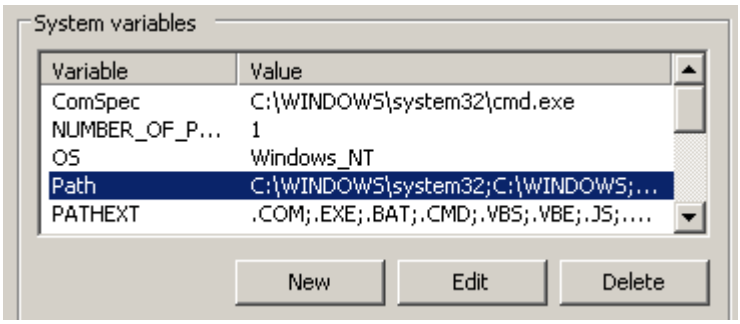
2. กดปุ่ม advanced บน System Properties



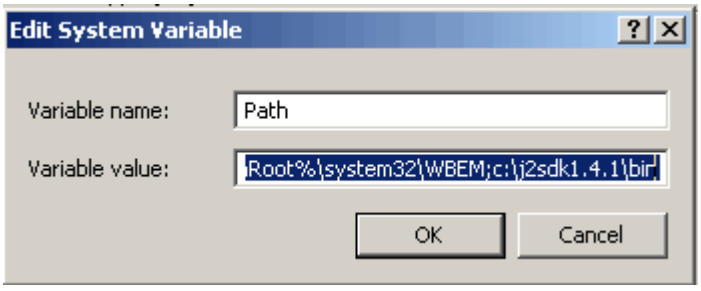
3. กดปุ่ม Environment Variables



4. ใน System Variables เลือกข้อความที่ขึ้นต้นด้วยคำว่า Path



5. เมื่อกดปุ่ม Edit ก็จะได้หน้าจอดังที่เห็น



6. เติม ; (semicolon) ถ้ายังไม่มี แล้วจึงใส่ path ที่ได้ install JDK ไว้ เสร็จแล้วกดปุ่ม OK เพื่อให้ระบบบันทึกการเปลี่ยนแปลงที่ได้ทำขึ้น

หลังจากที่ได้กำหนด path ให้กับเครื่องที่เราจะใช้ในการเขียนโปรแกรมแล้ว ขั้นตอนต่อไปก็ขึ้นอยู่กับผู้เขียน การพัฒนาโปรแกรมก็ทำได้โดยไม่ต้องวุ่นวายกับการกำหนด path ต่าง ๆ อีก (ยกเว้นเสียแต่ว่า path ที่ได้กำหนดขึ้นไม่มีอยู่จริง – เราก็ต้องกลับไปเริ่มต้นใหม่ พร้อมกับ path ที่มีอยู่จริง)

ทั้งนี้ทั้งนั้น ผู้พัฒนาโปรแกรมต้องเลือกเองว่า วิธีไหนเหมาะสมที่สุด ถ้าต้องใช้ Java อยู่บ่อย ๆ และสม่ำเสมอก็ควรที่จะเลือกกำหนด path แบบถาวร เพื่อจะได้ไม่ต้องเสียเวลากับการกำหนด path

สิ่งที่สำคัญอีกสิ่งหนึ่งที่ขาดไม่ได้ในการทดสอบและ execute code ที่เขียนขึ้นด้วย Java ก็คือ JRE หรือที่เรียกว่า Java Run-time Environment ถ้าเรานำเอา code ที่ได้ผ่านการ compile จาก JDK แล้วไป execute บนเครื่องที่ยังไม่ได้ติดตั้ง JRE ผลลัพธ์ก็คือ execute ไม่ได้ แต่ถ้าเครื่องนั้นได้รับการติดตั้ง JDK ก็จะได้รับการติดตั้ง JRE พร้อมกันไปด้วย ดังที่ได้กล่าวไว้ก่อนหน้านี้ สาเหตุที่ Sun ทำแบบนี้ก็เพราะว่าหลาย ๆ คนอาจไม่ใช่ผู้พัฒนา Java โปรแกรมแต่เป็นผู้ใช้โปรแกรมที่เขียนขึ้นด้วย Java ก็ได้ ดังนั้นหากเครื่องใด ๆ ต้องใช้โปรแกรมที่เขียนขึ้นด้วย Java บ่อย ๆ ก็ต้อง download JRE มาไว้ที่เครื่องด้วย

เขียน code ที่เป็น application หรือเขียน code ที่เป็น applet

Applet เป็น code ที่เขียนขึ้นมาเพื่อให้สามารถที่จะ execute ใน web browser ผ่านทาง HTML ไฟล์ ซึ่งโดยทั่วไปมักจะมีขนาดเล็ก เพื่อให้การ download applet ทำได้รวดเร็วยิ่งขึ้น ส่วน application เป็นการ execute ไฟล์ผ่านทาง command line ดังนั้นขนาดจึงไม่เป็นอุปสรรค (เพราะไม่ต้องมีการ download) หนังสือเล่มนี้จะแสดงโปรแกรมตัวอย่างด้วยการเขียน code ในรูปแบบของ application

เครื่องมือ หรือ คำสั่ง (SDK tools) ที่ต้องใช้ในการพัฒนาโปรแกรม

Java Development Kit ได้ถูกเปลี่ยนให้มีชื่อเป็น J2SDK – Java 2 Software Development Kit ดังนั้นเราจะใช้ชื่อเครื่องมือในการพัฒนาโปรแกรมนี้สลับกันไปมา แต่ให้ตั้งสมมติฐานว่าทั้งสอง ต่างก็เป็นเครื่องมือที่ใช้ในการพัฒนา Java โปรแกรมเช่นเดียวกัน

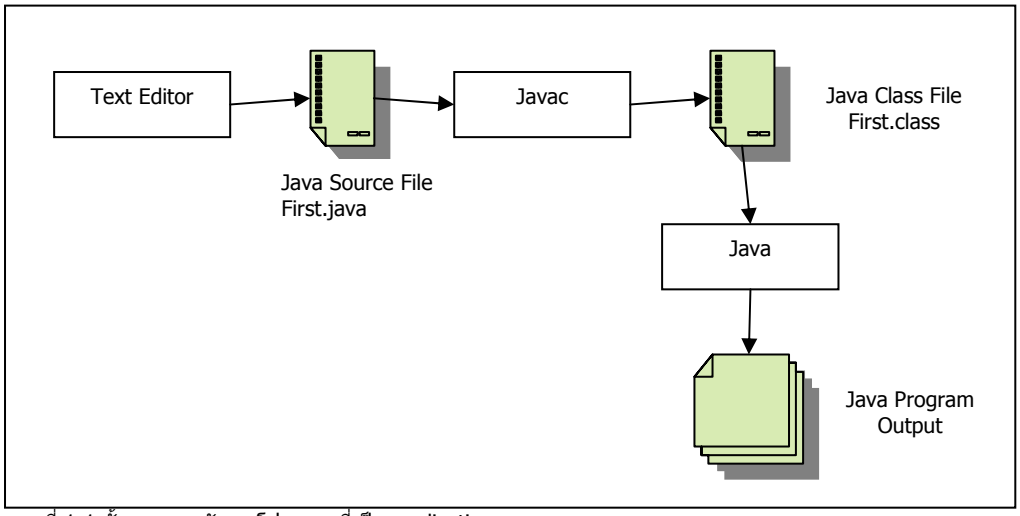
javac เป็นเครื่องมือที่ใช้ในการ compile (compiler) ที่ทำการเปลี่ยน source code ที่เราเขียนขึ้นให้เป็น byte code

java เป็นเครื่องมือที่ใช้ในการ execute byte code สำหรับโปรแกรมที่เขียนขึ้นในแบบของ application

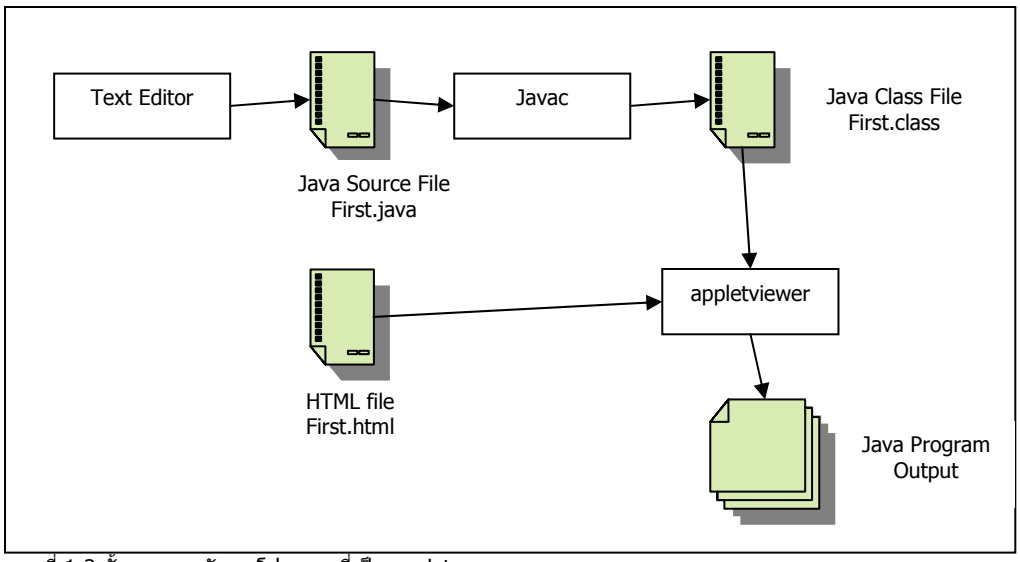
appletviewer เป็นเครื่องมือที่ใช้ในการ execute โปรแกรมที่เขียนขึ้นในแบบของ applet

ยังมีเครื่องมืออีกหลายตัวที่ java มีให้ แต่ตอนนี้เราจะใช้เฉพาะเครื่องมือที่ได้กล่าวไว้แค่สามตัวนี้ เท่านั้น

การพัฒนาโปรแกรมทั้งที่เป็น application และ applet นั้นจะมีขั้นตอนที่แตกต่างกันพอสมควร ดังแสดงให้เห็นในภาพที่ 1-1 และในภาพที่ 1-2 ในการเขียน code นั้นเราจะต้องมี editor ที่เป็น text editor กล่าวคือเป็น editor ที่ไม่มีการเก็บ format ต่าง ๆ ของตัวอักษรดังเช่นที่ Microsoft Word เก็บ แต่จะเก็บเป็นรหัส ASCII ที่ SDK สามารถอ่านและ execute ได้ สมมติว่าเรามี text editor และ SDK แล้วการพัฒนาก็เป็นไปตามขั้นตอนที่แสดงในภาพทั้งสอง



ภาพที่ 1-1 ขั้นตอนการพัฒนาโปรแกรมที่เป็น application



ภาพที่ 1-2 ขั้นตอนการพัฒนาโปรแกรมที่เป็น applet

การสร้างโปรแกรม Java ที่เป็น application

สมมติว่าเราเลือกใช้ text editor ตัวใดตัวหนึ่งที่มีอยู่มากมาย ทั้งที่ free และทั้งที่ต้องจ่ายเงินซื้อมา เราก็มาลองเริ่มเขียนโปรแกรมตัวแรก กันเลยดีกว่า

เพื่อให้เข้าใจได้ง่ายถึงขั้นตอนและที่มาที่ไปในการเขียนโปรแกรมด้วยภาษา Java เราจะเขียนโปรแกรมที่ไม่ทำอะไรมากมายนัก เพียงแต่ส่งข้อมูลไปยังหน้าจอเมื่อ user เรียกใช้ (execute) โปรแกรมนี้ เราจะตั้งชื่อโปรแกรมนี้ว่า HelloWorld.java และมีข้อมูลดังนี้

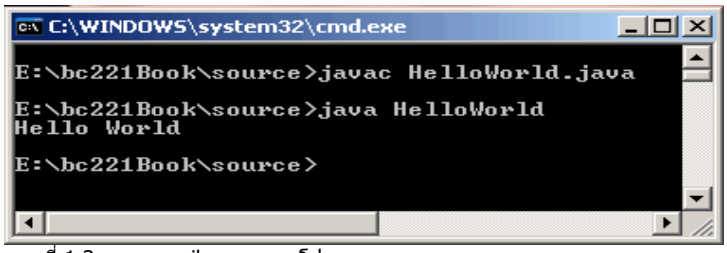
```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

ผู้อ่านยังไม่ต้องกังวลกับความหมายของ keyword ต่าง ๆ ที่เห็นในตอนนี้ หลังจากที่เขียนโปรแกรมได้สักพักหนึ่ง ผู้อ่านก็จะเข้าใจถึงความหมายเหล่านี้ดีขึ้น แต่ตอนนี้ต้องเข้าใจว่าไฟล์ที่เราเขียนขึ้น มีนิยามของ

class ที่เราเขียนขึ้นชื่อ HelloWorld ซึ่ง Java กำหนดไว้ว่า ชื่อของ class และชื่อของไฟล์ที่เก็บ code ของ class นี้จะต้องเป็นชื่อเดียวกัน

class นี้มี method อยู่หนึ่งตัวชื่อ main() เมื่อเรา execute โปรแกรมนี้ JRE จะค้นหา method ที่ชื่อ main() เพื่อทำการ execute ทั้งนี้ method main() จะต้องมี keyword ที่ชื่อ public, static, และ void อยู่ รวมไปถึง parameter ที่เป็น array ที่เก็บ String ด้วย ถ้าผู้อ่านเคยเขียนโปรแกรมด้วย ภาษา C หรือ ภาษา C++ มาก่อนก็จะเห็นถึงความคล้ายคลึงกันของ method main() นี้

Parameter ของ method main() ที่เป็น String[] args หมายถึง command-line argument ต่าง ๆ ที่ผู้ใช้โปรแกรมส่งผ่านไปยังตัวโปรแกรมเพื่อเอาไว้ใช้ในการทำงานต่าง ๆ ซึ่งเราจะยังไม่กล่าวถึงในเวลานี้ ในตัว method main() เองมีประโยคที่ขึ้นต้นด้วย System.out.println("...") ซึ่งเป็นคำสั่งที่ใช้ส่งข้อความไปยัง System.out ที่โดยทั่ว ๆ ไปก็คือ console window หรือที่เรียกกันติดปากว่า dos window หรือ dos prompt ดังตัวอย่างที่แสดงให้เห็นในภาพที่ 1-3



ภาพที่ 1.3 การ compile และ run โปรแกรม

การ compile โปรแกรม HelloWorld

ดังเช่นที่แสดงให้เห็นในภาพที่ 1-3 การ compile ไฟล์ที่เขียนขึ้นต้องใช้คำสั่ง javac ตามด้วยชื่อไฟล์พร้อมกับนามสกุล ดังนี้

```
javac HelloWorld.java
```

ในการ compile ทุกครั้งเราจะต้องไม่ลืมใส่นามสกุลให้กับไฟล์ที่เราได้เขียนขึ้น ไม่เช่นนั้นแล้ว compiler จะฟ้อง โดยการส่ง error message ให้เราเช่น ถ้าเรา compile ไฟล์ HelloWorld โดยไม่ใส่ .java เราก็จะได้ error ดังนี้

```
E:\bc221Book\source>javac HelloWorld
javac: invalid flag: HelloWorld
Usage: javac <options> <source files>
where possible options include:
  -g                        Generate all debugging info
  -g:none                   Generate no debugging info
  -g:{lines,vars,source}    Generate only some debugging info
    …
    …
    …
```

(ผู้เขียนได้ตัดข้อความบางส่วนที่ java ฟ้องออก - ใช้ ... แทน)

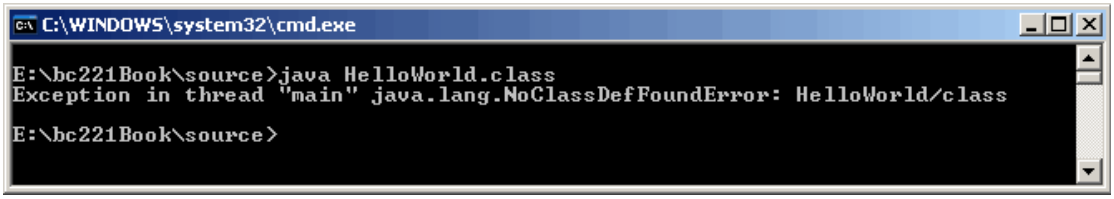
การ execute โปรแกรม HelloWorld

สมมติว่าเรา compile โปรแกรม HelloWorld โดยไม่มี error ใด ๆ เกิดขึ้นการ execute โปรแกรมก็ทำได้ด้วยคำสั่ง java ตามด้วยชื่อไฟล์ ดังนี้

```
java HelloWorld
```

และก็จะได้ output ดังที่เห็นในภาพที่ 1-3

ในการ execute โปรแกรมนั้นเราจะไม่ใส่ .class ตามหลังชื่อไฟล์ (ถึงแม้ว่าเราต้องมีไฟล์นี้ก็ตาม) ถ้าเราใส่ compilerก็จะฟ้องด้วย error ที่บอกว่า หาไฟล์ไม่เจอ (java.lang.NoClassDefFoundError) ดังที่เห็นนี้



สรุปขั้นตอนของการพัฒนาโปรแกรมที่เป็น application

1. compile ด้วยคำสั่ง javac ตามด้วยชื่อไฟล์ พร้อมกับนามสกุล

2. execute ด้วยคำสั่ง java ตามด้วยชื่อไฟล์โดยไม่มีนามสกุลใด ๆ ต่อท้าย

การเขียนโปรแกรมที่ดีนั้น ผู้เขียนควรใส่คำอธิบาย (comment) ถึงการทำงานของโปรแกรมที่เขียนขึ้น เพื่อให้ผู้ที่นำโปรแกรมไปใช้ หรือผู้อ่านคนอื่น เข้าใจถึงวิธีการใช้ที่ถูกต้อง ในการเขียน comment ด้วยภาษา Java นั้น มีการทำได้สองวิธี คือ

1. การใช้ //
2. การใช้ /* … */

การใช้ในแบบที่หนึ่งนั้น เป็นการใส่ comment ได้ไม่เกิน 1 บรรทัด ส่วนการใช้ในแบบที่สอง ผู้ใช้สามารถที่จะใส่ comment ได้มากกว่าหนึ่งบรรทัด แต่ comment ต้องอยู่ภายในเครื่องหมาย /* … */ เท่านั้น ดังตัวอย่างจากการนำโปรแกรม HelloWorld มาเขียนใหม่ด้วย comment

```
// My first Java program - HelloWorld
class HelloWorld {
    /* Java launcher will call this method (main).
       This will display a message to the screen.
    */
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

การสร้างโปรแกรม Java ที่เป็น applet

ในการเขียนโปรแกรมที่เป็น Java applet นั้นเราจะต้องมีไฟล์อยู่อย่างน้อย 2 ไฟล์ ซึ่งหนึ่งในนั้นคือ Java source file และอีกไฟล์หนึ่งคือ HTML file เราลองมาเขียน applet เล็ก ๆ ที่ส่งข้อความไปยัง web browser เหมือนกับที่เราเขียน application ก่อนหน้านี้

```
//my first Java applet - HelloWorldApplet
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorldApplet extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello World", 25, 25);
    }
}
```

เนื่องจากว่าเราต้องการ applet สำหรับการส่งข้อความไปยัง web browser ดังนั้นเราจึงต้องดึงเอาเครื่องมือที่ช่วยให้เราสามารถแสดงข้อความที่ว่า ด้วยการใช้การถ่ายทอดคุณสมบัติ (inheritance) ที่ Java มีให้ด้วยการเพิ่มคำว่า extends Applet ทางด้านหลังของ class HelloWorldApplet เราจะยังไม่พูดถึงรายละเอียดเกี่ยวกับการถ่ายทอดคุณสมบัติของ Java ตอนนี้ อีกสิ่งหนึ่งที่ขาดไม่ได้ในการเขียน applet ของเราตอนนี้ก็คือ method paint() ซึ่งเป็น method ที่ทำการวาดข้อความลงบน applet ด้วยการเรียกใช้ method drawstring() จาก class Graphics ในตำแหน่งที่กำหนดไว้ จากคำสั่งนี้

```
g.drawString("Hello World", 25, 25);
```

เมื่อ compile โปรแกรม HelloWorldApplet ด้วยคำสั่ง javac แล้วเราก็สามารถที่จะ execute applet ได้ด้วยการเรียก HTML ไฟล์ที่เขียนขึ้น ดังนี้

```
<html>
<applet code="HelloWorldApplet.class" width="200" height="80"></applet>
</html>
```

ด้วยการใช้ applet tag ใน HTML ไฟล์พร้อมกับการใส่ชื่อของ Java ไฟล์ที่ compile แล้วในส่วนของ field ที่ชื่อ code และกำหนดขนาดความกว้างของ applet ใน field ที่ชื่อ width และความยาว (สูง) ใน field ที่ชื่อ height

การ execute ไฟล์ HelloWorldApplet.html ก็สามารถทำได้ด้วยการใช้คำสั่ง appletviewer ตามด้วยชื่อไฟล์ เราก็จะได้ output ดังนี้

สรุปขั้นตอนการเขียนโปรแกรมแบบ applet

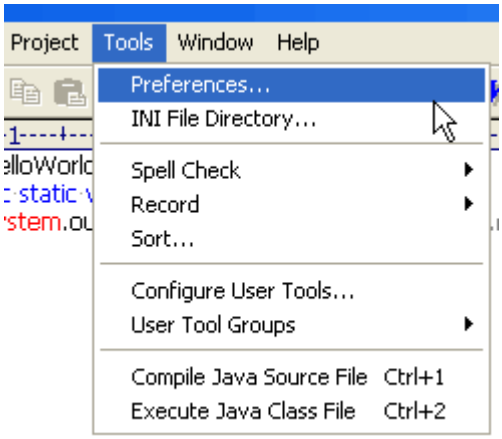
1. สร้าง Java source file ด้วยการ extends Applet
2. สร้าง HTML file ที่มี applet tag และข้อมูลที่สำคัญในการแสดง applet คือ (1) class file ของ Java โปรแกรม (2) ขนาดของ applet – width และ height

เราจะกลับมาดูวิธีการเขียน applet อีกครั้งหนึ่งหลังจากที่เรา ได้ศึกษาถึงโครงสร้างต่าง ๆ ของภาษา Java ขั้นตอนการออกแบบ และการเรียกใช้ class รวมไปถึงเทคนิคและวิธีการใช้ method ต่าง ๆ ที่ Java มีให้

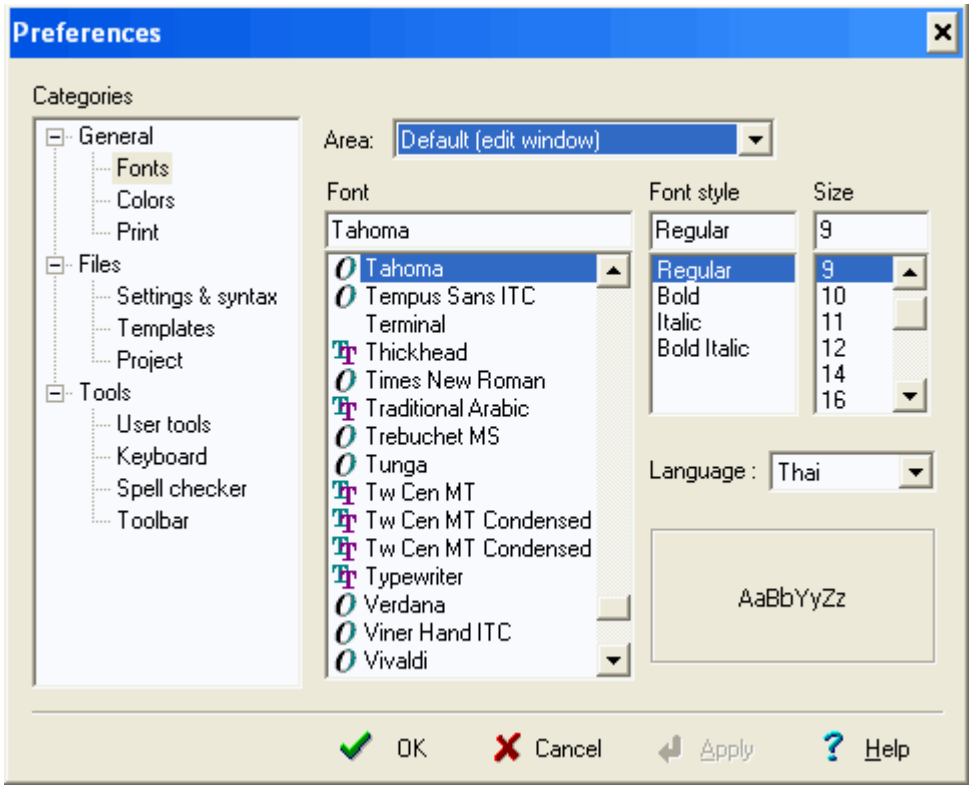
การเขียน Java program ด้วย EditPlus

ตัวอย่างของโปรแกรม HelloWorld.java ที่เราได้เขียนขึ้นก่อนหน้านี้ใช้ภาษาอังกฤษทั้งหมดในการเขียน ผลลัพธ์ที่ได้จากการ run โปรแกรมก็เป็นภาษาอังกฤษ ผ่านทาง Dos Window ถ้าเราต้องการที่จะเขียนโปรแกรมเพื่อให้แสดงผลลัพธ์ หรือ ข้อความที่เป็นภาษาไทยนั้น จะต้องใช้ Operating System ที่รองรับการป้อนและส่งข้อมูลที่เป็นภาษาไทย เช่น Windows ต่าง ๆ ที่สร้างขึ้นมาเพื่อคนไทยโดยเฉพาะ ซึ่งจะสังเกตได้จากคำว่า Thai Edition หรือ คำว่า ไทย ต่อท้าย แต่ก็มี OS หลาย ๆ ตัวที่ยอมให้มีการป้อนและส่งข้อมูลที่เป็นภาษาไทยผ่านทางช่องทางอื่น ๆ ที่ไม่ใช่ Dos Window ในการฝึกฝนการเขียนโปรแกรม Java ใหม่ ๆ นั้นเราจำเป็นที่จะต้องมีช่องทางในการแสดงผลที่สามารถใช้ได้ทั้งภาษาไทย และภาษาอังกฤษ ดังนั้นเพื่อให้การป้อนและส่งข้อมูลใช้ได้ทั้งภาษาไทยและภาษาอังกฤษ เราจะเลือกใช้ Text Editor ที่รองรับภาษาไทย เช่น EditPlus (มี Text Editor หลายตัวที่รองรับภาษาไทย แต่เราจะพูดถึงเฉพาะ EditPlus เท่านั้น)

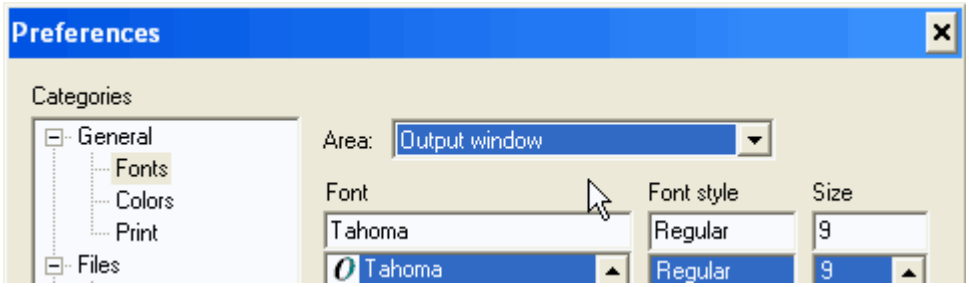
ก่อนอื่นเราต้องหา EditPlus มาติดตั้งในเครื่องที่เราใช้ในการสร้างโปรแกรมด้วยภาษา Java เสียก่อน ซึ่งสามารถที่จะไป download มาได้จาก www.editplus.com โปรแกรมตัวนี้เป็น shareware ที่มีอายุการใช้งานเพียงแค่ 30 วันหลังจากนั้นก็คงต้องควักกระเป๋าจ่ายให้กับเจ้าของโปรแกรมตามระเบียบ ซึ่งก็ไม่แพงเท่าไรนัก เมื่อ install เรียบร้อยแล้วก็ให้เข้าไปใน menu ที่ชื่อว่า Tools



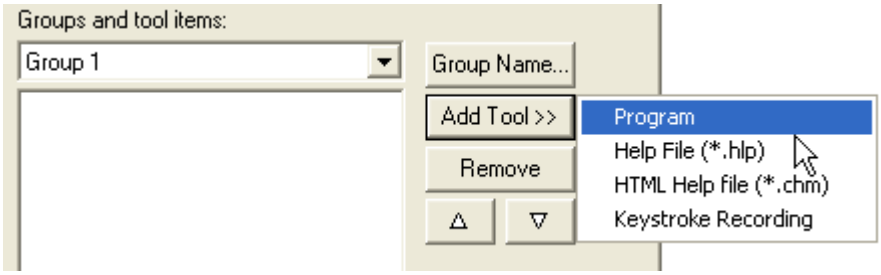
เลือก Preferences



click ที่ Fonts และเลือก Default (edit window) จาก Drop-down menu ในช่องที่ชื่อ Area เสร็จแล้วให้เลือก font ที่สามารถใช้ภาษาไทยได้ เช่น Tahoma ดังที่เห็นในรูป เสร็จแล้วก็เริ่มทำในลักษณะเดียวกันกับ Output window



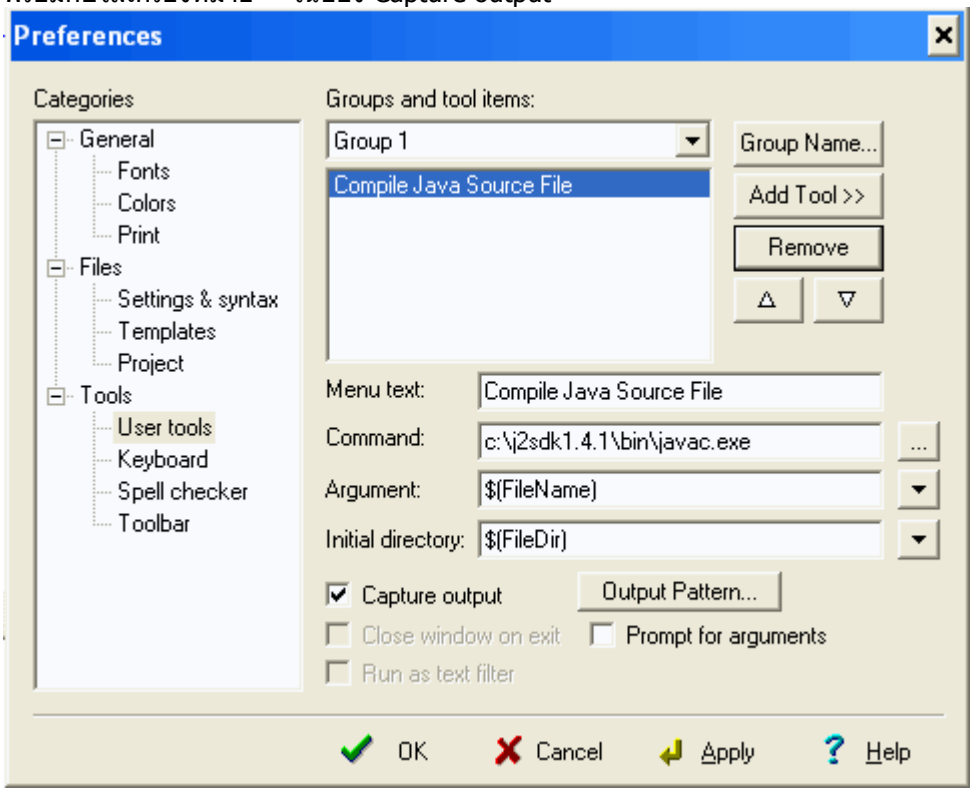
หลังจากที่ set ค่าให้กับ editor และ output window เรียบร้อยแล้ว เราก็ต้องมา set ให้ EditPlus รู้จักกับการ compile และ execute โปรแกรมในภาษา Java เริ่มต้นด้วยการ click ที่ User tools ใน Preferences และกดปุ่ม Add Tool>> เพื่อเลือก Program



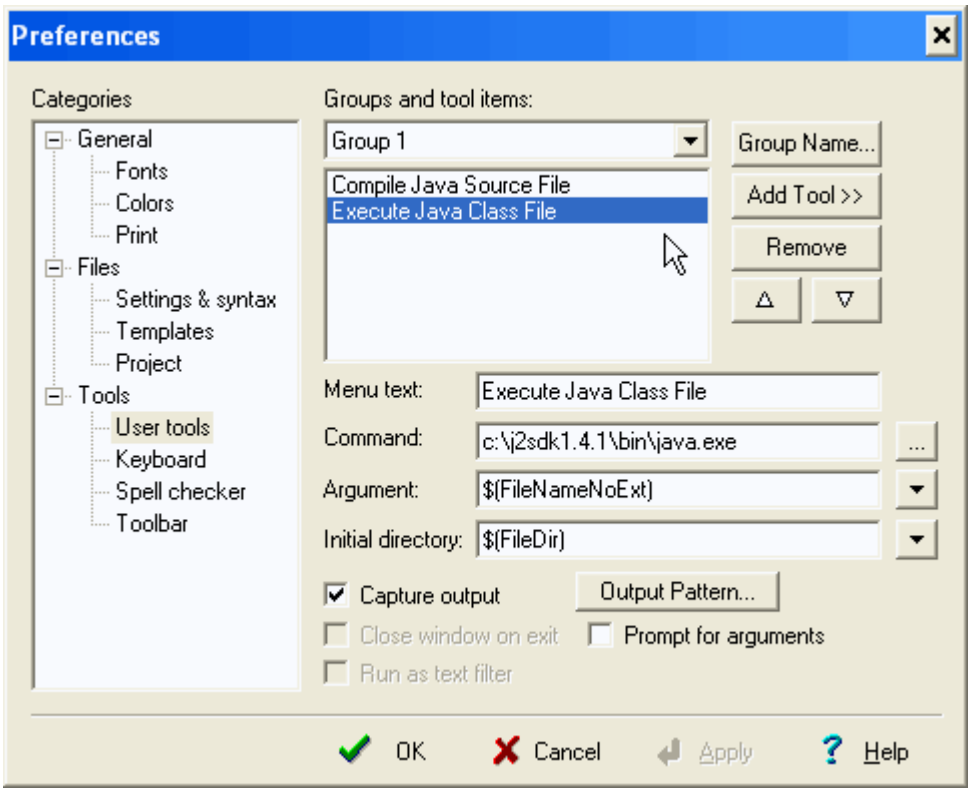
เสร็จแล้วให้ใส่ข้อมูลดังตัวอย่างที่เห็น

Menu Text: *Compile Java Source File*
Command: ที่ที่ผู้อ่านได้ install Java ไว้ เช่น *c:\j2sdk1.4.1\bin\javac.exe*
Argument: *$(FileName)* เลือกจาก drop-down list
Initial directory: *$(FileDir)* เลือกจาก drop-down list

พร้อมกับใส่เครื่องหมาย ✓ ในช่อง Capture output



เสร็จแล้วให้เพิ่มคำสั่งให้ EditPlus รู้จักการ run หรือ execute โปรแกรมที่ได้ compile เรียบร้อยแล้ว ด้วยการกดปุ่ม Add tool>> อีกครั้งหนึ่ง พร้อมกับใส่ข้อมูลที่เห็นนี้

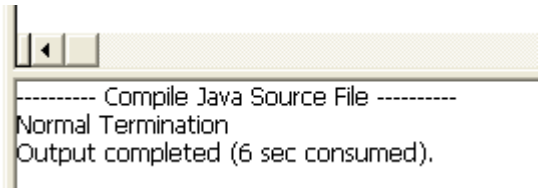


เมื่อทำทุกอย่างเรียบร้อยแล้วให้กดปุ่ม OK เพื่อทำการบันทึกคำสั่งที่เราสร้างขึ้นให้ EditPlus รู้จัก

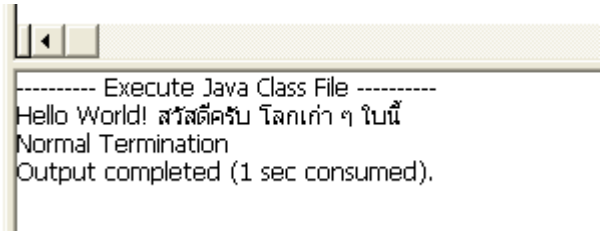
และเมื่อได้เปลี่ยนแปลงไฟล์ HelloWorld.java ด้วยการเพิ่มข้อความที่เป็นภาษาไทยเข้าไปในประโยค System.out.println(...) เช่นในตัวอย่างนี้

```
class  HelloWorld {
    public static void main(String[] args)  {
        System.out.println("Hello World! สวัสดีครับ โลกเก่า ๆ ใบนี้");
    }
}
```

เราก็ compile ด้วยการกด Ctrl + 1 และเมื่อ compile เสร็จเราก็จะได้รับข้อความดังที่เห็นนี้



หลังจากนั้นเราก็ run ด้วยการกด Ctrl + 2 ก็จะได้ผลลัพธ์ดังที่เห็นนี้



EditPlus ยังมี features ต่าง ๆ อีกมากมายที่เราสามารถที่จะ set ไว้ช่วยในการทำงานของเราได้ ทั้งนี้ผู้อ่านสามารถที่จะหาข้อมูลเพิ่มเติมได้จาก help file ของตัว EditPlus เอง แต่ถ้าผู้อ่านอยากใช้ Text Editor ตัวอื่นผู้เขียนขอแนะนำอีกตัวหนึ่งคือ Crimson ซึ่งเป็น editor ที่ไม่ต้องเสียค่าใช้จ่ายใด ๆ เลย และสามารถที่จะ download ได้จากหลาย ๆ ที่ใน WWW

ที่มาของภาษา Java โดยสังเขป

Java เป็นภาษาที่เกิดขึ้นจากการพัฒนาของบริษัท Sun Microsystems โดยมีที่มาจากภาษาที่ชื่อว่า Oak ซึ่งเป็นภาษาที่ใช้ภายใน Sun เองแต่หลังจากที่ Sun ได้พัฒนา Oak และใช้มาอีกระยะหนึ่ง Sun ก็ได้นำเอา Oak ออกมาสู่สายตาชาวบ้านทั่วไป แต่เนื่องจากว่า ชื่อของ Oak ได้มีผู้ใช้อยู่ก่อนแล้ว Sun จึงต้องหาชื่อใหม่ให้กับภาษานี้ ซึ่งในตอนแรกก็ได้ทดลองหลายชื่อ เช่น Silk, Ruby, และ WRL (Web Runner Language) แต่ชื่อเหล่านี้ก็ไม่ได้ถูกเลือก ในที่สุด Java ก็กลายเป็นชื่อของภาษานี้ (ทั้ง ๆ ที่ไม่มีอะไรเกี่ยวข้องกับ กาแฟ จาก ชวา หรืออะไรที่เกี่ยวข้องกับเกาะชวาเลย)

Java มีอยู่มากมายหลาย version แต่ version ล่าสุดของ Java ขณะที่เขียนตำราเล่มนี้ คือ J2SDK1.4.1 ผู้อ่านสามารถติดตาม version ใหม่ ๆ และ download ได้ที่ web site ของ Sun ที่ได้ให้ไว้ก่อนหน้านี้

ในบทที่สองเราจะมาทำความรู้จักกับข้อกำหนด ต่าง ๆ ของ โปรแกรม ข้อมูล (data) ตัวแปร (variable) และ การประมวลผล (calculation – evaluation)

2
ข้อมูล ตัวแปร และ การประมวลผล

ในบทที่สองนี้เราจะมาทำความเข้าใจในเรื่องของชนิดข้อมูลที่ Java มีไว้ในการรองรับการเขียนโปรแกรม ทั้งข้อมูลที่เป็นตัวเลข และข้อมูลที่เป็นตัวอักษร เราจะทดลองเขียนโปรแกรมด้วยข้อมูลชนิดต่าง ๆ นี้

หลังจากจบบทเรียนนี้แล้ว ผู้อ่านจะได้ทราบถึง

- วิธีการประกาศตัวแปร (variable declaration) ที่มีชนิดเป็น integer และ floating-point
- วิธีการประกาศตัวแปรที่มีชนิดเป็น character และ boolean
- วิธีการกำหนด (assignment) ค่าให้กับตัวแปรต่าง ๆ
- การสร้างประโยค (statement หรือ expression)
- การประมวลผล (calculation และ evaluation) ที่เกี่ยวกับตัวแปรชนิดต่าง ๆ
- การเปลี่ยนแปลง (casting) ชนิดของข้อมูล
- การใช้ฟังค์ชั่นทางคณิตศาสตร์ (Mathematical Functions) ที่ Java มีให้

ข้อมูล และ ตัวแปร (data and variable)

โดยทั่วไปถ้าเราต้องการคำนวณหาค่าทางคณิตศาสตร์ เช่น 354 + 45 * 12 เรามักจะใช้เครื่องคิดเลข หรือไม่ก็ใช้วิธีคิดในกระดาษ ซึ่งวิธีคิดแบบที่สองนี้เราใช้กระดาษเป็นตัวกลางในการหาค่านั้น ๆ กล่าวคือ เราใช้กระดาษเป็นที่เก็บตัวเลขที่ต้องการหาค่า รวมไปถึงผลลัพธ์ของการหาค่านั้น ๆ ด้วย เช่นเดียวกับการเขียนโปรแกรมทางคอมพิวเตอร์ ค่าต่าง ๆ เหล่านี้จำเป็นจะต้องมีที่เก็บ และที่เก็บข้อมูลเหล่านี้ต้องเป็นที่เก็บที่สามารถรองรับการประมวลผล หรือ การคำนวณได้อย่างพอเพียง

ดังนั้นในการเขียนโปรแกรม เราจึงต้องใช้ตัวแปรเป็นที่เก็บข้อมูลต่าง ๆ ที่เราจะต้องใช้ในการประมวลผล ตัวแปรที่ว่านี้เป็นสัญลักษณ์แทนหน่วยความจำ (memory) ที่อยู่ในเครื่อง และ compiler จะเป็นผู้กำหนดว่าอยู่ที่ใด มีขนาดเท่าใด (address เริ่มต้น และ offset) ที่เก็บข้อมูลนี้จะเก็บข้อมูลได้เพียงชนิดที่เราได้กำหนดไว้ เพียงชนิดเดียวเท่านั้น ข้อมูลถูกแบ่งออกตามชนิดของข้อมูล หลัก ๆ คือ ข้อมูลที่เป็นตัวเลข และ ข้อมูลที่ไม่ใช่ตัวเลข แต่จะแบ่งย่อย ๆ ลงไปอีก เช่น ถ้าเรากำหนดให้ตัวแปรตัวใดตัวหนึ่งเก็บข้อมูลชนิดที่เป็นตัวเลขในแบบของ integer เราก็ไม่สามารถที่จะนำเอาข้อมูลที่เป็นตัวเลขในรูปแบบอื่น เช่น double หรือ float มาเก็บไว้ในตัวแปรนี้ได้ เนื่องจากว่าข้อมูลที่ตัวแปรเก็บได้นั้นมีขนาดตายตัว ตามการออกแบบของภาษา ดังนั้น การนำเอาข้อมูลชนิดอื่นมาใส่ไว้ในตัวแปรนี้จะถูกตรวจสอบโดย compiler และ compiler จะฟ้องไปยังผู้เขียนโปรแกรมทันที

การประกาศตัวแปรนั้น ผู้เขียนโปรแกรมจะใช้สัญลักษณ์ใด ๆ ก็ได้ที่ Java ยอมให้ใช้ ซึ่งต้องอยู่ในกฎเกณฑ์ ต่อไปนี้

1. ต้องขึ้นต้นด้วยตัวอักษร (letter) หรือ เครื่องหมาย _ (underscore) หรือ เครื่องหมาย $ (dollar sign)

2. เครื่องหมายอื่น ๆ ที่ตามมาเป็นได้ทั้ง ตัวอักษร ตัวเลข และ สัญลักษณ์อื่น ๆ ที่ไม่ได้ถูกใช้โดย Java (เช่น operator ต่าง ๆ ที่ Java ใช้ดังตัวอย่างของการ + (บวก) – (ลบ) * (คูณ) / (หาร) เป็นต้น)

ตัวอย่างของตัวแปรที่ถูกต้อง

taxRate
pricePerUnit

_timeInSecond
$systemClock
unit897

เนื่องจากว่า Java ได้ออกแบบให้ผู้ใช้สามารถที่จะกำหนดตัวแปรให้เป็น Unicode[1] เพื่อเปิดโอกาสให้การออกแบบตัวแปรเป็นไปได้ในภาษานั้น ๆ ของ ผู้เขียนโปรแกรม เราจึงสามารถที่จะใช้ตัวแปรที่เป็นภาษาไทยได้ สิ่งที่เราต้องคำนึงถึงก็คือ text editor ที่เราใช้ในการเขียน code ต้องยอมให้เราใช้ภาษาไทยด้วย ในบทนี้และบทอื่น ๆ ของตำรานี้จะใช้ตัวแปรที่เป็นภาษาอังกฤษเท่านั้น เพื่อให้ง่ายต่อการเขียน และการทำความเข้าใจในเนื้อหาได้รวดเร็วขึ้น

โดยทั่วไปเราจะเรียกตัวแปรที่ได้กำหนดให้เก็บข้อมูลชนิดใด ชนิดหนึ่งว่า variable หรือ identifier และ identifier ที่ว่านี้ มีความยาวเท่าไรก็ได้ (จำนวนของตัวอักษร) แต่ก็ต้องอยู่ภายใต้ข้อกำหนดที่ได้กล่าวมาแล้ว พร้อมกันนี้ ตัวแปรที่สร้างขึ้นจะต้องไม่ใช่ keyword หรือ reserved word ที่ Java ได้สร้างขึ้นไว้ใช้งาน ดังที่แสดงในตาราง 2.1

ตารางที่ 2.1 keyword

abstract	const	finally	int	public	this
boolean	continue	float	interface	return	throws
break	default	for	long	short	throws
byte	do	goto	native	static	transient
case	double	if	new	strictfp	try
catch	else	implements	package	super	void
char	extends	import	private	switch	volatile
class	final	instanceof	protected	synchronized	while

ชนิดของข้อมูล (datatype)

ในการประกาศตัวแปรทุกครั้งเราจะต้องบอกชนิดของข้อมูลให้กับ ตัวแปรนั้นด้วย ซึ่งการประกาศตัวแปรนั้น ต้องเริ่มต้นด้วย ชนิดของข้อมูล และตามด้วย ชื่อของตัวแปร ตามรูปแบบที่กำหนดดังนี้

datatype identifier

โดยที่ datatype คือชนิดของข้อมูลที่ ตัวแปรสามารถเก็บได้ และในภาษา Java *datatype* เป็นได้ทั้งข้อมูลที่เรียกว่า primitive datatype และข้อมูลที่เป็น reference (เราจะพูดถึง reference ในบทอื่น) Java กำหนดให้มี primitive datatype จำนวนทั้งสิ้น 8 ชนิด ดังที่แสดงให้เห็นในตารางที่ 2.2

ตาราง 2.2 primitive datatype

datatype	**คำอธิบาย**	**ขนาด (bit)**
boolean	ค่าทางตรรกะ	8 มีค่าเป็น true หรือ false
byte	ตัวเลขที่ไม่มีจุดทศนิยม	8
short	ตัวเลขที่ไม่มีจุดทศนิยม	16
int	ตัวเลขที่ไม่มีจุดทศนิยม	32
long	ตัวเลขที่ไม่มีจุดทศนิยม	64
float	ตัวเลขที่มีจุดทศนิยม	32
double	ตัวเลขที่มีจุดทศนิยม	64
char	ตัวอักษร	16

ขนาดของ datatype จะเป็นตัวกำหนดค่าที่เราสามารถเก็บได้ในตัวแปรที่ได้ประกาศในโปรแกรม ยิ่งจำนวนของ bit มากเท่าไรเราก็สามารถเก็บค่าสูง ๆ ได้ ตารางที่ 2.3 แสดงค่าต่าง ๆ ที่สามารถเก็บได้ใน datatype ชนิดต่าง ๆ

[1] มีความจุ 2 byte เพื่อรองรับภาษาที่ใช้กันในประเทศต่าง ๆ (เดิมใช้ ASCII ซึ่งมีความจุเพียง 1 byte)

ตาราง 2.3 ค่าที่เก็บได้ใน primitive datatype ต่าง ๆ

datatype	**ค่าต่ำสุด**	**ค่าสูงสุด**
byte	-128	127
short	-32768	32767
int	-2147483648	2147483647
long	-9223372036854775808	9223372036854775807
float	-3.4×10^{38}	3.4×10^{38}
double	-1.7×10^{308}	1.7×10^{308}

ตัวอย่างการประกาศตัวแปร

```
int numberOfBikes;
float taxRate;
double interests, PI;
boolean yesOrNo;
char response;
```

โดยทั่วไปนักพัฒนาโปรแกรมมักจะใช้ ตัวอักษรตัวเล็กเป็นตัวเริ่มต้นของตัวแปร และจะเปลี่ยนเป็นอักษรตัวใหญ่เมื่อคำนั้น ๆ สิ้นสุดลง เช่นการประกาศตัวแปรที่เห็นด้านบนนี้ อีกวิธีหนึ่งที่หลาย ๆ คนชอบใช้ คือ การใช้เครื่องหมาย _ (underscore) เป็นตัวแบ่งคำ เช่น

```
int number_of_bikes;
float tax_rate;
double cum_gpa;
boolean yes_or_no;
```

ทั้งนี้และทั้งนั้น การประกาศตัวแปรทั้งสองวิธีทำให้การอ่านตัวแปรทำได้ง่ายขึ้น แต่สิ่งสำคัญที่เราจำเป็นจะต้องคำนึงถึงในการประกาศตัวแปรก็คือ การเลือกใช้คำที่สื่อถึงความหมายของข้อมูลที่ตัวแปรนั้น ๆ รองรับ หรือ เก็บอยู่ เช่น ถ้าเราต้องการเก็บข้อมูลที่เป็นเกรดที่เป็นตัวอักษร เราก็ควรใช้ ตัวแปรที่สื่อถึงเกรดที่เป็นตัวอักษร เช่น คำว่า letterGrade และถ้าเราต้องการเก็บเกรดที่เป็นค่าเฉลี่ยเราก็ควรใช้ ตัวแปร เช่น gpa หรือ cumulativeGPA เป็นต้น

การกำหนดค่าให้กับตัวแปร (Assignment)

การกำหนดค่าให้กับตัวแปรนั้นทำได้ด้วยการใช้เครื่องหมาย = (เรียกกันว่า assignment operator) ซึ่งมีความหมายว่า "*ให้นำค่าทางขวามือ (rvalue) ของเครื่องหมาย = ไปเก็บไว้ในตัวแปรที่อยู่ทางด้านซ้ายมือ (lvalue) ของเครื่องหมาย =*" rvalue สามารถเป็นได้ทั้ง ค่าคงที่ ตัวแปร หรือประโยคใด ๆ ที่สามารถหาค่าได้ ส่วน lvalue เป็นได้เฉพาะตัวแปรเท่านั้น เพราะจะต้องเป็นที่ที่เก็บข้อมูลได้ (physical memory) เช่น

```
numberOfBikes = 2;
area = PI * r * r;
taxReturned = calculateTaxReturned(income);
```

แต่ไม่ใช่แบบนี้

```
2 = numberOfBikes;
PI * r * r = area;
calculateTaxReturned(income) = taxReturned;
```

ในการกำหนดค่าให้กับตัวแปรนั้น ถ้าเรามองในมุมมองของที่มาของค่าที่นำมาใส่ให้กับตัวแปร เราก็สามารถที่จะแบ่งการกำหนดค่าได้เป็นสองแบบ คือ (1) การกำหนดค่าให้กับตัวแปรภายในตัวโปรแกรม และ (2) การกำหนดค่าให้กับตัวแปรผ่านทางสื่อการนำเข้าข้อมูลมาตรฐาน (standard I/O channel) เราจะมาทำความเข้าใจการกำหนดค่าในแบบที่หนึ่งกันก่อน เพราะเป็นการกำหนดค่าที่ง่ายที่สุด

การกำหนดค่าให้กับตัวแปรภายในโปรแกรม

การกำหนดค่าให้กับตัวแปรนั้นสำคัญมาก เพราะโดยทั่วไปตัวแปรจะไม่มีค่าใด ๆ เลยหลังจากการประกาศ ภาษาในการเขียนโปรแกรมหลาย ๆ ภาษาบังคับให้ผู้เขียนโปรแกรมกำหนดค่าเบื้องต้น ก่อนที่จะใช้ตัวแปรนั้น ๆ ซึ่งก็เป็นสิ่งที่ดี เพราะจะทำให้การเขียนโปรแกรมเป็นไปได้ง่าย และสะดวกต่อการตรวจสอบหากมีข้อผิดพลาดเกิดขึ้น โปรแกรมตัวอย่างที่เห็นนี้แสดงถึงวิธีการประกาศตัวแปรพร้อมกับการกำหนดค่าเบื้องต้นให้กับตัวแปรเหล่านั้น

```
/*
 * Variables.java - Variable decalration and initialization
*/

 class Variables {
    public static void main(String[] args) {
        boolean booleanVar = true;
        byte byteVar = 0;
        short shortVar = 0;
        int intVar = 0;
        long longVar = 0L;
        float floatVar = 0.0F;
        double doubleVar = 0.0D;
        char charVar = 'A';

        System.out.println("boolean variable and its initial
                            value is " + booleanVar);
        System.out.println("byte variable and its initial
                            value is " + byteVar);
        System.out.println("short variable and its initial
                            value is " + shortVar);
        System.out.println("int variable and its initial
                            value is " + intVar);
        System.out.println("long variable and its initial
                            value is " + longVar);
        System.out.println("float variable and its initial
                            value is " + floatVar);
        System.out.println("double variable and its initial
                            value is " + doubleVar);
        System.out.println("char variable and its initial
                            value is " + charVar);
    }
 }
```

เรากำหนดค่าที่เป็นศูนย์ให้กับตัวแปรทุกตัว ยกเว้นตัวแปรที่เป็น boolean ซึ่งเรากำหนดค่าให้เป็น true เราบอกให้ Java รู้ว่าค่าที่กำหนดให้กับตัวแปรต่าง ๆ มีชนิดเป็นอะไรได้ด้วยการใส่ตัวอักษรที่บ่งบอกชนิดนั้น ๆ ของตัวแปร เช่น 0L 0.0F และ 0.0D ซึ่งหมายถึง ค่าของศูนย์ที่เป็น long ค่าของศูนย์ที่เป็น float และ ค่าของศูนย์ที่เป็น double ตามลำดับ ทั้งนี้เพื่อบอกให้ Java รู้ว่าเราต้องการใช้ค่าที่เก็บชนิดนั้น ๆ จริง ซึ่งเราจำเป็นที่จะต้องทำเพียงค่าของตัวแปรที่เป็น float และ double เท่านั้น เพราะว่าตัวแปรที่มีชนิดเป็น float ถ้ามีการกำหนดค่า Java จะเหมาเอาว่าเป็น double เราจึงต้องกำหนดค่าตามชนิดของตัวแปรจริง ๆ ที่เราต้องการ

ผลลัพธ์ของโปรแกรม Variables.java

```
boolean variable and its initial value is true
byte variable and its initial value is 0
short variable and its initial value is 0
int variable and its initial value is 0
long variable and its initial value is 0
```

```
float variable and its initial value is 0.0
double variable and its initial value is 0.0
char variable and its initial value is A
```

เรามาดูขั้นตอนที่ Java เก็บข้อมูลในตัวแปรกันดีกว่า (หากจะบอกว่าภาษาอื่น ๆ ก็เก็บแบบนี้ก็คงจะไม่ผิดนัก) เราจะใช้ตัวแปรจากโปรแกรมตัวอย่างข้างบน จากประโยคของการประกาศและกำหนดค่า

```
long longVar = 0L;
```

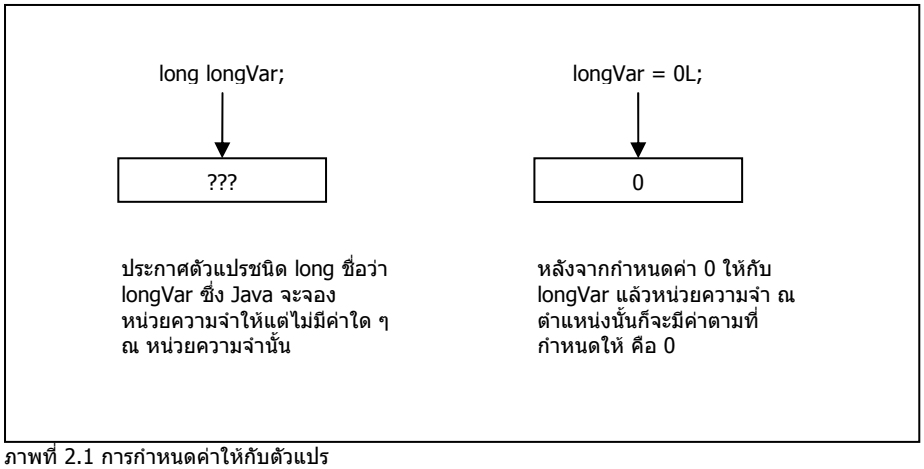
ประโยคที่เห็นด้านบนนี้สามารถที่จะเขียนได้อีกแบบหนึ่งคือ

```
long longVar;
longVar = 0L;
```

ประโยค long longVar; นั้นเป็นการประกาศตัวแปรชื่อ longVar ที่มีชนิดเป็น long ส่วนประโยค longVar = 0L; นั้นเป็นการกำหนดค่า 0 ให้กับตัวแปร longVar (ภาพที่ 2.1 แสดงถึงขั้นตอนในการสร้างตัวแปรและการกำหนดค่าให้กับตัวแปรของ Java) การประกาศพร้อมกับกำหนดค่าให้กับตัวแปรในประโยคเดียวกัน ช่วยให้การเขียน code เมื่อมองดูแล้วสวยและกะทัดรัดขึ้น ทั้งนี้ก็คงจะขึ้นอยู่กับตัวผู้อ่านเองว่ามีความคิดเห็นอย่างไร สำหรับตัวผู้เขียนเองคิดว่าถ้าเป็นการประกาศที่มีตัวแปรไม่กี่ตัวในบรรทัดนั้นก็คงมองดูแล้วสวยดี แต่ถ้ามีมากเกินไปก็คงจะไม่เข้าท่าเท่าไรนัก ตัวอย่างเช่น

```
double price = 0.0D, tax = 0.7D, returned = 0.0D, interests = 0.0D,
principal = 10000.0D;
```

โปรแกรมเมอร์ควรจะแยกการประกาศที่เห็นด้านบนนี้ให้อยู่คนละบรรทัดเพื่อให้ง่ายต่อการอ่าน และเปลี่ยนแปลง หากมีการเปลี่ยนแปลงเกิดขึ้นในโปรแกรม



ภาพที่ 2.1 การกำหนดค่าให้กับตัวแปร

ในการกำหนดค่าให้กับตัวแปรนั้น เราไม่จำเป็นที่จะต้องกำหนดค่าของตัวแปรต่าง ๆ ให้เป็นศูนย์ เราสามารถที่จะกำหนดค่าอะไร ก็ได้ให้กับตัวแปรเหล่านี้ ทั้งนี้ขึ้นอยู่กับการใช้งานของตัวแปรนั้น ๆ หลาย ๆ ครั้งที่เราต้องตัดสินใจว่าควรจะใช้ตัวแปรชนิดใดดี ในกรณีที่ข้อมูลเป็นชนิดที่อยู่ในหมวดเดียวกัน เช่น ข้อมูลเป็นเลขที่ไม่มีจุดทศนิยม หรือ ข้อมูลเป็นเลขที่มีจุดทศนิยม ซึ่งถ้าเหตุการณ์เหล่านี้เกิดขึ้น เราก็จำเป็นที่จะต้องคิดถึงค่าความเป็นไปได้ของข้อมูลว่าอยู่ใน range เท่าใด เช่น ถ้าเราต้องการใช้ตัวแปรรองรับข้อมูลที่เป็นเลขที่ไม่มีจุดทศนิยมที่มีขนาดไม่เกิน 127 เราก็ควรที่จะประกาศตัวแปรด้วยการใช้ byte แทนการใช้ int เป็นต้น

เราสามารถที่จะประกาศตัวแปรพร้อมกับการกำหนดค่าให้กับตัวแปร หลาย ๆ ตัวในหนึ่งประโยค เช่น

```
int length = 2, height = 8, width = 4;
double angles = 30.5, distance = 24.50;
```

การประกาศตัวแปรในรูปแบบนี้ต้องใช้เครื่องหมาย , (comma) เป็นตัวแบ่งตัวแปรออกจากกัน และตัวแปรเหล่านั้นจะมีชนิดเป็นชนิดเดียวกันตลอด เราไม่สามารถประกาศตัวแปรต่างชนิดกันได้ด้วยวิธีการประกาศแบบนี้ เช่น

```
int cats = 90, float tax = 0.50; // ERROR - mixed-type declaration
                                 // and initialization
```

การประกาศตัวแปรไม่จำเป็นต้องกำหนดค่าให้ทันที แต่สามารถที่จะกำหนดได้ภายหลัง เช่น

```
double radius;
radius = 2.54;
```

ลองมาดูการประกาศและกำหนดค่าในรูปแบบอีกรูปแบบหนึ่ง

```
int one = 1, two = 2, three = one + two;
```

Java ยอมให้มีการประกาศและกำหนดค่าแบบนี้ได้ เนื่องจากว่า ตัวแปร one และ two มีการกำหนดค่าเรียบร้อยแล้ว ดังนั้น การประกาศและกำหนดค่าให้กับตัวแปร three จึงทำได้โดยไม่มีปัญหาใด ๆ จาก compiler เลย ประโยค three = one + two; เป็นประโยคที่มีการประมวลผลของ one + two ก่อนแล้วจึงนำค่าที่ได้มาเก็บไว้ที่ตัวแปร three เราจะพูดถึง ประโยค และ การประมวลผลที่เกิดขึ้นในประโยคในโอกาสต่อไป

การใช้ final ในการกำหนดค่าให้กับตัวแปร

หลาย ๆ ครั้งที่เราต้องการใช้ค่าบางค่าตลอดอายุการทำงานของโปรแกรม เช่น ค่าของ PI หรือ ค่าคงที่ที่จะไม่มีการเปลี่ยนแปลงใด ๆ ในขณะใช้งานในโปรแกรม Java มีคำสั่งที่ user ไม่สามารถที่นำมาใช้ในการประกาศตัวแปรได้ (reserved word) แต่เอาไว้ใช้ในงานนี้โดยเฉพาะ คือ คำว่า final

ถ้าเราขึ้นต้นการประกาศตัวแปรด้วยคำว่า final ตัวแปรนั้น ๆ จะไม่สามารถมีค่าอื่น ๆ ได้อีก ไม่ว่าจะเป็นการเปลี่ยนค่าภายในโปรแกรม หรือ การเปลี่ยนแปลงที่เกิดจากการใส่ข้อมูลใหม่ผ่านทาง keyboard ดังโปรแกรมตัวอย่างที่แสดงให้เห็นนี้

```
/*
 * Final.java - Use of final keyword
*/

class Final {
    public static void main(String[] args) {
        final double PI = 3.14;
        int radius = 2;

        double area = radius * radius * PI;
        System.out.println("Area of a circle is " + area);
    }
}
```

หลังจากที่ compile และ run โปรแกรม ผลลัพธ์ที่ได้คือ

```
Area of a circle is 12.56
```

ถ้าเราเปลี่ยนแปลงโปรแกรม Final.java ด้วยการกำหนดค่าให้กับตัวแปร PI ใหม่ ดังที่เห็นในโปรแกรม FinalWithError.java นี้ compiler จะฟ้องด้วยข้อมูลดังที่เห็น

```
/*
 * FinalWithError.java - Changing the value of final keyword
*/
```

```
class FinalWithError {
    public static void main(String[] args) {
        final double PI = 3.14;
        int radius = 2;

        double area = radius * radius * PI;
        System.out.println("Area of a circle is " + area);

        PI = 3.142;
        area = radius * radius * PI;
        System.out.println("Area of a circle is " + area);
    }
}
```

ผลของการ compile

```
FinalWithError.java:14: cannot assign a value to final variable PI
                PI = 3.142;
                ^
1 error
```

ขออธิบายถึงข้อความ error ที่ Java ฟ้องสักนิดหน่อย ถ้าผู้อ่านสังเกต error ที่เกิดขึ้นจะเห็นเลข 14 ถัดจากชื่อของโปรแกรมที่ได้รับการ compile ซึ่งบ่งบอกถึงเลขที่บรรทัดที่ error นี้เกิดขึ้น รวมไปถึงตำแหน่งที่ error เกิดขึ้นซึ่งเป็นตัวช่วยให้เรากลับเข้าไปแก้ไขโปรแกรมของเราได้ง่ายยิ่งขึ้น (ไม่ต้องเสียเวลาค้นหาบรรทัดที่เกิด error)

การใช้ final ทำให้โปรแกรมมีความแม่นยำและมั่นคงมากยิ่งขึ้น ความพยายามที่จะเปลี่ยนค่าเหล่านี้ก็ไม่สามารถทำได้ ทำให้การออกแบบและพัฒนาโปรแกรมเป็นไปได้ด้วยดี ลดข้อผิดพลาดที่อาจเกิดขึ้นจากตัวโปรแกรมเมอร์เอง

ในบางครั้งการประกาศตัวแปรต้องการใช้ค่าที่สูงมาก ๆ หรือ ค่าที่เล็กมาก ๆ เช่น ระยะทางจากโลกไปยังดวงอาทิตย์ ซึ่งมีค่าโดยประมาณเท่ากับ 149, 600,000 กิโลเมตร (1.496×10^8) หรือ การกำหนดค่าของมวลของอิเลคตรอน (electron mass) ซึ่งมีค่าเล็กมาก ขนาดของมวลโดยประมาณมีค่าเท่ากับ 0.00000000000000000000000000009 กรัม (9.0×10^{-28}) เราก็สามารถทำได้ด้วยการใช้การประกาศในรูปแบบของการกำหนดค่าทางวิทยาศาสตร์ (scientific notation) เช่น

```
final double sunDistance = 1.496E8; // 1.496 x 10^8
final float electronMass = 9.0E-28F;    // 9.0 x 10^-28
```

การประกาศและกำหนดค่าในรูปแบบนี้ทำให้การมองเห็น และ การเขียนหรือการอ่านทำได้ง่ายยิ่งขึ้น ลองนึกถึงการที่เราต้องเขียนประโยคในการกำหนดค่าให้กับ มวลของอิเลคตรอน ที่ได้กล่าวถึงข้างต้นซึ่งมีศูนย์ถึง 27 ตัวว่าน่ารำคาญขนาดไหน (โปรแกรมเมอร์ส่วนใหญ่รู้เรื่องนี้ดี) ยิ่งถ้าเราเขียน code เยอะ ๆ การกำหนดค่าให้กับตัวแปรเหล่านี้ด้วยการเขียนที่สั้นที่สุดย่อมเป็นการดีเสมอ (เขียนโปรแกรมได้ไวขึ้น?)

อายุ และ ขอบเขตการใช้งาน (life-time and scope) ของตัวแปร

หลังจากที่มีการประกาศใช้ตัวแปรใด ๆ ในโปรแกรม ตัวแปรเหล่านี้จะมีขอบเขต หรืออายุการใช้งานตามลักษณะการประกาศตัวแปรของผู้เขียนโปรแกรมเอง โดยทั่วไปตัวแปรจะมีอายุการใช้งานตามเนื้อที่ (block) ที่ตัวแปรเหล่านั้นปรากฏอยู่ ซึ่งจะอยู่ในเครื่องหมาย {} เช่นโปรแกรมตัวอย่างต่อไปนี้

```
/* Scope.java - Showing scope of variables*/

class Scope {
    public static void main(String[] args) {
        int x = 5, y = 8;
```

```
        System.out.print("Value of x is ");
        System.out.println(" Value of y is " + y);

        {
            int x = 45; //error x is already defined
            int w = 89;
            System.out.println("Value of x in brackets is " + x);
            System.out.println("Value of y in brackets is " + y);
        }
        //error w is out of scope
        System.out.println("Value of w is " + w);
    }
}
```

เป็นที่แน่นอนอยู่แล้วว่าโปรแกรม Scope.java จะ compile ไม่ผ่าน เนื่องจากเหตุผลสองอย่างดังที่ compiler ฟ้อง คือ (1) ตัวแปร int x ได้มีการประกาศใน main() แล้ว และ (2) ตัวแปร int w ไม่ได้อยู่ใน scope ของมันเอง ซึ่งในที่นี้คือ block ที่มีการประการ int x และ int w

```
Scope.java:13 x is already defined in main(java.lang.String[])
                        int x = 45;
                            ^
Scope.java:19: cannot resolve symbol
symbol  : variable w
location: class Scope
                System.out.println("Value of w is " + w);
                                                     ^
2 errors
```

ในภาษาอื่น เช่น C++ การประกาศตัวแปรใน block ใหม่ด้วยตัวแปรที่มีชื่อซ้ำกันสามารถทำได้โดยไม่มีเงื่อนไขใด ๆ (ดังที่เห็นในโปรแกรม Scope.java) แต่ตัว Java เองไม่ยอมให้มีการประกาศตัวแปรซ้ำ ถึงแม้ว่าจะอยู่ใน block ใหม่ที่ซ้อนกันอยู่ก็ตาม

ลองมาดูโปรแกรมตัวอย่างอีกตัวหนึ่ง

```
/* Scope2.java - Showing scope of variables*/

class Scope2 {
    static int x = 8, y;
    public static void main(String[] args) {
        {
            int x;
            x = 5;
            y = x;
            System.out.println("Value of y is " + y);
        }
        y = x;
        System.out.println("Value of y is " + y);
    }
}
```

โปรแกรม Scope2.java หลังจากที่ compile ผ่านแล้วจะได้ผลลัพธ์ดังนี้

```
Value of y is 5
Value of y is 8
```

จะเห็นว่า compiler ยอมให้ผ่านทั้งนี้เพราะตัวแปร x ใน block ด้านในมีการประกาศและกำหนดค่าอย่างสมบูรณ์ และ ตัวแปร x ใน scope ด้านนอกมีการกำหนดให้เป็น static ซึ่งเป็นตัวแปรที่ Java เรียกว่า class variable ดังนั้นตัวแปรทั้งสองจึงไม่ใช่ตัวแปรเดียวกัน เพราะฉะนั้นการกำหนดค่าให้กับตัวแปร y ใน

block ด้านในจะได้ค่าของตัวแปร x ที่อยู่ด้านใน และการกำหนดค่าให้กับ y อีกครั้งด้านนอกจึงได้ค่าของ class variable ที่ชื่อว่า x

เรามาลองดูโปรแกรมตัวอย่างอีกโปรแกรมหนึ่ง

```
/* Scope2v1.java - Showing scope of variables*/

class Scope2v1 {
    static int x = 8, y;
    public static void main(String[] args) {
        {
            //int x;
            x = 5;
            y = x;
            System.out.println("Value of y is " + y);
        }
        y = x;
        System.out.println("Value of y is " + y);
    }
}
```

โปรแกรม Scope2v1.java แสดงการใช้ตัวแปร x ตัวเดียวที่ได้ประกาศก่อน method main() ซึ่งมีค่าเป็น 8 ดังนั้นการเปลี่ยนค่าของ x ใน block ด้านในทำให้ค่าของตัวแปร x เป็นค่าล่าสุดคือ 5 ผลลัพธ์ของการ run โปรแกรมนี้จึงได้อย่างที่คาดหวังไว้ คือ

```
Value of y is 5
Value of y is 5
```

ลองมาดูอีกตัวอย่างหนึ่ง

```
/* Scope3.java - Showing scope of variables */

class Scope3 {
    static int x;
    public static void main(String[] args) {
        int x = (x = 12) + 8;
        System.out.println("Value of local x is " + x);
    }
}
```

ในโปรแกรม Scope3.java นี้เราประกาศตัวแปรสองตัวที่มีชื่อเหมือนกันคือ x โดยที่ตัวหนึ่งมี scope อยู่ใน class Scope3 และอีกตัวหนึ่งมี scope อยู่ใน method main() และเมื่อ compile ผ่านแล้วผลลัพธ์ที่ได้คือ

```
Value of local x is 20
```

คำถามที่เกิดขึ้นคือ ตัวแปร x ตัวใดที่ Java นำมาใช้ในการกำหนดค่าของประโยค int x = (x = 12) + 8;

เพื่อให้เห็นว่า Java ใช้ตัวแปร x ตัวไหนเราจึงดัดแปลง code ใหม่โดยกำหนดค่าให้กับตัวแปร x ที่อยู่ด้านนอกให้เป็น 4 ดังโปรแกรม Scope3v1.java ที่เห็นนี้

```
/* Scope3v1.java - Showing scope of variables */

class Scope3v1 {
    static int x = 4;
    public static void main(String[] args) {
        int x = x + 8;
```

```
        System.out.println("Value of local x is " + x);
    }
}
```

และเมื่อ compile เราจึงรู้ว่า Java ใช้ตัวแปร x ที่อยู่ใน block ด้านในเพราะว่า compiler ฟ้องด้วย error

```
Scope3v1.java:8 variable x might not have been initialized
                int x = x + 8;
                        ^
1 error
```

เพื่อให้การ compile ผ่านเราจึงแก้ไข code ใหม่ดังที่แสดงให้เห็นในโปรแกรม Scope4.java

```
/* Scope4.java - Showing scope of variables */

class Scope4 {
    public static void main(String[] args) {
        int x = (x = 12) + 8;
        System.out.println("Value of local x is " + x);
    }
}
```

โปรแกรม Scope4.java เป็นโปรแกรมเดียวกันกับโปรแกรม Scope3.java เพียงแต่เราลบการประกาศตัวแปรที่อยู่ด้านนอกออก และเมื่อ compile และ run ผลลัพธ์ที่ได้ก็คือผลลัพธ์เดิมที่ได้จากโปรแกรม Scope3.java แต่ถ้าอยากจะให้แน่ใจว่า Java ใช้ตัวแปรตัวไหนจริง ๆ ก็ลองเปลี่ยน code ให้เป็นดังที่เห็นในโปรแกรม Scope5.java (ถึงตอนนี้ผู้อ่านคงสามารถทายได้ว่าอะไรจะเกิดขึ้นถ้ามีการ compile)

```
/* Scope5.java - Showing scope of variables */

class Scope5 {
    public static void main(String[] args) {
        int x = x + 8;
        System.out.println("Value of local x is " + x);
    }
}
```

compiler ก็จะฟ้องด้วย error

```
Scope5.java:7: variable x might not have been initialized
                int x = x + 8;
                        ^
1 error
```

ซึ่งเป็น error ที่เหมือนกันกับ error ที่เกิดขึ้นจากการ compile โปรแกรม Scope3v1.java (เพราะว่าตัวแปร x ยังไม่มีค่ากำหนดใด ๆ เลย) ลองมาดูอีกตัวอย่างหนึ่ง

```
/* Scope6.java - Showing scope of variables */

class Scope6 {
    public static void main(String[] args) {
        int x = 12, y = x + 8;
        System.out.println("Value of local y is " + y);
    }
}
```

ซึ่งได้ผลลัพธ์หลังจากการ compile และ run ดังนี้

```
Value of local y is 20
```

การประกาศและกำหนดค่าแบบนี้สามารถทำได้เพราะ x ได้รับการกำหนดค่าก่อนที่จะถูกนำมาใช้ในการประกาศและกำหนดค่าให้กับตัวแปร y

การประกาศและกำหนดค่าให้กับตัวแปรใด ๆ นั้นจะต้องคำนึงถึงการใช้ และขอบเขตของการใช้เสมอ ทั้งนี้ในการกำหนดค่าใด ๆ ให้กับตัวแปรนั้นสิ่งที่นำมากำหนดค่านั้นในตัวของมันเองจะต้องมีค่าอยู่แล้ว หรือสามารถที่จะทำการประมวลค่าได้ก่อนที่จะนำมาใช้ในการกำหนดค่าให้กับตัวแปรอื่น ดังเช่นตัวอย่างที่เห็นก่อนหน้านี้

เราจะมาดูเรื่องของ scope ในสถานะอื่น ๆ เช่น scope ของ ตัวแปรที่เกี่ยวข้องกับ method, class และตัวแปรในสถานะอื่น ๆ ที่อยู่ในเนื้อหาที่เราจะได้พูดถึงในบทต่อ ๆ ไปเมื่อเราได้พูดถึงเรื่องนั้น ๆ แต่จะขอบอกในที่นี้ว่าตัวแปรถูกแบ่งออกเป็น 7 แบบคือ

1. class variable
2. instance variable
3. array component
4. method parameter
5. constructor parameter
6. exception-handler parameter
7. local variable

แต่เราได้พูดไปเพียงสองแบบคือ แบบที่ 1 (class variable) และแบบที่ 7 (local variable)

การสร้างประโยค (statement and expression)

ในภาษา Java ประโยค (statement) จะต้องจบด้วยเครื่องหมาย ; (semicolon) เสมอดังตัวอย่างที่แสดงให้เห็นนี้

```
int priceOfBook = 125;
float taxReturn;
```

เราไม่จำเป็นจะต้องเขียนประโยคให้จบภายในบรรทัดเดียว แต่ที่สำคัญต้องมีเครื่องหมาย ; ปิดท้ายเสมอ เช่น

```
int
    priceOfBook
        =
            125
    ;
```

Java จะมองหาเครื่องหมาย ; เสมอโดยไม่แยแสว่าประโยคนั้น ๆ จะกินเนื้อที่กี่บรรทัด แต่ที่สำคัญก็คือการเขียนที่ทำให้อ่านได้ง่าย ย่อมจะทำให้ทั้งเรา และผู้อื่นสามารถที่จะดู code ได้รวดเร็วยิ่งขึ้น ถ้าไม่เชื่อลองดู code จากโปรแกรมนี้ดูสิ

```
/* Scope2v1.java - Showing scope of variables* @author faa
xumsai*/class Scope2v1{static int x = 8, y;public static void
main(String[] args) { {int x;x = 5;y = x;System.out.println("Value of
y is " + y);}y = x; System.out.println("Value of y is " + y);}}
```

การเขียน code ให้มีรูปแบบที่เหมาะสม อ่านได้ง่าย จะทำให้การพัฒนาโปรแกรมเป็นไปได้ด้วยความรวดเร็วและเป็นที่ยอมรับตามระบบสากล รูปแบบที่เหมาะสมนั้นคือมีการย่อหน้า (indentation) เพื่อให้การอ่านทำได้ง่าย และมองดูแล้วสวยงาม ดังโปรแกรมตัวอย่างหลาย ๆ ตัวที่แสดงให้เห็นก่อนหน้านี้

Expression หมายถึงประโยคในภาษา Java ที่ได้รับการยอมรับว่าอยู่ในรูปแบบที่ได้กำหนดไว้ เช่น ประโยคในเรื่องของการกำหนดค่าให้กับตัวแปร ประโยคที่ต้องมีการประมวลผลในรูปแบบต่าง ๆ เช่น การ

ประมวลผลทางด้านคณิตศาสตร์ (mathematical evaluation) การประมวลผลทางด้านตรรกะ (relational evaluation) เป็นต้น

การประมวลผลทางด้านคณิตศาสตร์นั้น มี operator ที่ใช้อยู่ดังนี้ คือ + (บวก), – (ลบ), * (คูณ), / (หาร) และ % (การหาเศษที่เหลือจากการหารตัวเลขที่เป็น integer) โปรแกรมตัวอย่างต่อไปนี้แสดงถึงการประมวลผลของข้อมูลด้วย operator ชนิดต่าง ๆ ที่ได้กล่าวถึง

```
/* Operators.java - Shows mathematical operators */

class Operators {
   public static void main(String[] args) {
    int i, j, k;

    //generate random number between 1 and 10
    j = 1 + (int)(Math.random() * 10);
    k = 1 + (int)(Math.random() * 10);
    System.out.println("j = " + j + " and k = " + k);

    //evaluate and display results of math operators
    i = j + k; System.out.println("j + k = " + i);
    i = j - k; System.out.println("j - k = " + i);
    i = j * k; System.out.println("j * k = " + i);
    i = j / k; System.out.println("j / k = " + i);
    i = j % k; System.out.println("j % k = " + i);
   }
}
```

เนื่องจากว่าเราต้องการที่จะแสดงถึงการสร้างประโยคและการประมวลผลด้วยค่าที่มาจาก ข้างในโปรแกรมเอง และเราได้ใช้เครื่องมือในการสร้างค่าเหล่านี้ให้เราแบบอัตโนมัติ ซึ่งเครื่องมือที่ว่านี้ก็คือ method random() ที่มาจาก class Math (มีอยู่ใน Java แล้ว) method random() จะสร้างค่าระหว่าง 0 – 0.9+ พูดง่าย ๆก็คือ ค่าต่ำสุดคือ 0 และค่าสูงสุดคือ ค่าที่ใกล้ 1 แต่ไม่เท่ากับ 1 (ค่า n ใด ๆ ที่อยู่ในเงื่อนไขนี้ → 0.0 <= n < 1.0) เมื่อได้ค่านี้แล้วเราคูณค่านี้ด้วย 10 แล้วบวก 1 เราก็จะได้ค่าระหว่าง 1 – 10 หลังจากนั้นเราก็คำนวณหาค่า i ด้วยการใช้ operator ต่าง ๆ พร้อมทั้งแสดงค่าที่หาได้ไปยังหน้าจอ และเมื่อ run โปรแกรมผลลัพธ์ที่ได้คือ

```
j = 5 and k = 2
j + k = 7
j - k = 3
j * k = 10
j / k = 2
j % k = 1
```

และเนื่องจากว่าเราใช้ method random() เป็นตัวหาค่า ดังนั้นการ run ในแต่ละครั้งก็จะได้ผลลัพธ์ที่อาจซ้ำกัน หรือแตกต่างกัน

Operator ตัวอื่น ๆ ก็เหมือนกับการคำนวณ ทางด้านคณิตศาสตร์ที่เราคุ้นเคยโดยทั่วไป ยกเว้น % ซึ่งเป็นการหาเศษจากการหาร integer

ประโยค

```
i = j % k;
```

จะนำเอาค่าของตัวแปร k คือ 2 ไปหารค่าของตัวแปร j ซึ่งมีค่าเป็น 5 แล้วเก็บเศษที่ได้จากการหารไว้ (ซึ่งก็คือ 1) เมื่อได้แล้วก็นำค่านี้ไปเก็บไว้ในตัวแปร i ต่อไป

โปรแกรมตัวอย่างต่อไปนี้แสดงถึงการกำหนดค่าด้วยข้อมูลชนิด int และ double

```
/* Operators.java - Shows mathematical operators */
import java.util.Random;

class Operators {
    public static void main(String[] args) {
        //do the ints
        int i, j, k;

        //generate random number between 1 and 10
        Random rand = new Random();
        j = 1 + rand.nextInt(10);
        k = 1 + rand.nextInt(10);
        System.out.println("j = " + j + " and k = " + k);

        //evaluate and display results of math operators
        i = j + k; System.out.println("j + k = " + i);
        i = j - k; System.out.println("j - k = " + i);
        i = j * k; System.out.println("j * k = " + i);
        i = j / k; System.out.println("j / k = " + i);
        i = j % k; System.out.println("j % k = " + i);

        //do the doubles
        double v1, v2, v3;
        //generate random double between 1.0 and 10.0 (exclusive)
        v1 = 1.0 + rand.nextDouble() * 9;
        v2 = 1.0 + rand.nextDouble() * 9;
        System.out.println("v1 = " + v1 + " v2 = " + v2);
        v3 = v1 + v2; System.out.println("v1 + v2 = " + v3);
        v3 = v1 - v2; System.out.println("v1 - v2 = " + v3);
        v3 = v1 * v2; System.out.println("v1 * v2 = " + v3);
        v3 = v1 / v2; System.out.println("v1 / v2 = " + v3);
    }
}
```

โปรแกรม Operators.java แสดงตัวอย่างของการกำหนดค่า และ การคำนวณด้วย operator อย่าง ง่าย ๆ ทางด้านคณิตศาสตร์ โดยเริ่มต้นด้วยการกำหนดค่าให้กับตัวแปร i และ j ด้วยการสร้างค่าแบบสุ่ม (random) จาก method nextInt(max value) ซึ่ง method นี้จะกำหนดค่าที่อยู่ระหว่าง 0 ถึง ค่าที่กำหนดใน max value แต่ไม่รวมตัว max value ด้วยซึ่งถ้าเราต้องการที่จะสร้างค่าที่อยู่ระหว่าง 1 ถึง 10 เราก็ต้องบวก 1 เข้ากับค่าที่เกิดจากการเรียกใช้ method นี้ด้วย ดังที่ได้กำหนดไว้ในโปรแกรม

เราสามารถที่จะเขียนประโยคที่มีการประมวลผลในรูปของประโยคที่มีหลายตัวแปรในประโยคเดียวกันได้ ดังตัวอย่างนี้

```
/* Operators1.java - Shows mathematical operators */

class Operators1 {
    public static void main(String[] args) {
        int a, b, c, d, e;

        b = 1 + (int)(Math.random() * 10);
        c = 1 + (int)(Math.random() * 10);
        d = 1 + (int)(Math.random() * 10);
        e = 1 + (int)(Math.random() * 10);

        System.out.print("b = " + b + " c = " + c);
        System.out.println(" d = " + d + " e = " + e);
        a = b + c - d; System.out.println("Value of a is " + a);
        a = b * c - d; System.out.println("Value of a is " + a);
```

```
        a = b / c * d; System.out.println("Value of a is " + a);
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
b = 8 c = 2 d = 6 e = 5
Value of a is 4
Value of a is 10
Value of a is 24
```

การประมวลผล

เมื่อมีตัวแปรหลายตัวในการประมวลผลของประโยค ลำดับขั้น (precedence) ของการประมวลผลมีความสำคัญทันที ทั้งนี้ก็เนื่องมาจากรูปแบบของการประมวลผลตามชั้นของ operator ต่าง ๆ มีลำดับการประมวลผลที่ไม่เท่ากัน (ก่อน หรือ หลัง) ซึ่งเป็นไปตามลำดับขั้นของการประมวลผลตามกฎของการประมวลผลทางด้านคณิตศาสตร์ ดังตารางที่ 2.4

ตาราง 2.4 Operator Precedence

Operator ตามกลุ่ม	การประมวลผล
(), [], ., postfix ++, postfix --	ซ้ายไปขวา
unary +, unary –, prefix ++, prefix --	ขวาไปซ้าย
(datatype), new	ซ้ายไปขวา
*, /, %	ซ้ายไปขวา
+, -	ซ้ายไปขวา
<<, >>, >>>	ซ้ายไปขวา
<, <=, >, >=, instance of	ซ้ายไปขวา
==, !=	ซ้ายไปขวา
&	ซ้ายไปขวา
^	ซ้ายไปขวา
|	ซ้ายไปขวา
&&	ซ้ายไปขวา
||	ซ้ายไปขวา
?:	ซ้ายไปขวา
=, +=, -=, *=, /=, %=, <<=, >>=, >>>=, &=, |=, ^=	ขวาไปซ้าย

การประมวลผลข้อมูลที่เป็น integer

การประมวลผลข้อมูลที่เป็น integer ทั้งหมดไม่มีความสลับซับซ้อนอะไรเลย เป็นการประมวลผลเหมือนกับที่เราเคยเรียนมา ในวิชาคณิตศาสตร์ซึ่งลำดับขั้นของการประมวลผลก็ขึ้นอยู่กับชนิดของ operator ที่ใช้ในการประมวลผลนั้น ๆ เช่น การคูณและการหาร ทำก่อน การบวกและการลบ statement

50 * 2 – 45 / 5

จะได้ผลลัพธ์คือ 91 ซึ่งได้มาจากการนำเอา 2 ไปคูณ 50 ก่อนหลังจากนั้นก็เอา 5 ไปหาร 45 เมื่อได้แล้วก็นำเอาผลลัพธ์ที่ได้ไปลบออกจากผลลัพธ์ของการคูณ 50 กับ 2 ซึ่งมีค่าเท่ากับ 100 – 9

แต่ถ้าเราต้องการที่จะให้การประมวลผลอยู่ในรูปแบบอื่น ที่เราต้องการเราก็ใช้วงเล็บเป็นตัวกำหนด เช่น ถ้าเราต้องการที่จะให้การลบเกิดก่อน ในประโยคที่เห็นก่อนหน้านี้เราก็เขียนใหม่เป็น

50 * (2 – 45) / 5

ซึ่งมีค่าเท่าเทียมกับประโยค

50 * -43 / 5

และมีค่าเท่ากับ -430

เนื่องจากว่าการประมวลผลข้อมูลที่เป็น integer โดยเฉพาะการหารนั้น เราจะไม่มีเศษเหลือให้ทำการหารต่อไปเหมือนที่เราเคยหารกันมา เช่น ถ้าเราเอา 4 ไปหาร 9 เราก็จะได้ 2.25 แต่ถ้าเป็นการหารด้วย integer ใน Java แล้วผลลัพธ์ที่เราได้จากการหารนี้จะเป็น 2 เท่านั้น เราจะไม่ได้เศษ เราจะได้แต่ส่วน ลองดูตัวอย่างต่อไปนี้

5 / 2 = 2
125 / 4 = 31
3 / 9 = 0
15 / 15 = 1

แต่เราก็หาเศษได้ด้วยการใช้ operator % ดังที่ได้แสดงในโปรแกรมตัวอย่างก่อนหน้านี้ หรือดังตัวอย่างการหาเศษต่อไปนี้

5 % 2 = 1
125 % 4 = 1
3 % 9 = 3
15 % 15 = 0

โดยทั่วไปเท่าที่พบเห็นและประสพมา ผู้อ่านที่ใหม่ต่อการเขียนโปรแกรม มักจะสับสนในเรื่องของการหาเศษ หรือการหาส่วนของข้อมูลที่เป็น integer อย่างไรก็ตามเรื่องเหล่านี้มักจะหายไปถ้าได้ทดลองทำอย่างสม่ำเสมอ สำหรับ Java นั้นการหาเศษสามารถที่จะทำได้กับข้อมูลที่เป็น double หรือ float ด้วย แต่ในที่นี้จะทิ้งไว้ให้ผู้อ่านได้ทดลองทำการตรวจสอบดูเอง (เนื่องจากว่าผู้เขียนเองยังมองไม่เห็นประโยชน์ของการหาเศษจากข้อมูลที่เป็น double หรือ float เลย)

ในการเขียนโปรแกรมนั้นเราเก็บข้อมูลไว้ในตัวแปร ดังนั้นการประมวลผลก็ต้องทำกับตัวแปรเหล่านั้น เช่น เราสามารถที่จะคำนวณหาค่าของภาษีมูลค่าเพิ่ม (vat) จากตัวแปร price และ rate ถ้าตัวแปรทั้งสองได้รับการกำหนดค่าเรียบร้อยแล้ว เช่น

```
double price = 120.0, rate = 7.5;
double vat = price * rate;
```

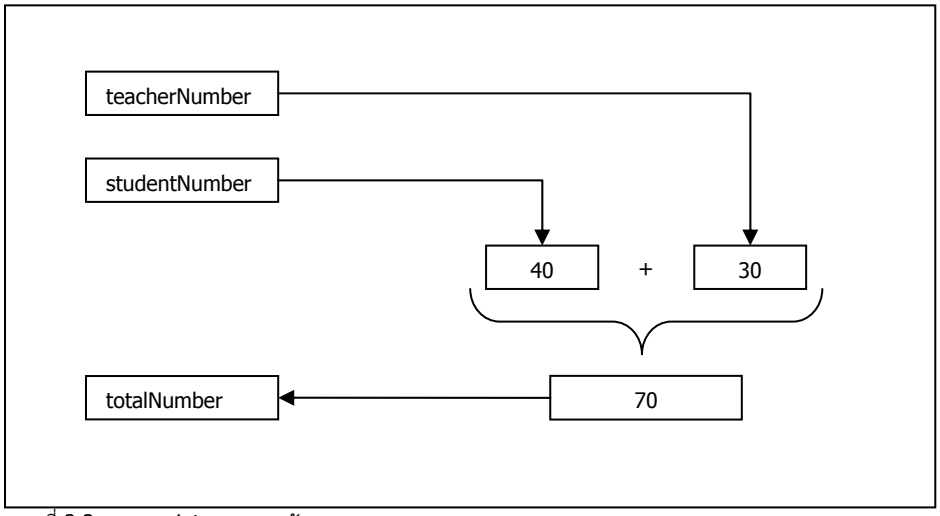
ลองมาดูโปรแกรมตัวอย่างการประมวลผลด้วยข้อมูลที่เป็น integer

```
/*
 * Integer.java - Integers calculation */

class Integer {
    public static void main(String[] args) {
        //declares and initialize three variables
        int studentNumber = 40;
        int teacherNumber = 30;
        int totalNumber = 0;

        totalNumber = studentNumber + teacherNumber;

        //display result
        System.out.print(studentNumber + " students + " +
                         teacherNumber);
        System.out.println(" teachers = " + totalNumber + " people");
    }
}
```

โปรแกรม Integer.java เป็นการแสดงถึงการคำนวณหาค่าของผลรวมของจำนวนนักศึกษา (studentNumber)และครู (teacherNumber) จากประโยค

```
totalNumber = studentNumber + teacherNumber;
```

ซึ่งขั้นตอนของการประมวลผล อาจพูดได้คร่าว ๆ คือ Java นำเอาค่าที่เก็บไว้ในตัวแปร studentNumber และค่าที่เก็บไว้ในตัวแปร teacherNumber มารวมกันและนำเอาผลลัพธ์ที่ได้ไปเก็บไว้ในตัวแปร totalNumber ผ่านทางเครื่องหมาย = (assignment operator) ลองดูภาพที่ 2.2



ภาพที่ 2.2 การบวก integer สองตัว

ในการแสดงผลลัพธ์ที่ได้จากการประมวลผลไปยังหน้าจอนั้น เราส่งผ่านทาง System.out.print() และ System.out.println() ประโยค

```
System.out.print(studentNumber + " students + " + teacherNumber);
```

จะให้ผลลัพธ์คือ ค่าที่เก็บไว้ในตัวแปร studentNumber ตามด้วย string ที่มีค่าเป็น "student +" ตามด้วยค่าที่เก็บไว้ในตัวแปร teacherNumber ในบรรทัดเดียวกัน หลังจากนั้นประโยค

```
System.out.println(" teachers = " + totalNumber + " people");
```

ก็จะให้ผลลัพธ์ คือ string ที่มีค่าเป็น " teachers = " ตามด้วยค่าที่เก็บไว้ในตัวแปร totalNumber ตามด้วย string ที่มีค่าเป็น " people" ซึ่งเมื่อรวมแล้วทั้งหมดก็จะได้ผลลัพธ์ที่แสดงไปยังหน้าจอดังนี้

```
40 students + 30 teachers = 70 people
```

คำว่า "ตามด้วย" จากคำอธิบายด้านบนนี้เกิดจากการประมวลผลของ Java กับเครื่องหมาย + ในการใช้ประโยคของการแสดงผล (System.out) ข้อมูลใด ๆ ที่อยู่ในเครื่องหมาย " " จะถูกส่งไปยังหน้าจอ แต่ถ้าเป็นตัวแปรหรือประโยคที่ต้องมีการประมวลผลอีก Java จะทำการประมวลผลก่อนเพื่อให้ได้ค่าสุดท้ายที่สามารถจะส่งออกไปหน้าจอได้ เช่น

```
System.out.println("30 + 45 = " + (30 + 45));
```

สิ่งที่ถูกส่งออกไปก่อนคือ "30 + 45 = " ตามด้วยค่าของการประมวลผลของประโยค (30 + 45) ซึ่งก็คือ 75 ดังนั้นผลลัพธ์ที่ส่งไปยังหน้าจอคือ

```
30 + 45 = 75
```

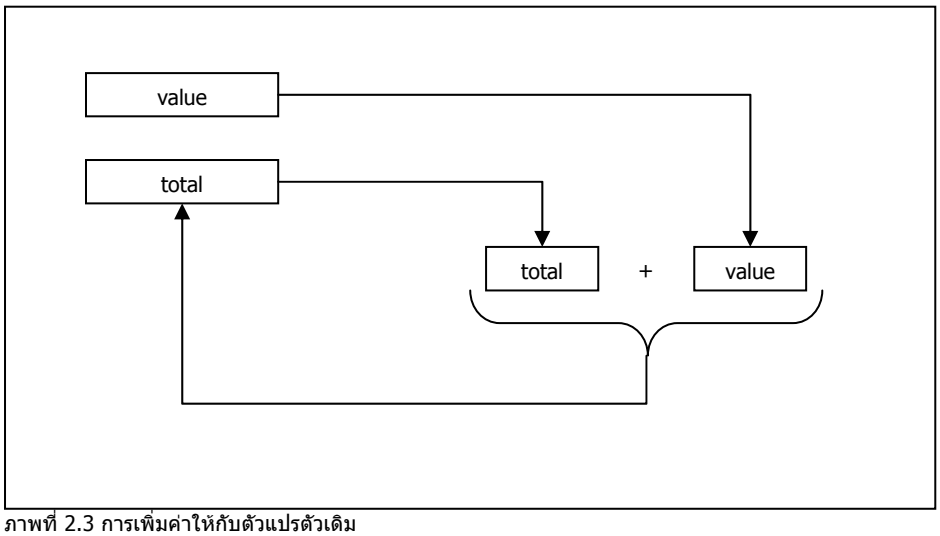
วงเล็บในประโยคของการบวก 30 กับ 45 ก็มีส่วนสำคัญเพราะถ้าไม่มี Java ก็จะแสดงผลเป็น

```
30 + 45 = 3045
```

การใส่วงเล็บเป็นการบอก Java ให้ทำการคำนวณหาค่าของประโยคนั้น ๆ ทั้งนี้เฉพาะในประโยคที่ใช้ System.out เท่านั้น ส่วนการประมวลผลที่อื่น ๆ ก็ทำตามเดิม เหตุที่เป็นเช่นนี้ก็เพราะว่า System.out จะนำค่านั้น ๆ ที่เป็นอยู่ไปแสดงถ้าไม่มีการคำนวณใด ๆ

สมมติว่าเราต้องทำการบวกค่าใดค่าหนึ่งให้กับตัวแปร เราก็สามารถที่จะสร้างประโยคได้เหมือนกับที่เราเคยทำมาก่อนหน้านี้ เช่น total = total + value;

ถ้าเราลองเขียนภาพแทนประโยคดังกล่าว เราก็อาจเขียนได้ดังนี้



ภาพที่ 2.3 การเพิ่มค่าให้กับตัวแปรตัวเดิม

ประโยคที่เห็นด้านบนนี้สามารถที่จะเขียนอีกแบบได้ ดังนี้คือ

```
total += value;
```

เราใช้เครื่องหมาย += ในการสร้างประโยคของเรา แต่ได้ผลลัพธ์เช่นเดียวกันกับประโยคก่อนหน้านี้ และลดเวลาในการเขียนลง (ถึงแม้ว่าจะไม่มากก็ตาม) เช่นเดียวกับการประมวลผลด้วย operator ตัวอื่น ๆ เราก็สามารถจะใช้วิธีเดียวกันนี้ได้ ดังที่ได้สรุปไว้ใน ตาราง 2.5

ตาราง 2.5 การใช้ operator ต่าง ๆ ร่วมกับการกำหนดค่า

ประโยคตัวอย่าง	ประโยคที่ได้ผลลัพธ์เหมือนกัน
sum = sum + count;	sum += count;
sum = sum - count;	sum -= count;
sum = sum * count;	sum *= count;
sum = sum / count;	sum /= count;
sum = sum % count;	sum %= count;

การกำหนดค่าแบบนี้ไม่จำเป็นที่จะต้องทำกับตัวแปรเท่านั้น เราสามารถที่จะทำกับค่าคงที่อื่น ๆ ได้ เช่น

```
total += 4;
sum += 1;
```

ยังมี operator ที่เราสามารถใช้ได้โดยไม่ต้องเสียเวลาในการเขียนมากมายนัก เช่น การเพิ่ม หรือ ลดทีละหนึ่งค่า ดังประโยคตัวอย่าง

```
count = count + 1;
value -= 1;
```

ทั้งสองประโยคที่เห็นด้านบนทำการเพิ่มค่าและลดค่าทีละหนึ่งค่า ตามลำดับ เราสามารถที่จะเขียนใหม่ด้วยการใช้ operator ++ และ operator -- ได้ดังนี้ คือ

```
count++;
value--;
```

จะเห็นว่าถ้าเราต้องการเพิ่มค่าเพียงแค่หนึ่งค่าเราก็ใช้เครื่องหมาย ++ ตามหลังตัวแปรนั้น เช่นเดียวกันถ้าเราต้องการลดค่าลงหนึ่งค่า เราก็ใช้เครื่องหมาย –- ตามหลังตัวแปรนั้น เราเรียกการเขียนแบบนี้ว่า การเขียนแบบ postfix (postfix notation) ซึ่งเป็นการใช้ operator ตามหลังตัวแปร และ Java ยอมให้มีการเพิ่มหรือลดค่าได้เพียงแค่หนึ่งค่าเท่านั้น ถ้าเราต้องการเพิ่มหรือลดค่ามากกว่าหนึ่งค่า เราต้องใช้วิธีสองวิธีที่ได้แสดงการใช้ก่อนหน้านี้ เช่น ถ้าต้องการเพิ่มทีละสอง เราก็เขียนเป็น

```
total += 2;
```

และถ้าเราต้องการลดทีละสาม เราก็เขียนเป็น

```
total -= 3;
```

Java ยังมี Operator อีกตัวหนึ่งที่เราสามารถเลือกใช้ ในการประมวลผลถ้าเราคำนึงถึงขั้นตอนของการประมวลผล ว่าควรจะทำอะไรก่อน อะไรหลัง ดังประโยคตัวอย่างต่อไปนี้

```
int count = 8, value = 5;
count = count + value++;
```

ถ้ามองดูเผิน ๆ เราก็อาจบอกว่าประโยคนี้เป็นการบวกค่าของตัวแปร value เข้ากับตัวแปร count พร้อมกับการเพิ่มค่าให้กับตัวแปร value อีกหนึ่งค่า แต่เรารู้หรือไม่ว่าอะไรได้รับการประมวลผลก่อน count + value ก่อน หรือ value++ ก่อน และผลลัพธ์ที่ได้จากการบวกคืออะไร

จากประโยคด้านบน Java จะนำเอาค่าของ value มาบวกเข้ากับค่าของ count ก่อน หลังจากนั้นจึงเพิ่มค่าให้กับตัวแปร value อีหนึ่งค่า ดังนั้นถ้าเรา execute ประโยค count = count + value++ เราก็จะได้ผลลัพธ์เป็น 13 และค่าของ value หลังจากการบวกจะเป็น 6 แต่ถ้าเราเปลี่ยนประโยคให้เป็นดังนี้

```
count = count + ++value;
```

เราจะได้ผลลัพธ์จากการบวกเป็น 14 (ทำไม)

เนื่องจากว่าเราใช้เครื่องหมาย ++ นำหน้าตัวแปร ดังนั้น Java จึงทำการเพิ่มค่าให้กับตัวแปร value ก่อน หลังจากนั้นจึงนำค่าที่ได้นี้ไปบวกเข้ากับค่าของตัวแปร count ผลลัพธ์ที่ได้จึงเป็น 14 ดังที่เห็น เราเรียกการเขียนแบบนี้ว่าการเขียนแบบ prefix (prefix notation)

ขึ้นอยู่กับตำแหน่งที่ใช้ prefix หรือ postfix ในประโยค บางครั้งการประมวลผลจาก Java อาจได้ผลลัพธ์ที่เหมือนกัน เราจะกลับมาพูดถึงกรณีดังกล่าวในเรื่องของการทำงานแบบวน หรือ loop (repetition)

ลองมาดูตัวอย่างการใช้ prefix และ postfix กันอีกสักหน่อย

```
int count = 10, value = 5;
count--;
value++;
value = ++count + ++value;
```

ถ้าเรา execute ประโยค value = ++count + --value; เราก็จะได้ผลลัพธ์ คือ value มีค่าเป็น 17 เพราะก่อนหน้านั้น count มีค่าเป็น 9 และ value มีค่าเป็น 6 (จาก count—และ value++) แต่ได้รับการเพิ่มให้อีกหนึ่งค่าก่อนการบวกในประโยคสุดท้าย

ถ้าหากว่าเราต้องการให้ code ที่เขียนนั้นอ่านได้ง่ายขึ้นเราก็อาจใช้วงเล็บเป็นตัวช่วยก็ได้ เช่น

```
value = (++count) + (++value);
```

หรืออาจเขียนให้ดูแล้วเวียนหัวแบบนี้

```
value = (count += 1) + (value += 1);
```

หรือแบบนี้

```
value = (count = count + 1) + (value = value + 1);
```

ส่วนประโยคต่อไปนี้จะให้ผลลัพธ์ต่างจากตัวอย่างด้านบน (ทำไม?)

```
value =  (count++) + (value++);
```

ถ้าเราลองวิเคราะห์ประโยคด้านบนนี้ ด้วยค่าของ count = 10 และค่าของ value = 5 และตามข้อกำหนดการใช้งานของ postfix ++ เราก็ต้องสรุปว่า Java จะนำเอาค่าของ count ซึ่งมีค่าเท่ากับ 10 บวกเข้ากับค่าของ value ซึ่งมีค่าเท่ากับ 5 เมื่อได้แล้วก็นำผลลัพธ์ที่ได้ไปเก็บไว้ในตัวแปร value ดังนั้น value จึงมีค่าเท่ากับ 15 หลังจากนั้นก็จะเพิ่มค่าให้กับ count และ value อีกหนึ่งค่า ซึ่งทำให้ count มีค่าเท่ากับ 11 และ value มีค่าเท่ากับ 16

แต่เมื่อทดลอง run ดูแล้วผลลัพธ์ที่ได้กลับเป็น value = 15 และ count = 11 ทำไม? เพื่อให้เกิดความกระจ่างเราก็ทดลองเขียนประโยคตรวจสอบใหม่ คือ

```
value = 5;
value = value++;
```

และเมื่อ run ดูเราก็ได้ผลลัพธ์เป็น value = 5 ทำให้เราค่อนข้างจะมั่นใจว่า Java ไม่ยอมให้ postfix operator ++ เพิ่มค่าให้กับตัวแปรเดิมที่ได้รับการกำหนดค่าในประโยคเดียวกัน แต่เพื่อให้แน่ใจขึ้นอีกเราก็ทดลอง run ด้วยประโยค

```
int sum = value++;
```

เราได้ผลลัพธ์เป็น sum มีค่าเท่ากับ 5 และ value มีค่าเท่ากับ 6 ดังนั้นเราจึงมั่นใจว่า postfix operator ++ ไม่ยอมให้มีการเพิ่มค่าให้กับตัวแปรตัวเดียวกันกับที่ใช้ในการกำหนดค่าในประโยคเดียวกัน (ดังที่ได้กล่าวไว้)

การเขียนโปรแกรมที่ดีนั้นไม่ควรเขียนให้อ่านยาก เช่นตัวอย่างของการใช้ ++ ที่ได้กล่าวถึงก่อนหน้านี้ ถ้าการเขียนนั้นได้ผลลัพธ์ดังที่เราต้องการ ถึงแม้ว่าจะเขียนด้วยประโยคหลาย ๆ ประโยค ก็ยังดีกว่าที่เขียนด้วยจำนวนประโยคที่สั้น แต่อ่านได้ยากกว่า ตัวอย่างต่อไปนี้เป็นการใช้ operator ++ และ operator – ในรูปแบบต่าง ๆ

```
/* Increments.java - Shows increment operators */
class Increments {
    public static void main(String[] args)  {
        int value = 25, number = 25;

        System.out.println("value is " + value);
        System.out.println("number is " + number);
        --value;
        number++;
        System.out.println("value is " + value);
        System.out.println("number is " + number);
        value = value + --number;
        System.out.println("value is " + value);
```

```
        System.out.println("number is " + number);
        number = number - value++;
        System.out.println("value is " + value);
        System.out.println("number is " + number);
        number--;
        value++;
        System.out.println("value is " + value);
        System.out.println("number is " + number);
        value = --value;
        number = number++;
        System.out.println("value is " + value);
        System.out.println("number is " + number);
    }
}
```

ผลลัพธ์ที่ได้จากการ run

```
value is 25
number is 25
value is 24
number is 26
value is 49
number is 25
value is 50
number is -24
value is 51
number is -25
value is 50
number is -25
```

ผู้อ่านควรทดสอบการใช้ operator increment ทั้งสอง (++ และ --) ในประโยคสมมติต่าง ๆ เพื่อให้เกิดความเข้าใจในการใช้ operator ทั้งสองได้ดียิ่งขึ้น

การประมวลผลด้วย integer ขนาดเล็ก (short และ byte)

ในการประมวลผลด้วยข้อมูลที่เป็น byte หรือ short นั้น Java จะทำการประมวลผลเช่นเดียวกันกับการประมวลผลด้วยข้อมูลที่เป็น int และผลลัพธ์ของการประมวลผลข้อมูลที่เป็น short หรือ byte นั้นจะถูกเก็บอยู่ในรูปแบบของข้อมูลที่เป็น int ดังตัวอย่างต่อไปนี้

```
/* ByteShort.java - Shows byte and short integer calculation */
class ByteShort {
    public static void main(String[] args) {
        short numScooters = 10;
        short numChoppers = 5;
        short total = 0;

        total = numScooters + numChoppers;

        System.out.println("Number of scooters is " + numScooters);
        System.out.println("Number of choppers is " + numChoppers);
        System.out.println("Total number of bikes is " + total);
    }
}
```

เมื่อได้ compile แล้ว Java จะฟ้องว่ามี error เกิดขึ้น

```
ByteShort.java:10: possible loss of precision
```

```
found   : int
required: short
       total = numScooters + numChoppers;
                                     ^
1 error
```

เนื่องจากว่าผลลัพธ์ของการบวกจะต้องเก็บเป็น integer ที่มีขนาดเท่ากับ 64 bit แต่ตัวแปร total สามารถเก็บข้อมูลได้สูงสุดเท่ากับ 16 bit ดังนั้น compiler จึงฟ้องทันที วิธีการแก้ไขก็อาจจะต้องเปลี่ยนตัวแปรทั้งสามให้เป็น int หรือไม่ก็ใช้วิธีการเปลี่ยนชนิดของข้อมูลที่เรียกว่า casting

การเปลี่ยนแปลงชนิดของข้อมูลด้วยการ cast

ในการ cast นั้นเราจำเป็นที่จะต้องคำนึงถึงการจัดเก็บข้อมูลว่าเราจะต้องใช้ข้อมูลชนิดไหน ในการจัดเก็บการประกาศตัวแปรก็ต้องทำด้วยความระมัดระวัง เนื่องจากว่าการ cast นั้นถ้าเรา cast ข้อมูลที่มีขนาดใหญ่ไปสู่ข้อมูลที่มีขนาดเล็ก เราจะสูญเสียค่าความเป็นจริงของข้อมูลนั้น เรามาทำการแก้ไขโปรแกรม ByteShort.java เพื่อให้ Java ยอมให้การ compile เป็นไปได้ด้วยดีกันดีกว่า

วิธีการก็ไม่ยากเพียงแต่เปลี่ยนประโยค total = numScooters + numChoppers; ให้เป็น

```
total = (short)(numScooters + numChoppers);
```

เท่านี้ Java ก็ยอมให้เรา compile และเมื่อเราทดลอง run ดูก็ได้ผลลัพธ์ดังที่คาดไว้ คือ

```
Number of scooters is 10
Number of choppers is 5
Total number of bikes is 15
```

ทีนี้เราลองมาดูกันว่าถ้าเราเปลี่ยนค่าของ numScooters ให้เป็น 32767 ดังที่เห็นด้านล่าง ผลลัพธ์ที่ได้ในการ compile และ run โปรแกรมของเราจะเป็นอย่างไร

```
short numScooters = 32767;    //ค่าสูงสุดที่เป็นไปได้
short numChoppers = 5;
```

ผลลัพธ์ที่ได้ คือ

```
Number of scooters is 32767
Number of choppers is 5
Total number of bikes is -32764
```

จะเห็นว่าค่าของ total เป็น -32764 ซึ่งเป็นค่าที่ไม่ถูกต้อง ซึ่งถ้าบวกกันแล้วค่าที่ได้ต้องเป็น 32772 ทำไม?

เนื่องจากว่าตัวแปรทั้งสามตัวมีชนิดเป็น short และสามารถเก็บข้อมูลได้สูงสุดเพียงแค่ 32767 ดังนั้นการนำเอาผลรวมที่มีค่าเกินกว่าค่าสูงสุดที่เป็นไปได้มาเก็บไว้ในตัวแปร total จึงไม่สามารถทำได้แต่ Java ก็จะนำเอาค่าอื่นที่ได้จากการประมวลผล (ที่ถูกต้องตามหลักการแต่มีค่าความคลาดเคลื่อนของการประมวลผล – จำนวน bit ที่ใช้ในการคำนวณหายไปครึ่งหนึ่ง จาก 64 bit เหลือเพียง 32 bit) มาใส่ไว้ให้ ผู้เขียนโปรแกรมจะต้องทำความเข้าใจเกี่ยวกับชนิดของข้องมูล และการประมวลผลตามชนิดนั้น ๆ ของข้อมูล ถ้าหากว่าหลีกเลี่ยงได้ ผู้เขียนโปรแกรมก็ควรหลีกเลี่ยงการใช้ cast นอกเสียจากว่าจำเป็นต้องใช้จริง ๆ เท่านั้น

เรามาดูการ cast อื่น ๆ จากโปรแกรม Casting.java

```
/* Casting.java - Changing type of data */
class Casting {
    public static void main(String[] args) {
        byte byteVar = 127;
```

```
        short shortVar = 32767;
        long longVar = 100000;
        int intVar = 300000;

        System.out.println("byteVar is " + byteVar);
        System.out.println("shortVar is " + shortVar);
        System.out.println("longVar is " + longVar);
        System.out.println("intVar is " + intVar);

        byteVar = (byte)shortVar;
        shortVar = (short)longVar;
        longVar = (int)intVar * (int)longVar;
        intVar = (short)intVar * (short)intVar;

        System.out.println("byteVar is " + byteVar);
        System.out.println("shortVar is " + shortVar);
        System.out.println("longVar is " + longVar);
        System.out.println("intVar is " + intVar);
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
byteVar is 127
shortVar is 32767
longVar is 100000
intVar is 300000

byteVar is -1
shortVar is -31072
longVar is -64771072
intVar is 766182400
```

ในการประมวลผลด้วยการใช้ข้อมูลที่มีชนิดเป็น long นั้น (บังคับให้เป็น long ด้วยการใส่ L ด้านหลังค่าที่กำหนดให้กับตัวแปรนั้น ๆ) ข้อมูลชนิดอื่นจะถูกเปลี่ยนให้เป็น long ก่อนการประมวลผล เช่น

```
long value = 125;
long price = 200L;
int total = 0;

total = value * price;
```

ก่อนการประมวลผลค่าของตัวแปร value จะถูกเปลี่ยนให้เป็น long แล้วจึงนำมาคูณกับตัวแปร price เพราะฉะนั้นในการประมวลผลข้อมูลที่เป็น integer นั้นข้อควรคำนึงถึงก็คือ

- **การประมวลผลกับตัวแปรที่ประกาศเป็น long ที่มีการกำหนดค่าด้วยตัว L จะเป็นการประมวลผลด้วยการใช้จำนวน bit เท่ากับ 64 bit**
- **การประมวลผลกับตัวแปรที่ประกาศเป็น int จะเป็นการประมวลผลด้วยการใช้จำนวน bit เท่ากับ 32 bit**

ปัญหาของการประมวลผลด้วยข้อมูลที่มีชนิดเป็น integer

- **ถ้าเราหารตัวแปรที่มีชนิดเป็น int ด้วย 0 เราไม่สามารถหาคำตอบได้ เพราะฉะนั้น Java จะทำการฟ้องด้วย error แต่เราก็สามารถที่จะตรวจสอบและแก้ไขได้ด้วยการใช้ exception ซึ่งเป็นวิธีดักจับ error ที่ Java เอื้ออำนวยให้ผู้เขียนโปรแกรมได้ทำการเอง เราจะพูดถึง exception ในโอกาสต่อไป**
- **การประมวลผลด้วยค่าที่เกินขีดจำกัดของตัวแปรจะทำให้ได้ผลลัพธ์ที่คลาดเคลื่อน**

- การประมวลผลด้วย % ที่มีค่าของ 0 อยู่ทางขวาก็ทำให้เกิด error เหมือนกับการหารด้วย 0

การประมวลผลด้วยตัวแปรที่มีจุดทศนิยม (Floating Point Calculations)

การประมวลผลของข้อมูลที่เป็นเลขทศนิยมนั้นก็คล้ายกันกับการประมวลผลด้วยข้อมูลที่เป็น integer การใช้ operator ต่าง ๆ ก็เหมือนกัน คือ + - * / ลองมาดูตัวอย่างกัน

```
/* FloatingPoint.java - Calculating income */
class FloatingPoint {
    public static void main(String[] args) {
        double weeklyPay = 1075 * 5; //1075 per day
        double extras = 1580;
        final double TAX_RATE = 8.5; //tax rate in percent

        //calculate tax and income
        double tax = (weeklyPay + extras) * TAX_RATE / 100;
        double totalIncome = (weeklyPay + extras) - tax;

        System.out.println("Tax = " + tax);
        System.out.println("Total income = " + totalIncome);
    }
}
```

โปรแกรม FloatingPoint.java เป็นการคำนวณหาภาษี และรายได้ต่อหนึ่งอาทิตย์ โดยเราสมมติให้รายได้ต่อวันเท่ากับ 1075 บาท และรายได้พิเศษเท่ากับ 1580 บาท อัตราภาษีเท่ากับ 8.5% เมื่อเรา execute โปรแกรมเราก็จะได้ผลลัพธ์ดังนี้

```
Tax = 591.175
Total income = 6363.825
```

ในการคำนวณหาภาษีนั้นเราจำเป็นที่จะต้องใช้วงเล็บในประโยค

```
double tax = (weeklyPay + extras) * TAX_RATE / 100;
```

เนื่องจากว่าเราต้องการให้การบวกทำก่อนการคูณและการหาร ถ้าไม่เช่นนั้นแล้วเราจะไม่ได้ค่ารายได้โดยรวมที่เป็นจริง ถ้าเราสังเกตให้ดีผลลัพธ์ที่ได้จะมีเลขหลังจุดทศนิยมอยู่สามตัว เราทำให้เป็นสองตัวดังที่ใช้กันอยู่ทั่วไปได้มั้ย? คำตอบก็คือ ได้ ด้วยการเพิ่ม code เข้าไปในโปรแกรมดังนี้

```
/* FloatingPoint2.java - Calculating income with formatted output */

import java.text.*;   //for NumberFormat
class FloatingPoint2 {
    public static void main(String[] args) {
        double weeklyPay = 1075 * 5; //1075 per day
        double extras = 1580;
        final double TAX_RATE = 8.5; //tax rate in percent

        //set format output
        NumberFormat form = NumberFormat.getNumberInstance();
        form.setMaximumFractionDigits(2);
        //calculate tax and income
        double tax = (weeklyPay + extras) * TAX_RATE / 100;
        double totalIncome = (weeklyPay + extras) - tax;

        System.out.println("Tax = " + form.format(tax));
        System.out.println("Total income = " +
                           form.format(totalIncome));
```

```
    }
}
```

เราเพิ่มประโยค

```
NumberFormat form = NumberFormat.getNumberInstance();
form.setMaximumFractionDigits(2);
```

และเปลี่ยนการแสดงผลลัพธ์ด้วยการใช้ form.format(tax) และ form.format(totalIncome) เพื่อให้ผลลัพธ์ที่ออกมาเป็น

```
Tax = 591.18
Total income = 6,363.82
```

จะเห็นว่า 591.175 จะถูกปัดให้เป็น 591.18 และ 6363.825 ถูกปัดให้เป็น 6363.82 โดยที่ค่าของ totalIncome มี , (comma) เป็นตัวบอกถึงจำนวนของหลักร้อยด้วย

ในการใช้ method ทั้ง getNumbereInstance() และ setMaximumFractionDigits() นั้นเราจำเป็นที่จะต้อง import class NumberFormat เข้าสู่โปรแกรมของเรา ถ้าไม่แล้วเราจะไม่สามารถใช้ method เหล่านี้ได้

ยังมี method อื่นใน class DecimalFormat ที่เราสามารถเรียกใช้ในการ format ผลลัพธ์ที่เป็นจุดทศนิยมได้ เช่นถ้าเราไม่ต้องการ , (comma) เป็นตัวแบ่งหลักเราจะทำอย่างไร? ลองดูตัวอย่างนี้ดู

```
/* FloatingPoint3.java - Calculating income with formatted output */

import java.text.*;   //for DecimalFormat
class FloatingPoint3 {
    public static void main(String[] args) {
        double weeklyPay = 1075 * 5; //1075 per day
        double extras = 1580;
        final double TAX_RATE = 8.5; //tax rate in percent

        //set format output
        DecimalFormat form = new DecimalFormat("#.00");
        //calculate tax and income
        double tax = (weeklyPay + extras) * TAX_RATE / 100;
        double totalIncome = (weeklyPay + extras) - tax;

        System.out.println("Tax = " + form.format(tax));
        System.out.println("Total income = " +
                           form.format(totalIncome));
    }
}
```

ผลลัพธ์ที่ได้คือ

```
Tax = 591.18
Total income = 6363.82
```

เราเพียงแต่เรียกใช้ class DecimalFormat แทน NumberFormat และเปลี่ยนการกำหนด format ให้เป็น

```
DecimalFormat form = new DecimalFormat("#.00);
```

เราก็จะได้ผลลัพธ์ที่มีเลขหลังจุดทศนิยมอยู่สองตัว และไม่มี , (comma) เป็นตัวแบ่งหลัก ยังมีวิธีการกำหนดรูปแบบของการแสดงผลลัพธ์อีกมากมายแต่เราก็คงจะพูดเพียงเท่านี้ก่อน

การประมวลผลข้อมูลต่างชนิดกัน (Mixed Type Calculations)

ในการประมวลผลข้อมูลที่มีความหลากหลายของชนิดในประโยคใด ๆ นั้น โดยทั่วไป Java จะทำการเปลี่ยนชนิดของข้อมูลที่มีขนาดเล็กกว่าให้เป็นขนาดที่ใหญ่ที่สุดในกลุ่มของข้อมูลนั้น ๆ แต่ผู้เขียนโปรแกรมจะต้องใช้ความระมัดระวังในเรื่องของการจัดเก็บข้อมูลที่มีขนาดใหญ่ไปไว้ในตัวแปรที่มีขนาดเล็ก เช่นเดียวกับที่ได้ทำมาก่อนหน้านี้ในเรื่องของการประมวลผลข้อมูลที่เป็น integer ลองมาดูตัวอย่างกัน

```
/* FloatingPoint4.java - Mixed-type calculations */

import java.text.*;   //for NumberFormat
class FloatingPoint4 {
    public static void main(String[] args) {
        double hourlyPay = 85.50;//85.50 per hour
        int hoursPerDay = 8;      //8 hours per day
        int numberOfDays = 5;     //total days worked
        int extraHoursWorked = 15;   //overtime hours
        final double TAX_RATE = 8.5;//tax rate in percent

        //set format output
        NumberFormat form = NumberFormat.getNumberInstance();
        form.setMaximumFractionDigits(2);
        //calculate weekly pay
        double weeklyPay = hourlyPay * hoursPerDay * numberOfDays;
        //calculate extra pay: one and a half time of
        //regular hourly pay
        double extras = extraHoursWorked * (hourlyPay +
                        (hourlyPay / 2.0));
        //calculate tax
        double tax = (weeklyPay + extras) * TAX_RATE / 100;
        //calculate total income
        double totalIncome = (weeklyPay + extras) - tax;

        System.out.println("Income before tax = " +
                           form.format((weeklyPay + extras)));
        System.out.println("Tax = " + form.format(tax));
        System.out.println("Income after tax = " +
                            form.format(totalIncome));
    }
}
```

โปรแกรม FloatingPoint4.java ใช้ตัวแปรที่มีชนิดเป็นทั้ง double และ integer ปนกันในการคำนวณหารายได้ต่อหนึ่งอาทิตย์ โดยกำหนดให้ตัวแปร hoursPerDay numberOfDays และ extraHoursWorked เป็น integer ส่วนตัวแปร hourlyPay weeklyPay extras tax และ totalIncome เรากำหนดให้เป็น double

การประมวลผลก็ไม่ยุ่งยากอะไร Java จะเปลี่ยนตัวแปรที่เป็น int ให้เป็น double ก่อนการประมวลผล เพราะฉะนั้นการจัดเก็บผลลัพธ์ก็ไม่มีค่าความคลาดเคลื่อนใด ๆ เกิดขึ้น ดังผลลัพธ์ที่แสดงนี้

```
Income before tax = 5,343.75
Tax = 454.22
Income after tax = 4,889.53
```

ข้อควรจำในการประมวลผลข้อมูลต่างชนิดกัน (ในประโยคเดียวกัน)

- ถ้ามีตัวแปรใดเป็น double ตัวแปรที่นำมาประมวลผลด้วยจะถูกเปลี่ยนให้เป็น double ก่อนการประมวลผล

- ถ้ามีตัวแปรใดเป็น float ตัวแปรที่นำมาประมวลผลด้วยจะถูกเปลี่ยนให้เป็น float ก่อนการประมวลผล
- ถ้ามีตัวแปรใดเป็น long ตัวแปรที่นำมาประมวลผลด้วยจะถูกเปลี่ยนให้เป็น long ก่อนการประมวลผล

สิ่งที่สำคัญก็คือข้อมูลที่เล็กกว่าในประโยคจะถูกเปลี่ยนให้เป็นชนิดที่ใหญ่ที่สุดในประโยคนั้น ๆ ซึ่งบางครั้งก็ไม่ใช่การประมวลผลที่เราต้องการ เช่น ตัวอย่างของโปรแกรมต่อไปนี้

```
/* MixedTypes.java - Mixed-type calculations */

import java.text.*;
class MixedTypes {
    public static void main(String[] args) {
        double result = 0.0;
        int one = 1, two = 2;

        result = 1.5 + one / two;
        //formatting output
        DecimalFormat f = new DecimalFormat("#.00");
        System.out.println("Result is " + f.format(result));
    }
}
```

ผลลัพธ์ที่เราคาดหวังว่าจะได้ คือ 2.0 แต่ Java กลับให้ 1.5 ซึ่งเป็นผลลัพธ์ที่เราไม่ต้องการ ถ้าสังเกตให้ดีจะเห็นว่าประโยค

```
result = 1.5 + one / two;
```

นั้น การหารจะเกิดก่อน และเนื่องจากว่าทั้ง one และ two เป็น int ดังนั้นผลลัพธ์ของการหารจึงมีค่าเป็น int และมีค่าเป็น 0 หลังจากนั้น Java ก็จะเปลี่ยน 0 ที่ได้ให้เป็น double แล้วจึงทำการบวกเข้ากับ 1.5 เพื่อให้ผลลัพธ์เป็นที่เราต้องการ ดังนั้นเราจึงต้องทำการ cast ให้ตัวแปร one เป็น double เสียก่อน ดังนี้

```
/* MixedTypes.java - Mixed-type calculations */

import java.text.*;
class MixedTypes {
    public static void main(String[] args) {
        double result = 0.0;
        int one = 1, two = 2;

        result = 1.5 + (double)one / two;
        //formatting output
        DecimalFormat f = new DecimalFormat("#.00);
        System.out.println("Result is " + f.format(result));
    }
}
```

เราก็จะได้ผลลัพธ์เป็น 2.0 ดังที่เราต้องการ

อย่างไรก็ตามการ cast นั้นต้องใช้ความระมัดระวังเป็นพิเศษ หากเราทำการ cast ข้อมูลที่ใหญ่กว่าไปเก็บไว้ในตัวแปรที่เล็กกว่าก็จะทำให้เกิดความคลาดเคลื่อนของข้อมูลได้ ดังที่ได้พูดไว้ก่อนหน้านี้

เรามาลองดูตัวอย่างอีกตัวอย่างหนึ่ง

```
/* FloatingPoint5.java - Mixed-type calculations */

import java.text.*;   //for NumberFormat
```

```
class FloatingPoint5 {
    public static void main(String[] args) {
        double hourlyPay = 85.50D;   //85.50 per hour
        int hoursPerDay = 8;         //8 hours per day
        float numberOfDays = 5F;     //total days worked
        int extraHoursWorked = 15;   //overtime hours
        final double TAX_RATE = 8.5D;//tax rate in percent

        //set format output
        NumberFormat form = NumberFormat.getNumberInstance();
        form.setMaximumFractionDigits(2);
        //calculate weekly pay
        double weeklyPay = hourlyPay * hoursPerDay * numberOfDays;
        //calculate extra pay: one and a half time of
                               regular hourly pay
        double extras = extraHoursWorked * (hourlyPay +
                        (hourlyPay / 2.0));
        //calculate tax
        double tax = (weeklyPay + extras) * TAX_RATE / 100;
        //calculate total income
        double totalIncome = (weeklyPay + extras) - tax;

        System.out.println("Income before tax = " +
                           form.format((weeklyPay + extras)));
        System.out.println("Tax = " + form.format(tax));
        System.out.println("Income after tax = " +
                           form.format(totalIncome));
    }
}
```

เราได้เปลี่ยนประโยค int numberOfDays = 5; ให้เป็น float numberOfDays = 5F; และผลลัพธ์ที่ได้ก็ยังคงถูกต้องเหมือนเดิมทั้งนี้เพราะ Java จะเปลี่ยน float ให้เป็น double เช่นเดียวกันกับที่เปลี่ยน int ให้เป็น double จากตัวอย่างก่อนหน้านี้

การใช้ Mathematical Functions ต่าง ๆ ที่ Java มีให้

หลาย ๆ ครั้งที่การเขียนโปรแกรมต้องการใช้ function ที่เกี่ยวข้องกับการหาค่าทางคณิตศาสตร์ เช่น การหาค่าของ sine cosine tangent หรือ ค่าอื่น ๆ ที่เกี่ยวข้องกับการคำนวณทางคณิตศาสตร์ Java ก็ได้ทำการออกแบบและจัดสรร function เหล่านี้ไว้ให้ผู้ใช้ได้เรียกใช้อย่างเต็มที่ เราจะมาทดลองเรียกใช้ function ต่าง ๆ เหล่านี้ดู

ตัวอย่างต่อไปนี้จะเป็นการคำนวณหาค่าของสมการ ที่เรียกว่า Quadratic Equation → $ax^2 + bx + c = 0$ ซึ่งหาได้จากการใช้สูตร

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

ซึ่ง mathematical function ที่เราต้องใช้ก็คือ การหา square root และ Java ได้จัดเตรียม method ตัวนี้ไว้ให้เราแล้วก็คือ Math.sqrt() เรามาลองดูโปรแกรมที่หาค่าของสมการตัวนี้กันดู

```
/* Quadratic.java - Calculating roots of function */
```

```
class Quadratic {
    public static void main(String[] args) {
        double a, b, c;
        //4x^2 + 20x + 16 = 0
        a = 4;
        b = 20;
        c = 16;
        //calculate determimant
        double det = Math.sqrt(b * b - 4 * a * c);
        //calculate roots
        double x1 = (-b + det) / (2 * a);
        double x2 = (-b - det) / (2 * a);

        System.out.println("Root 1 = " + x1);
        System.out.println("Root 2 = " + x2);
    }
}
```

ผลลัพธ์ที่ได้คือ

```
Root 1 = -1.0
Root 2 = -4.0
```

ในการเขียนโปรแกรม Quadratic.java ในครั้งนี้นั้นเราจะยังไม่สนใจในเรื่องที่ค่าของ determinant จะเป็น 0 เนื่องจากว่าเราไม่สามารถหาค่าของ square root ของ 0 ได้ เราจะใช้ตัวเลขที่ทำให้ค่าของ b2 – 4ac เป็นบวกเท่านั้นเพื่อหลีกเลี่ยงเหตุการณ์ดังกล่าว อีกอย่างหนึ่งก็คือ เรายังไม่ได้เรียนเรื่องการใช้ ชุดคำสั่งในการตรวจสอบ (if –else) เราจะกลับมาดูตัวอย่างนี้อีกครั้งหนึ่งเมื่อเราได้เรียนเรื่องของการใช้ ชุดคำสั่งในการตรวจสอบ ในที่นี้เราสนใจแต่เรื่องของการใช้ Math.sqrt() เท่านั้นเอง

เรามาลองใช้ Mathematical function อื่น ๆ ดูกันดีกว่า

```
/* MaxMin.java - Finding max and min */

class MaxMin {
    public static void main(String[] args) {
        int num1 = 58, num2 = 97, num3 = 15;

        //finding maximum between num1, num2, and num3
        int maxOfTwo = Math.max(num1, num2);
        int maxOfAll = Math.max(maxOfTwo, num3);

        //finding minimum between num1, num2, and num3
        int minOfTwo = Math.min(num1, num2);
        int minOfAll = Math.min(minOfTwo, num3);

        System.out.println("Maximum number is " + maxOfAll);
        System.out.println("Minimum number is " + minOfAll);
    }
}
```

โปรแกรม MaxMin.java แสดงการเรียกใช้ method max() และ min() ในการเปรียบเทียบและหาค่าที่มากที่สุด และค่าที่น้อยที่สุดของ integer 3 ตัว ตามลำดับ และเนื่องจากว่า method ทั้งสองยอมรับ parameter เพียงแค่สองตัวเท่านั้น เราจึงต้องทำการหาค่าสองครั้ง ครั้งแรกระหว่าง integer 2 ตัวแรก และครั้งที่สองระหว่างค่าที่หาได้ กับ integer ที่เหลือ และเมื่อ execute โปรแกรมเราก็จะได้ผลลัพธ์ดังที่เห็นนี้

```
Maximum number is 97
```

```
Minimum number is 15
```

โปรแกรมที่แสดงให้ดูต่อไปนี้เป็นโปรแกรมที่เรียกใช้ method ต่าง ๆ ที่มีอยู่ใน class Math

```
/* Maths.java - Other math functions */

class Maths {
    public static void main(String[] args) {
        double angle = 0.7853982;//45 degrees angle in radians

        System.out.println("sin(45) is " + Math.sin(angle));
        System.out.println("cos(45) is " + Math.cos(angle));
        System.out.println("tan(45) is " + Math.tan(angle));

        //absolute value of -456
        System.out.println("Absolute value of -456 is " +
                           Math.abs(-456));
        //two to the fourth power
        System.out.println("2 to the 4th power is " +
                           Math.pow(2D, 4D));
        //round to the nearest integer
        System.out.println("2.345267 rounded to " +
                           Math.round(2.345267));
        //round to the nearest integer
        System.out.println("2.859 rounded to " + Math.round(2.859));
        //the remainder of 3.25 / 2.25
        System.out.println("The remainder of 3.25 / 2.25 is " +
                                Math.IEEEremainder(3.25, 2.25));
        //nearest integer of PI
        System.out.println("The nearest integer of PI is " +
                           Math.rint(Math.PI));
    }
}
```

โปรแกรม Maths.java ใช้ method จาก class Math ทำการหาค่าของ sine cosine และ tangent ของมุม 45 ซึ่งมีค่าเท่ากับ 0.7853982 radians สาเหตุที่ต้องใช้มุมที่เป็น radians ก็เพราะว่า method เหล่านี้บังคับให้มุมที่รับเข้าเป็น parameter มีชนิดเป็น radians

หลังจากนั้นก็ทำการหาค่า absolute ของ -456 ค่า 2^4 และค่าการปัดเศษในรูปแบบต่าง ๆ ผลลัพธ์ที่ได้จากการ run คือ

```
sin(45) is 0.7071068070684596
cos(45) is 0.7071067553046345
tan(45) is 1.0000000732051062
Absolute value of -456 is 456
2 to the 4th power is 16.0
2.345267 rounded to 2
2.859 rounded to 3
The remainder of 3.25 / 2.25 is 1.0
The nearest integer of PI is 3.0
```

โปรแกรมตัวอย่างการใช้ method toRadians() สำหรับการเปลี่ยนมุมที่เป็น degrees ให้เป็น radians ถ้าเราไม่รู้ว่ามุมที่เราเคยชินอยู่ด้วยการวัดแบบ degrees มีค่าเป็นเท่าไรในการวัดแบบ radians

```
/* Maths1.java - Other math functions */

import java.text.DecimalFormat;
class Maths1 {
```

```
    public static void main(String[] args) {
        double angle = 45;    //45 degrees angle
        double sin, cos, tan;

        //set format output
        DecimalFormat f = new DecimalFormat("0.00000");
        //calculate sin, cos, and tan
        sin = Math.sin(Math.toRadians(angle));
        cos = Math.cos(Math.toRadians(angle));
        tan = Math.tan(Math.toRadians(angle));

        System.out.println("sin(" + angle + ") = " + f.format(sin));
        System.out.println("cos(" + angle + ") = " + f.format(cos));
        System.out.println("tan(" + angle + ") = " + f.format(tan));

    }
}
```

โปรแกรมตัวอย่างอีกตัวหนึ่งที่ใช้ math function ในการคำนวณหาค่าของรัศมีของวงกลม จากพื้นที่ของวงกลมที่กำหนดให้

```
/* Radius.java - Other math functions */

import java.text.DecimalFormat;
class Radius {
    public static void main(String[] args) {
        double radius;
        double area = 850.0;
        int meters, centimeters;

        //calculate radius
        radius = Math.sqrt(area / Math.PI);
        //convert to meters and centimeters
        meters = (int)Math.floor(radius);
        centimeters = (int)Math.round(100.0 * (radius - meters));

        System.out.print("Circle with area of " + area +
                         " square meters ");
        System.out.print("has a radius of " + meters +
                         " meters and ");
        System.out.println(centimeters + " centimeters");
    }
}
```

เราใช้ floor() ในการหาค่าของรัศมีที่เป็นจำนวนเต็ม และใช้ round() ในการคำนวณหาค่าของ centimeters ที่เหลือจากการนำเอา meters ไปลบออกจาก radius ซึ่งเมื่อ run ก็จะได้ผลลัพธ์ดังนี้

```
Circle with area of 850.0 square meters has a radius of 16 meters and
45 centimeters
```

ยังมี method อีกหลายตัวที่ผู้เขียนโปรแกรมอาจต้องใช้ ซึ่งได้สรุปไว้ในตารางที่ 2.6

ตาราง 2.6 Mathematical Functions สำหรับการหาค่าทางตรีโกณ

Method	Function	Argument	Result Type
sin(arg)	หาค่า sine	double (radians)	double
cos(arg)	หาค่า cosine	double (radians)	double
tan(arg)	หาค่า tangent	double (radians)	double

asin(arg)	หาค่า arc sine	double	double (radians ระหว่าง $-\pi/2$ และ $\pi/2$)
acos(arg)	หาค่า arc cosine	double	double (radians ระหว่าง 0.0 และ π)
atan(arg)	หาค่า arc tangent	double	double (radians ระหว่าง $-\pi/2$ และ $\pi/2$)
atan2(arg1, arg2)	หาค่า arc tangent of arg1 และ arg2	double	double (radians ระหว่าง $-\pi$ และ π)
abs(arg)	หาค่า absolute ของ arg	int long float double	เหมือนกันกับ Argument
max(arg1, arg2)	หาค่าสูงสุดระหว่าง arg1 และ arg2	int long float double	เหมือนกันกับ Argument
min(arg1, arg2)	หาค่าต่ำสุดระหว่าง arg1 และ arg2	int long float double	เหมือนกันกับ Argument
ceil(arg)	หาค่าที่เล็กที่สุดที่ใหญ่กว่าหรือเท่ากับ arg	double	double
floor(arg)	หาค่าที่ใหญ่ที่สุดที่น้อยกว่าหรือเท่ากับ arg	double	double
round(arg)	หาค่า integer ที่ใกล้เคียงกับ arg ที่สุด	float double	ได้ int ถ้า argument เป็น float และ long ถ้าเป็น double
rint(arg)	หาค่า integer ที่ใกล้เคียงกับ arg ที่สุด	double	double
IEEEremainder(arg1, arg2)	หาค่าเศษที่เหลือจากการหาร arg1 ด้วย arg2	double	double
sqrt(arg)	หาค่า root ที่สองของ arg	double	double
pow(arg1, arg2)	หาค่าของ arg1 ยกกำลัง arg2	double	double
exp(arg)	หาค่า e ยกกำลัง arg	double	double
log(arg)	หาค่า log ของ arg	double	double
random()	หาค่าสุ่มที่มากกว่า 0.0 แต่น้อยกว่า 1.0	ไม่มี	double

การใช้ข้อมูลที่เป็น Character

เราได้ดูการประกาศตัวแปรที่เป็น character มาก่อนหน้านี้เพียงตัวอย่างเดียว เพื่อให้เกิดความเข้าใจถึงการใช้ข้อมูลที่เป็น character ใน Java เรามาลองดูตัวอย่างการใช้ character ในรูปแบบต่าง ๆ กันอีกสักหน่อย

```
/* Characters.java - Using character data */

class Characters {
    public static void main(String[] args) {
        char H, e, l, o;

        H = 72;    //ASCII for 'H'
        e = 101;   //ASCII for 'e'
        l = 108;   //ASCII for 'l'
        o = 111;   //ASCII for 'o'
```

```
        System.out.print(H);
        System.out.print(e);
        System.out.print(l);
        System.out.print(l);
        System.out.print(o);
        System.out.println(" World");
    }
}
```

ผลลัพธ์จากการ run โปรแกรม Characters.java ก็คือ Hello World ซึ่งตัวโปรแกรมเองใช้รหัส ASCII ในการกำหนดค่าให้กับตัวแปร H e l และ o ซึ่งเมื่อนำไปแสดงผลก็จะได้ค่าที่ตัวแปรเหล่านั้นเป็นอยู่ เรามาลองดูการประมวลผลด้วยการใช้ character ด้วย operator ง่าย ๆ ที่มีอยู่

```
/* Characters1.java - Using character data */

class Characters1 {
    public static void main(String[] args) {
        char ch1 = 88, ch2 = 77;

        System.out.println("ch1 = " + ch1);
        System.out.println("ch2 = " + ch2);

        ch1 -= 23; //ASCII for 'A'
        ch2 += 6;  //ASCII for 'S'

        System.out.println("ch1 = " + ch1);
        System.out.println("ch2 = " + ch2);
    }
}
```

โปรแกรมเริ่มต้นด้วยการกำหนดค่าให้ ch1 และ ch2 มีค่าเป็นตัวอักษร X และ M ตามลำดับ และเมื่อแสดงผลของทั้งสองตัวแปรแล้ว โปรแกรมก็เปลี่ยนค่าให้กับตัวแปร ch1 และ ch2 ด้วยการลดค่าของ ch1 ลงเป็นจำนวนเท่ากับ 23 และเพิ่มค่าให้กับตัวแปร ch2 อีก 6 เพื่อให้ได้ตัวอักษร A และ X และเมื่อ execute โปรแกรมเราก็จะได้ผลลัพธ์ทั้งหมดเป็น

```
ch1 = X
ch2 = M
ch1 = A
ch2 = S
```

ทีนี้เรามาลองดูการใช้ Unicode ในการแสดงผลตัวอักษรต่าง ๆ

```
/* Characters2.java - Using character data */

class Characters2 {
    public static void main(String[] args) {
        char ch1 = '\u0058';  //unicode for 'X'
        char ch2 = '\u004D';  //unicode for 'M'
        char ch3 = '\u0041';  //unicode for 'A'
        char ch4 = '\u0053';  //unicode for 'S'

        System.out.println("ch1 = " + ch1);
        System.out.println("ch2 = " + ch2);
        System.out.println("ch3 = " + ch3);
        System.out.println("ch4 = " + ch4);
    }
```

```
}
```

ซึ่งเมื่อ run แล้วก็จะได้ผลลัพธ์เหมือนกันกับผลลัพธ์ของโปรแกรม Characters1.java

เราสามารถที่จะใช้ Unicode ในการแสดงตัวอักษรในภาษาไทยได้ เช่นเดียวกันกับที่เราใช้ Unicode ในการแสดงตัวอักษรในภาษาอังกฤษ แต่ว่าระบบของเราจะต้องเอื้อต่อการแสดงผลที่เป็นตัวอักษรไทยด้วย ไม่เช่นนั้นแล้วผลลัพธ์ที่ได้ก็จะเป็นตัวอักษรที่อ่านไม่รู้เรื่อง หรือไม่ก็แสดงผลเป็นเครื่องหมาย '?' ทุกตัว แต่ถ้าเราใช้ Unicode เพื่อการแสดงผลใน applet เราก็จะไม่เจอปัญหาใด ๆ เลยเพราะ Unicode ได้ถูกออกแบบมาเพื่อแสดงผลบน applet อยู่แล้ว

ตารางที่ 2.7 แสดงถึงตัวอักษรที่ใช้ในการแสดงผลลัพธ์ที่ไม่ใช่ตัวอักษรธรรมดา หรือที่ภาษาอังกฤษเรียกว่า Character Escape Sequences ซึ่งเราสามารถนำมาใช้ในการจัดวางรูปแบบของผลลัพธ์ที่เราต้องการให้ดูแล้วสวยงามขึ้น (อย่างหยาบ ๆ และ ง่าย ๆ)

ตาราง 2.7 Character Escape Sequence

Escape Sequence	ความหมาย
\ddd	แสดงผลลัพธ์เป็นเลขฐานแปด (Octal)
\uxxxx	แสดงตัวอักษร Unicode (Hexadecimal)
\'	ตัว ' (Single quote)
\"	ตัว " (Double quote)
\\	ตัว \ เอง
\r	กลับไปยังจุดเริ่มต้นของบรรทัด (Carriage return)
\n	ขึ้นบรรทัดใหม่ (New line หรือ Linefeed)
\f	เลื่อนไปหน้าถัดไป (Form feed)
\t	เลื่อนไปยังตำแหน่งถัดไป (Tab)
\b	เลื่อนไปทางซ้ายหนึ่งตัวอักษร

การใช้ตัวแปรชนิด boolean

เราได้เห็นการประกาศตัวแปรที่เป็น boolean มาเพียงน้อยนิดก่อนหน้านี้ ต่อไปเราจะมาทำความเข้าใจกับข้อมูลที่เป็น boolean และการประมวลตัวแปรหรือประโยคที่มีค่าของการประมวลผลเป็น boolean อีกสักเล็กน้อยก่อนที่เราจะใช้มันในโอกาสต่อไป ผู้เขียนหลายท่านเรียก operator เหล่านี้ว่า Logical Operator

โดยทั่วไปเราสามารถประมวลผลข้อมูลที่เป็น boolean ด้วย operator ต่าง ๆ ที่เห็นนี้

&& หรือที่เรียกว่า Conditional AND
|| หรือที่เรียกว่า Conditional OR
& หรือที่เรียกว่า Logical AND
| หรือที่เรียกว่า Logical OR
! หรือที่เรียกว่า Logical NOT

&& และ || มีการประมวลผลในรูปแบบที่หนังสือหลาย ๆ เล่มให้ชื่อว่า Lazy หรือ Short Circuit ซึ่งมีรูปแบบการประมวลผลที่แตกต่างจาก & และ |

operator && ต่างจาก & โดยที่ && นั้นจะไม่ประมวลผล ประโยคหรือตัวแปรที่ตามหลังตัวมันเอง ถ้าประโยคหรือตัวแปรที่อยู่ก่อนหน้ามีค่าเป็น false เพราะว่าค่าที่เป็น false เมื่อนำมาประมวลผลกับค่าอื่น ๆ แล้วจะได้ false เสมอ

operator || ก็เช่นเดียวกันกับ && แต่จะประมวลผลประโยคหรือตัวแปรที่ตามมาเมื่อประโยคแรกหรือตัวแปรตัวแรกมีค่าเป็น true เพราะว่าประโยคหรือตัวแปรที่มีค่าเป็น true เมื่อนำมาประมวลกับค่าใด ๆ ก็ตามจะได้ค่า true เสมอ

ส่วน operator & และ | จะประมวลผลประโยคและตัวแปรทั้งสองเสมอ

และ operator ! นั้นเมื่อไปอยู่หน้าตัวแปรหรือประโยคที่มีค่าทาง boolean ค่าใดค่าหนึ่งก็จะเปลี่ยนค่านั้นให้เป็นตรงกันข้ามทันที

Operator ตัวอื่น ๆ ที่ใช้กับตัวแปรหรือประโยคที่มีค่าเป็น boolean มีดังนี้

< ใช้ในการเปรียบเทียบว่า ค่าทางด้านซ้ายของเครื่องหมาย น้อยกว่า ค่าทางด้านขวาหรือไม่
> ใช้ในการเปรียบเทียบว่า ค่าทางด้านซ้ายของเครื่องหมาย มากกว่า ค่าทางด้านขวาหรือไม่
<= ใช้ในการเปรียบเทียบว่า ค่าทางด้านซ้ายของเครื่องหมาย น้อยกว่าหรือเท่ากับ ค่าทางด้านขวาหรือไม่
>= ใช้ในการเปรียบเทียบว่า ค่าทางด้านซ้ายของเครื่องหมาย มากกว่าหรือเท่ากับ ค่าทางด้านขวาหรือไม่
== ใช้ในการเปรียบเทียบว่า ค่าทางด้านซ้ายของเครื่องหมาย เท่ากับ ค่าทางด้านขวาหรือไม่
!= ใช้ในการเปรียบเทียบว่า ค่าทางด้านซ้ายของเครื่องหมาย ไม่เท่ากับ ค่าทางด้านขวาหรือไม่

เราเรียก operator เหล่านี้ว่า Relational operator ซึ่งเอาไว้ใช้ในการตรวจสอบความสัมพันธ์ของตัวแปรหรือประโยคว่ามีค่าแตกต่างกันอย่างไรเมื่อนำมาเปรียบเทียบกัน

เราจะลองเขียนโปรแกรมตรวจสอบ operator ต่าง ๆ เหล่านี้ดู

```
/* BooleanOp.java - Testing boolean operations */

class BooleanOp {
    public static void main(String[] args) {
        boolean A = true, B = false;

        System.out.println("Conditional AND");
        System.out.println("F && F is " + (B && B));
        System.out.println("F && T is " + (B && A));
        System.out.println("T && F is " + (A && B));
        System.out.println("T && T is " + (A && A));

        System.out.println("Conditional OR");
        System.out.println("F || F is " + (B || B));
        System.out.println("F || T is " + (B || A));
        System.out.println("T || F is " + (A || B));
        System.out.println("T || T is " + (A || A));

        System.out.println("Logical AND");
        System.out.println("F & F is " + (B & B));
        System.out.println("F & T is " + (B & A));
        System.out.println("T & F is " + (A & B));
        System.out.println("T & T is " + (A & A));

        System.out.println("Logical OR");
        System.out.println("F | F is " + (B | B));
        System.out.println("F | T is " + (B | A));
        System.out.println("T | F is " + (A | B));
        System.out.println("T | T is " + (A | A));

        System.out.println("Logical NOT");
        System.out.println("!F is " + (!B));
        System.out.println("!T is " + (!A));
    }
}
```

โปรแกรมประกาศตัวแปรสองตัว A และ B ด้วยค่า true และ false ตามลำดับ และทำการแสดงผลลัพธ์ของการประมวลผลด้วย operator ผ่านทาง System.out.println() ผลลัพธ์ที่แสดงออกทางหน้าจอ คือ

```
Conditional AND
F && F is false
F && T is false
T && F is false
T && T is true

Conditional OR
F || F is false
F || T is true
T || F is true
T || T is true

Logical AND
F & F is false
F & T is false
T & F is false
T & T is true

Logical OR
F | F is false
F | T is true
T | F is true
T | T is true

Logical NOT
!F is true
!T is false
```

ทีนี้ลองมาดูโปรแกรมการใช้ Relational operator

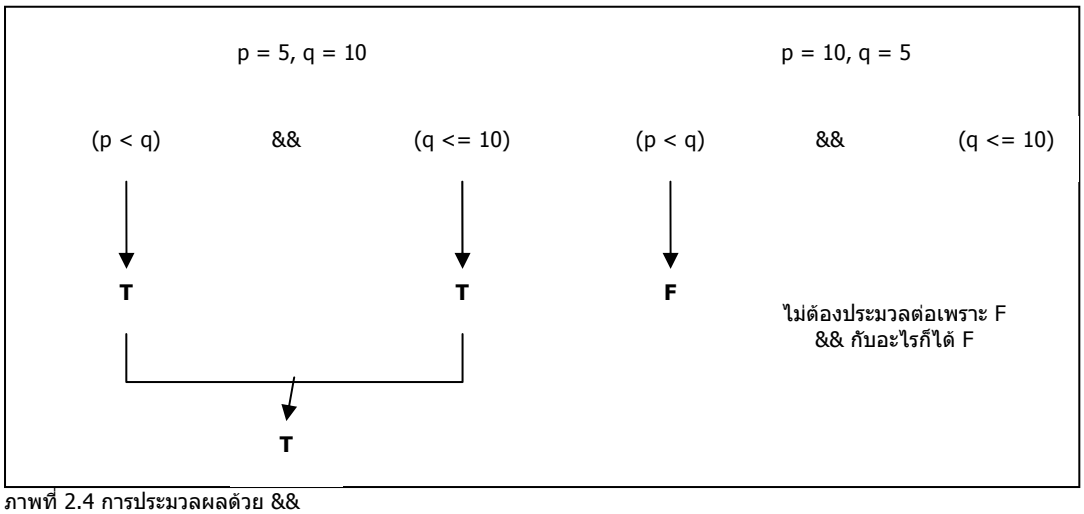
```
/* BooleanOp2.java - Using relational operators */

class BooleanOp2 {
    public static void main(String[] args) {
        int i = 5, j = 10;

        System.out.println("i = " + i);
        System.out.println("j = " + j);
        System.out.println("i > j is " + (i > j));
        System.out.println("i < j is " + (i < j));
        System.out.println("i >= j is " + (i >= j));
        System.out.println("i <= j is " + (i <= j));
        System.out.println("i == j is " + (i == j));
        System.out.println("i != j is " + (i != j));
        System.out.println("(i < 10) && (j < 10) is " + ((i < 10) &&
                           (j < 10)));
        System.out.println("(i < 10) || (j < 10) is " + ((i < 10) ||
                           (j < 10)));
    }
}
```

โปรแกรม BooleanOp2.java ทำการหาค่าของประโยคต่าง ๆ ในรูปแบบง่าย ๆ เพื่อให้ผู้อ่านได้เห็นถึงผลลัพธ์ของการใช้ operator ต่าง ๆ ได้อย่างชัดเจน เราประกาศตัวแปรที่เป็น int สองตัว คือ i และ j ซึ่งกำหนดให้มีค่าเป็น 5 และ 10 ตามลำดับ หลังจากนั้นโปรแกรมก็ใช้ operator ต่าง ๆ กับตัวแปรสองตัวนี้

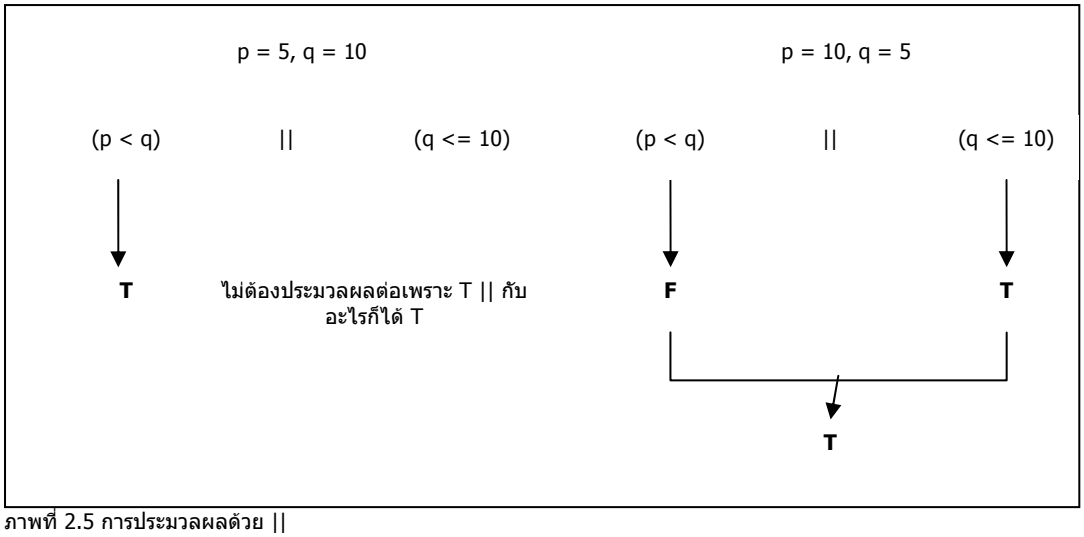
และเมื่อ run โปรแกรมนี้แล้วผลลัพธ์ที่ได้ คือ

```
i = 5
j = 10
i > j is false
i < j is true
i >= j is false
i <= j is true
i == j is false
i != j is true
(i < 10) && (j < 10) is false
(i < 10) || (j < 10) is true
```

เพื่อให้เห็นภาพของการประมวลของ operator && และ operator || ได้ชัดเจนยิ่งขึ้น ผู้อ่านควรตรวจสอบจากแผนภาพของการประมวลผลนี้ โดยเราได้กำหนดให้ตัวแปร p และ q มีค่าเป็น 5 และ 10 ตามลำดับในการประมวลผลทางด้านซ้าย และให้มีค่าเป็น 10 และ 5 ในการประมวลผลทางด้านขวา



ภาพที่ 2.4 การประมวลผลด้วย &&



ภาพที่ 2.5 การประมวลผลด้วย ||

ตาราง 2.8 ได้สรุปค่าของการประมวลผลด้วย operator && || & | ดังที่ได้กล่าวมาแล้ว

ตาราง 2.8 การใช้ operator && || & และ |

P	Q	P && Q	P || Q	P & Q	P | Q
true	true	true	true	true	true
false	true	false	true	false	true
true	false	false	true	false	true
false	false	false	false	false	false

เราคงมองไม่เห็นถึงประโยชน์ของการใช้ตัวแปรที่เป็น boolean และการใช้ operator ต่าง ๆ ที่ได้พูดถึงในตอนนี้อย่างมากมายนัก การประมวลผลของตัวแปรที่เป็น boolean ส่วนใหญ่จะเอาไว้ช่วยการตรวจสอบถึงเหตุการณ์ (condition) ต่าง ๆ ที่อาจเกิดขึ้นในโปรแกรม เช่น ถ้าประโยคที่หนึ่ง เมื่อนำมาประมวลผลกับ ประโยคที่สองแล้ว ผลลัพธ์ที่ได้เป็น true หรือ false ซึ่งถ้าเป็น true เราจะทำการอย่างอื่น และก็จะทำการอื่นหากเป็น false เป็นต้น เราจะพูดถึง boolean operator ต่าง ๆ เมื่อเราพูดถึงเรื่องของ logic และเรื่องของการประมวลผลในรูปแบบของการวนซ้ำ (logic and loop)

การประมวลผลในระดับ bit ด้วยการใช้ operator Shift และ Bitwise

ในการทำงานในระดับ bit นั้นมี operator ไม่กี่ตัวที่เราสามารถเรียกใช้ได้ ซึ่งการใช้งานโดยทั่วไปก็ไม่ง่ายนัก เพราะต้องทำในระดับ bit และผู้ใช้ต้องมีความเข้าใจในเรื่องของเลขฐานสองอย่างดีพอสมควร เราจะยกตัวอย่างเพียงไม่กี่ตัวอย่างเพื่อให้เห็นถึงการใช้งานของ operator shift ต่าง ๆ ที่ Java มีให้ คือ << >> และ >>>

ตาราง 2.9 การใช้ operator shift << >> และ >>>

Operator	วิธีใช้	ความหมาย
<<	operand1 << operand2	โยก bit ของ operand1 ไปทางซ้ายจำนวนเท่ากับค่าของ operand2 เติม 0 ทุก ๆ bit ทางขวา
>>	operand1 >> operand2	โยก bit ของ operand1 ไปทางขวาจำนวนเท่ากับค่าของ operand2 เติมด้วยค่า bit (sign bit)สูงสุดทางซ้าย
>>>	operand1 >>> operand2	โยก bit ของ operand1 ไปทางขวาจำนวนเท่ากับค่าของ operand2 เติม 0 ทุก ๆ bit ทางซ้าย

ตัวอย่าง

13 >> 1
13 = 00001101_2 เมื่อ shift ไปทางขวา 1 bit ผลลัพธ์ที่ได้คือ 00000110_2 (เท่ากับ 6 ในเลขฐาน 10)

13 << 1
shift ไปทางซ้าย 1 bit ผลลัพธ์ที่ได้คือ 00011010_2 (เท่ากับ 26 ในเลขฐาน 10)

13 >>> 1
shift ไปทางขวา 1 bit ผลลัพธ์ที่ได้คือ 00000110_2 (เท่ากับ 6 ในเลขฐาน 10)

การโยก bit ไปทางซ้ายครั้งละหนึ่ง bit นั้นที่จริงแล้วก็คือการคูณด้วยสอง เรามาดูตัวอย่างโปรแกรมการโยก bit ไปทางซ้ายกัน จากโปรแกรม ShiftLeftOp.java ที่เห็นด้านล่างนี้

```
/* ShiftLeftOp.java - Using shift operators */

class ShiftLeftOp {
    public static void main(String[] args) {
        byte bits = 1;

        System.out.println("Value of bits is " + bits);
        bits = (byte) (bits << 1);
        System.out.println("Value of bits is " + bits);
        bits = (byte) (bits << 1);
```

```
        System.out.println("Value of bits is " + bits);
        bits = (byte) (bits << 1);
        System.out.println("Value of bits is " + bits);
        bits = (byte) (bits << 1);
        System.out.println("Value of bits is " + bits);
        bits = (byte) (bits << 1);
        System.out.println("Value of bits is " + bits);
    }
}
```

เราเริ่มด้วยการกำหนดให้ตัวแปร bits มีชนิดเป็น byte และมีค่าเท่ากับหนึ่ง โปรแกรมทำการโยก bit ไปทางซ้ายจำนวนเท่ากับห้าครั้ง ซึ่งในการโยกแต่ละครั้งเราก็จะได้ค่าใหม่ที่มีค่าเท่ากับค่าเดิมคูณด้วยสอง ดังที่เห็นจากผลลัพธ์ของการ run โปรแกรมนี้

```
Value of bits is 1
Value of bits is 2
Value of bits is 4
Value of bits is 8
Value of bits is 16
Value of bits is 32
```

เราเลือกใช้ข้อมูลชนิดที่เป็น byte ก็เพื่อให้เห็นว่าถ้าเราโยก bit อีกครั้งด้วยจำนวน bit ที่ต้องการโยกเท่ากับสอง bit (ครั้งที่หก) เราจะได้ค่าที่ตัวแปร bits ไม่สามารถเก็บได้ ลองดู code ที่เพิ่ม และ ผลลัพธ์ของการ run

```
bits = (byte) (bits << 2);
System.out.println("Value of bits is " + bits);
```

ผลลัพธ์เมื่อ run อีกครั้งด้วย code ที่เพิ่มขึ้นจะเห็นว่าค่าของ bits ที่ได้มีค่าเป็น -128 ซึ่งเป็นค่าที่ Java เป็นผู้กำหนดให้ แต่ไม่ใช่ค่าที่ถูกต้อง เพราะข้อมูลที่เป็น byte นั้นเก็บค่าสูงสุดที่เป็นบวกได้เพียงแค่ 127 เท่านั้น

```
Value of bits is 1
Value of bits is 2
Value of bits is 4
Value of bits is 8
Value of bits is 16
Value of bits is 32
Value of bits is -128
```

สิ่งสำคัญอีกสิ่งหนึ่งก็คือ การ cast เราจำเป็นต้องเปลี่ยนข้อมูลให้ถูกต้องตามชนิดของตัวแปร เนื่องจากว่า Java ได้ทำการเปลี่ยนตัวแปร bits ของเราให้เป็น int ก่อนที่จะทำการ shift ดังนั้นเราต้องเปลี่ยนข้อมูลให้กลับมาเป็นชนิดเดิมก่อนการจัดเก็บ (หลังจากการ shift สิ้นสุดลง)

เราได้พูดถึงการประกาศตัวแปร การกำหนดค่าให้กับตัวแปรที่อยู่ภายในตัวโปรแกรม ด้วยข้อมูลทั้งที่มีชนิดเป็น integer และ floating-point ซึ่งการกำหนดค่าให้กับตัวแปรภายในโปรแกรมนั้น ไม่มีความยืดหยุ่นของข้อมูลที่ถูกเก็บอยู่ในตัวแปรนั้น หากผู้ใช้ หรือ ผู้เขียนโปรแกรมต้องการที่จะเปลี่ยนค่าที่ตัวแปรนั้นเก็บอยู่ ผู้เขียนจะต้องเป็นผู้แก้ไขแต่เพียงฝ่ายเดียว ผู้ใช้โปรแกรมไม่สามารถที่จะเปลี่ยนค่าเหล่านั้นได้ ดังนั้นการเขียนโปรแกรม จะต้องให้โอกาสผู้ใช้โปรแกรมในการเปลี่ยนแปลงค่าเหล่านั้น Java มี method อยู่หลายตัวที่รองรับการใส่ข้อมูลผ่านทาง keyboard เราจะมาทำความรู้จัก และเข้าใจถึงวิธีการ ซึ่งค่อนข้างจะซับซ้อนพอสมควรสำหรับผู้เริ่มหัดเขียนโปรแกรมใหม่ อย่างไรก็ตามหลังจากที่ได้ทำความเข้าใจแล้ว ก็จะเห็นว่าการรับค่าต่าง ๆ ผ่านทาง keyboard นั้นเป็นเรื่องที่ (จริง ๆ แล้ว) ไม่ยุ่งยากเลย

การกำหนดค่าให้กับตัวแปรผ่านทางสื่อการนำเข้าข้อมูลมาตรฐาน

Java ไม่มี method ใด ๆ ที่รับค่าข้อมูลตามชนิดนั้น ๆ ของข้อมูลที่เข้ามาเลย เช่น ถ้าผู้ใช้โปรแกรมต้องการที่จะใส่ค่าที่เป็น int float double หรือ char ผู้เขียนโปรแกรมก็ไม่สามารถที่จะรับค่าต่าง ๆ เหล่านี้ได้โดยตรง เหมือนกับการเขียนโปรแกรมด้วยภาษาอื่น ๆ เช่น C หรือ C++ แต่ตัวผู้เขียนเองจะต้องเขียน code ขึ้นมารองรับค่าเหล่านี้ผ่านทางการอ่านเข้าแบบที่เรียกว่า การรับข้อมูลที่เป็น String ซึ่งเมื่ออ่าน String ได้แล้วก็ต้องแปลง String ให้เป็นชนิดของข้อมูลที่ผู้เขียนโปรแกรมต้องการให้เป็น

ผู้อ่านอาจสงสัยว่า String คืออะไร ในที่นี้จะอธิบายอย่างคร่าว ๆ เนื่องจากว่าเราจะพูดถึง String ในบทอื่นต่อไป ใน Java นั้น String เป็น object (อีกแล้ว object? เช่นเดียวกันเราจะพูดถึง object อย่างละเอียดในบทอื่น) ซึ่งสามารถเก็บตัวอักษรได้คราวละมาก ๆ ขอยกตัวอย่างง่าย ๆ ให้ดูก็แล้วกัน

People of Thailand
สวัสดี
price_of_coffee
1234567890
23+popOfRock

ตัวอย่างที่เห็นด้านบนนี้ล้วนแล้วแต่เป็น String ทั้งนั้น ทีนี้เรามาลองดูวิธีการอ่าน String เข้ามาจาก keyboard แล้วก็เปลี่ยน String ให้เป็นข้อมูลชนิดต่าง ๆ ที่เราต้องการ เริ่มด้วยการเปลี่ยนแปลง code ของโปรแกรม Radius.java ให้รับค่าของ area จาก keyboard แทนการกำหนดภายในโปรแกรม

```
/* Radius2.java - Reading data from keyboard */
import java.io.*;

class Radius2 {
    public static void main(String[] args) throws IOException {
        double radius, area;
        int meters, centimeters;

        //prompt user for area of a circle
        System.out.print("Please enter area of a circle: ");
        //get string from keyboard
        InputStreamReader in = new InputStreamReader(System.in);
        BufferedReader buffer = new BufferedReader(in);
        String inputString = buffer.readLine();

        //convert string to double
        area = Double.parseDouble(inputString);

        //calculate radius
        radius = Math.sqrt(area / Math.PI);
        //convert to meters and centimeters
        meters = (int)Math.floor(radius);
        centimeters = (int)Math.round(100.0 * (radius - meters));

        System.out.print("Circle with area of " + area +
                         " square meters ");
        System.out.print("has a radius of " + meters +
                         " meters and ");
        System.out.println(centimeters + " centimeters");
    }
}
```

โปรแกรม Raadius2.java เพิ่ม code ให้อ่าน String จาก keyboard ด้วยประโยค

```
System.out.print("Please enter area of a circle: ");
```

```
InputStreamReader in = new InputStreamReader(System.in);
BufferedReader buffer = new BufferedReader(in);
String inputString = buffer.readLine();
```

ประโยคแรกสุดเป็นการส่งข้อความไปยัง user ให้รู้ว่าต้องใส่ข้อมูลอะไรเข้าสู่โปรแกรม ประโยคถัดมาอ่านข้อมูลในรูปแบบของ byte จากระบบ (System.in) เปลี่ยนข้อมูลเหล่านั้นให้เป็น character แล้วจึงนำมาเก็บไว้ในตัวแปร in ประโยคที่สามทำการจองเนื้อที่ให้กับตัวแปร buffer พร้อมกับนำค่า character ที่อยู่ในตัวแปร in มาเก็บไว้ในรูปแบบของ String เมื่อได้ String (ซึ่งตอนนี้อยู่ในตัวแปร buffer) เราก็ทำการอ่าน String ที่ว่านี้ด้วย method readLine() ผ่านทางตัวแปร buffer แล้วจึงนำไปเก็บไว้ในตัวแปรชนิด String ที่มีชื่อว่า inputString

ฟังดูแล้วรู้สึกว่ามีขั้นตอนมากมายในการอ่าน String แต่ตอนนี้ขอเพียงแต่จำว่าถ้าจะอ่าน String จาก keyboard ต้องทำด้วยวิธีนี้ (ณ เวลานี้ เพราะมีอีกหลายวิธี)

เมื่อเราได้ String แล้วขั้นตอนต่อไปก็คือการเปลี่ยน String ให้เป็นข้อมูลชนิดที่เราต้องการใช้ในโปรแกรม ซึ่งโปรแกรม Radius.java ของเราต้องการใช้ข้อมูลชนิดที่เป็น double เราจึงต้องทำการเปลี่ยน String นี้ให้เป็น double ด้วยการเรียกใช้ method parseDouble() จาก class Double พร้อมทั้งส่งตัวแปรที่เก็บข้อมูลจาก keyboard ไปให้แก่ method นี้ด้วย

```
area = Double.parseDouble(inputString);
```

หลังจากนั้น code ของโปรแกรมที่เหลือก็เหมือนกับที่เราได้เขียนก่อนหน้านี้ และเมื่อ run โปรแกรมดูสองครั้ง ด้วยค่า 850 ในครั้งแรกและค่า 54 ในครั้งที่สอง เราก็ได้ผลลัพธ์ ดังที่แสดงให้เห็นนี้

```
Please enter area of a circle: 850
Circle with area of 850.0 square meters has a radius of 16 meters and
45 centimeters

Please enter area of a circle: 54
Circle with area of 54.0 square meters has a radius of 4 meters and
15 centimeters
```

ถ้าสังเกตให้ดีจะเห็นว่าโปรแกรม Radius2.java นั้นมีข้อความเพิ่มต่อท้ายของ main(..) คือ

```
throws IOException
```

เหตุผลที่เราต้องเพิ่ม throws IOException ให้กับ main(…) ก็เพราะว่า การอ่านข้อมูลเข้าสู่โปรแกรมนั้นบางครั้งก็อาจเจอกับปัญหาในการอ่าน เช่น อ่านไม่ได้ และเหตุผลอีกอันหนึ่ง คือ Java บังคับให้เราต้องตรวจสอบการทำงานกับ I/O เอง ซึ่งถ้าหากมีปัญหาเกิดขึ้น เราจะได้แก้ไขได้ถูกต้อง มีวิธีการตรวจสอบและป้องกันไม่ให้การอ่านมีปัญหาหลากหลาย ซึ่งเราจะได้ดูกันในโอกาสต่อไป (บทที่ 7) ลองมาดูว่าถ้าเราไม่ใส่ข้อความที่ว่าลงไปในโปรแกรม Radius2.java เราจะได้ error อะไร

```
Radius2.java:16: unreported exception java.io.IOException; must be
caught or declared to be thrown
       String inputString = buffer.readLine();
                                           ^
1 error
```

จะเห็นว่า Java จะบอกให้เรารู้ว่าเราไม่สามารถที่จะอ่านข้อมูลเข้าสู่โปรแกรมได้ ถ้าเราไม่ทำการตรวจสอบด้วยวิธีที่ได้กล่าวไว้ ประโยคที่ว่า

must be caught or *declared to be thrown*

กล่าวถึงวิธี สองวิธี ที่เราสามารถใช้ได้ในการอ่านข้อมูลเข้าสู่โปรแกรม คือ (1) ตรวจจับ – catch และ (2) เรียกใช้ throws ของ Java

ต่อไปจะได้แสดงการใช้วิธีที่หนึ่ง ซึ่งเป็นการตรวจจับ error ในการอ่านข้อมูล แต่จะพูดเพียงเพื่อให้การอ่านข้อมูลทำได้สมบรูณ์เท่านั้น คงจะไม่กล่าวถึงอย่างละเอียดมากมายนัก เรามาเริ่มต้นด้วยการเขียน code เพิ่มขึ้นใหม่ ดังที่แสดงให้เห็นนี้

```
/* Radius2.java - Reading data from keyboard */
import java.io.*;

class Radius2 {
    public static void main(String[] args) throws IOException {
        double radius, area;
        int meters, centimeters;
          String inputString = null;

        try {
            //prompt user for area of a circle
            System.out.print("Please enter area of a circle: ");
            //get string from keyboard
            InputStreamReader in = new InputStreamReader(System.in);
            BufferedReader buffer = new BufferedReader(in);
            inputString = buffer.readLine();
        }
        catch(IOException ioe) {}

        //convert string to double
        area = Double.parseDouble(inputString);

        //calculate radius
        radius = Math.sqrt(area / Math.PI);
        //convert to meters and centimeters
        meters = (int)Math.floor(radius);
        centimeters = (int)Math.round(100.0 * (radius - meters));

        System.out.print("Circle with area of " + area +
                         " square meters ");
        System.out.print("has a radius of " + meters +
                         " meters and ");
        System.out.println(centimeters + " centimeters");
    }
}
```

สิ่งที่เราได้ปรับปรุงคือประโยค ต่อไปนี้

```
String inputString = null;

//prompt user for area of a circle
System.out.print("Please enter area of a circle: ");
try {
    //get string from keyboard
    InputStreamReader in = new InputStreamReader(System.in);
    BufferedReader buffer = new BufferedReader(in);
    inputString = buffer.readLine();
}
catch(IOException ioe) {}
```

เราย้ายการประกาศตัวแปร inputString ไปอยู่ก่อนประโยคที่อ่านข้อมูลเข้าพร้อมกับเพิ่มประโยค

```
try {
    …
```

```
    …
    …
}
catch(IOException ioe) {}
```

เข้าสู่โปรแกรมด้วยการนำเอาประโยคการอ่านข้อมูลเข้าสู่ string ทั้งสามประโยคไว้ใน block ของ try {} ส่วนคำว่า catch(IOException ioe) {} นั้นเราก็เพิ่มเข้าหลังจาก block ของ try {} เราไม่จำเป็นต้องใส่ code อะไรเข้าไปใน code ของ catch(IOExcpetion ioe) {} เพราะเราเพียงแต่ต้องการให้การอ่านเป็นไปได้ด้วยดีเท่านั้น เรายังจะไม่กังวลถึง error ที่เกิดขึ้น ซึ่งถ้ามี error เกิดขึ้น Java ก็จะเป็นผู้ฟ้องให้เราทันที

เพื่อให้เกิดความเข้าใจอีกสักหน่อย เราจะมาตรวจสอบการใส่ข้อมูลของผู้ใช้ดูว่า ได้มีการใส่ข้อมูลเข้าสู่โปรแกรม หรือเพียงแต่กดปุ่ม Enter เฉย ๆ เราทำการตรวจสอบด้วยการเพิ่ม try {..} และ catch() {} ให้กับโปรแกรม Radius2.java ดังนี้

```
/* Radius2.java - Reading data from keyboard */
import java.io.*;

class Radius2 {
    public static void main(String[] args) {
        double radius, area;
        int meters, centimeters;
        String inputString = null;

        //prompt user for area of a circle
        System.out.print("Please enter area of a circle: ");
        try {
            //get string from keyboard
            InputStreamReader in = new InputStreamReader(System.in);
            BufferedReader buffer = new BufferedReader(in);
            inputString = buffer.readLine();

            try {
                //convert string to double
                area = Double.parseDouble(inputString);
                //calculate radius
                radius = Math.sqrt(area / Math.PI);
                //convert to meters and centimeters
                meters = (int)Math.floor(radius);
                centimeters = (int)Math.round(100.0 *
                              (radius - meters));

                System.out.print("Circle with area of " + area +
                                 " square meters ");
                System.out.print("has a radius of " + meters +
                                 " meters and ");
                System.out.println(centimeters + " centimeters");
            }
            //catch empty string (user hits Enter)
            catch(Exception e) {
                System.out.println("Error! you must supply input");
                   e.printStackTrace();
                System.exit(1);
            }
        }
        //catch other I/O error - do nothing
        catch(IOException ioe) { /* leave it to Java! */ }
    }
```

```
}
```

โปรแกรม Radius2.java ที่เราเขียนขึ้นใหม่มี try {…} และ cath() {} อยู่สองตัว ตัวแรกใช้ตรวจสอบเรื่องของการอ่านทั่วไปที่ได้กล่าวไปแล้ว ส่วนตัวที่สองเป็นการตรวจสอบว่าผู้ใช้มีการใส่ข้อมูลหรือไม่ ซึ่งถ้าไม่ใส่ เราก็จะ

1. ส่งข้อความไปยังหน้าจอด้วยคำสั่ง System.out.println("Error! you must supply input");
2. ส่งข้อความ error ที่ Java ได้ฟ้องไว้ ด้วยคำสั่ง e.printStackTrace();
3. ออกจากโปรแกรมโดยไม่มีการประมวลผลใด ๆ ทั้งสิ้น ด้วยคำสั่ง System.exit(1);

และเมื่อ execute ดูเราก็จะได้ข้อความดังนี้

```
Please enter area of a circle: < ผู้ใช้กดปุ่ม Enter >
Error! you must supply input
java.lang.NumberFormatException: empty String
        at
java.lang.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:9
86)
        at java.lang.Double.parseDouble(Double.java:202)
        at Radius2.main(Radius2.java:22)
```

โดยทั่วไปการออกจากโปรแกรมที่ไม่มีปัญหาใด ๆ ในการเขียนด้วยภาษา Java นั้นเราจะใช้การออกด้วยเลข 1 ในประโยค System.exit(1) หรือเลขอื่น ๆ ที่ไม่ใช่ 0 ส่วนการออกแบบปกตินั้นเราใช้เลข 0 แทนเลข 1

เราคงจะพักการตรวจสอบ หรือ ตรวจจับ error ไว้เพียงแค่นี้ก่อนเราจะกลับมาพูดถึงการเขียน code รองรับ error ในบทอื่นต่อไป

สรุป

เราได้เรียนรู้ถึงข้อมูลชนิดต่าง ๆ ที่เป็นข้อมูลแบบพื้นฐาน หรือที่เรียกว่า primitive datatype ที่ Java มีให้ รวมไปถึงการประกาศตัวแปร การกำหนดค่าให้กับตัวแปร การประมวลตัวแปรชนิดเดียวกัน และ ต่างชนิดกัน การเปลี่ยนชนิดของข้อมูล (casting) การใช้ method ที่เกี่ยวข้องกับตัวเลข การแสดงผลตัวเลขที่มีจุดทศนิยม การอ่านข้อมูลเข้าสู่โปรแกรม และการตรวจสอบ และตรวจจับ error ที่อาจเกิดขึ้นในการอ่านข้อมูล

แบบฝึกหัด

1. จงอธิบายถึงความหมายของคำว่า final พร้อมทั้งยกตัวอย่างประกอบ

2. จงอธิบายถึงความแตกต่างระหว่าง 5, 5.0, '5', "5", และ "5.0"

3. จงหาผลลัพธ์ของการประมวลผลประโยคต่อไปนี้ ถ้ากำหนดให้

   ```
   double x = 4.5;
   double y = -45.25;
   int p = 12;
   int q = 2;
   ```

   3.1. x + p * y / (p – y) * x
   3.2. p % (p – q)
   3.3. (int) y
   3.4. (int) p
   3.5. Math.round(x)
   3.6. (int) Math.round(y)

3.7. -2 * y + (int) x
3.8. q / p
3.9. p % q
3.10. y * y – y + (x * (int) x)

4. จงเขียนโปรแกรมที่รับตัวเลขชนิด int จาก keyboard จำนวน 2 ตัว และข้อมูลชนิด double อีก 2 ตัว หลังจากนั้นให้โปรแกรมคำนวณหาผลรวมของเลขทั้ง 4 ตัว ให้เก็บผลลัพธ์ที่หาได้ไว้ในตัวแปรตัวใหม่ ส่งผลลัพธ์ที่ได้ไปยังหน้าจอ

5. จงเขียนประโยคด้วยภาษา Java จากสมการที่ให้นี้

5.1. a + bc / 2
5.2. a(a – c) + b(b – c) – (c(c – a) / d)
5.3. $b^2 – 4ac$
5.4. $(4/3)^2 / 2$
5.5. $q = \left(\frac{T_1 x M}{D - k} \right) + T_2$

6. จงเขียนโปรแกรมที่ทำการเปลี่ยน องศา Celsius ให้เป็นองศา Fahrenheit โดยกำหนดให้ข้อมูลที่ต้องการเปลี่ยน นำเข้าจาก keyboard โดยกำหนดให้สูตรการเปลี่ยน คือ

Celsius = $\frac{5}{9}$ (Fahrenheit – 32)

7. จงเขียนโปรแกรมที่ทำการเปลี่ยน องศา Fahrenheit ให้เป็นองศา Celsius โดยกำหนดให้ข้อมูลที่ต้องการเปลี่ยน นำเข้าจาก keyboard

8. จงเขียนโปรแกรมที่รับข้อมูลที่เป็นจำนวนกิโลกรัมจาก keyboard เปลี่ยนให้เป็น mile ส่งผลลัพธ์ไปยังหน้าจอ

9. จงเขียนโปรแกรมที่อ่านข้อมูลจาก keyboard ที่เป็นอายุของผู้ใช้ในรูปแบบของปี ให้โปรแกรมเปลี่ยนเป็นจำนวนวัน ส่งผลลัพธ์ออกทางหน้าจอ (กำหนดให้ 1 ปีเท่ากับ 365 วัน)

10. จงเขียนโปรแกรมที่รับ int จำนวน 5 ตัว หาค่าเฉลี่ยของตัวเลขเหล่านี้ เสร็จแล้วส่งผลลัพธ์ไปยังหน้าจอ

11. จงเขียนโปรแกรมที่คำนวณหาเวลาที่แสงเดินทางจากโลกไปยังดวงอาทิตย์ ให้ส่งผลลัพธ์ของการคำนวณในรูปแบบของนาที ส่งผลลัพธ์ที่หาได้ไปยังหน้าจอ

12. สมมติว่าอาจารย์ทุกคนในวิทยาลัยแห่งหนึ่งดื่ม beer 1 ขวดต่อวัน และในการทำ beer 1 ลัง (24 ขวด) นั้นต้องใช้ข้าว barley จำนวน 1 ถัง ถ้าจำนวนของอาจารย์ในวิทยาลัยที่ว่านี้มีทั้งสิ้น 200 คน จงเขียนโปรแกรมในการคำนวณหาจำนวนของข้าว barley ที่ต้องใช้ในการทำ beer ให้เพียงพอต่อการบริโภคของอาจารย์ในวิทยาลัยนี้ตลอดภาคการศึกษา

13. จงเขียนโปรแกรมที่รับข้อมูลชนิด long ที่มีจำนวนของตัวเลข (digit) อยู่ระหว่าง 10 – 12 ตัว ให้ส่งผลลัพธ์ในรูปแบบของ ตัวเลขที่ใช้เครื่องหมาย , เป็นตัวแบ่งกลุ่ม โดยเริ่มต้นจากทางขวามือ และให้มีตัวเลขกลุ่มละ 3 ตัวไปยังหน้าจอ

14. จงเขียนโปรแกรมที่รับค่าความยาวของด้านทุกด้านของสามเหลี่ยม จากนั้นให้คำนวณหาค่าของเส้นรอบรูป (perimeter) ส่งผลลัพธ์ไปยังหน้าจอ

15. จงเขียนโปรแกรมที่คำนวณหาพื้นที่ของวงกลม จากรัศมีที่รับเข้ามาจาก keyboard

16. จงเขียนโปรแกรมที่รับข้อมูลชนิด double จำนวน 5 ตัวจาก keyboard จากนั้นให้คำนวณหา ค่าเฉลี่ยของตัวเลขเหล่านั้น ส่งผลลัพธ์ที่หาได้ไปยังหน้าจอ

17. กำหนดให้เส้นผ่าศูนย์กลางของดวงอาทิตย์เท่ากับ 865,000 ไมล์ และเส้นผ่าศูนย์กลางของโลกเท่ากับ 7,600 ไมล์ จงเขียนโปรแกรมที่คำนวณหา

    17.1. ปริมาตรของโลก (cubic mile หรือ $mile^3$)
    17.2. ปริมาตาของดวงอาทิตย์ (cubic mile)
    17.3. อัตราส่วนของปริมาตรของดวงอาทิตย์ต่อปริมาตรของโลก

18. จงเขียนโปรแกรมที่รับข้อมูลชนิด double ที่อยู่ในรูปแบบ 123.4567 จาก keyboard จากนั้นให้ดึงเอาส่วนของข้อมูลที่อยู่หน้าจุดทศนิยมไปเก็บไว้ในตัวแปรที่มีชนิดเป็น long ตัวหนึ่ง เสร็จแล้วให้เก็บข้อมูลที่อยู่หลังจุดทศนิยมไปเก็บไว้ในตัวที่เป็น long อีกตัวหนึ่ง ส่งผลลัพธ์ที่อยู่ในตัวแปร long ทั้งสองไปยังหน้าจอ

3
การประมวลผลแบบวน (Repetition)

ในบทที่สามนี้เราจะมาทำความรู้จักกับการทำงานในรูปแบบของการประมวลผลชุดคำสั่งซ้ำ ๆ กันตามเงื่อนไขที่กำหนดให้ หรือที่เรียกว่าการประมวลผลแบบวน (repetition หรือ loop) การใช้ชุดคำสั่งในการตัดสินใจ และ การใช้ประโยคในการเปรียบเทียบข้อมูลต่าง ๆ

หลังจากจบบทเรียนนี้แล้วผู้อ่านจะได้ทราบถึง

- การเปรียบเทียบข้อมูล (comparison)
- การกำหนดประโยคในการเปรียบเทียบ (logical expression)
- การเปลี่ยนแปลงขั้นตอนการประมวลผลในแบบต่าง ๆ (control statement)
- การทำงานแบบวน (loop)
  - for loop
  - while loop
  - do ... while loop
- การยุติการทำงานของ loop ด้วยการใช้คำสั่ง break และ continue

หลาย ๆ ครั้งการเขียนโปรแกรมต้องมีการเลือกที่จะประมวลผล ไม่ว่าจะเป็นการเลือกตามข้อมูลที่นำเข้า หรือ การเลือกประมวลผลตามขั้นตอน กระบวนการที่ได้กำหนดไว้ เช่น ถ้าข้อมูลนำเข้ามีค่าน้อยกว่าศูนย์ เราจะยุติการทำงานของโปรแกรม ถ้ามากกว่าเราจะมวลผลต่อไป อะไรทำนองนี้ ในการตัดสินใจที่จะให้ code ทำอย่างใด อย่างหนึ่งนั้น เราจะต้องรู้จักวิธีการสร้างประโยคที่ใช้ในการเปรียบเทียบ ขั้นตอนของการใช้ประโยคเปรียบเทียบในรูปแบบต่าง ๆ

การตัดสินใจ และ การเปรียบเทียบ

การตัดสินใจเป็นส่วนประกอบที่สำคัญมาก ๆ ส่วนหนึ่งของการเขียนโปรแกรม โปรแกรมของเราต้องมีความสามารถในการตัดสินใจเลือกว่าจะทำอะไรต่อไป เช่น ถ้าจำนวนของนักศึกษาที่มาสมัครเรียนในภาควิชาคอมพิวเตอร์ธุรกิจมีมากกว่า ห้าร้อย เราจะมีการสอบคัดเลือก หรือ เช่น ถ้าเงินเดือนเพิ่มขึ้นฉันจะซื้อบ้าน อะไรทำนองนี้ ซึ่งในด้านของการเขียน code นั้น การตัดสินใจจะทำได้โปรแกรมก็ต้องมีความสามารถที่จะเปรียบเทียบได้ก่อน เช่น เปรียบเทียบถึงความมากกว่า น้อยกว่า หรือ เปรียบเทียบถึงความเท่ากัน และความไม่เท่ากัน

ในบทที่สองเราได้พูดถึง relational operator เพียงเล็กน้อยในเรื่องของการทำงานกับตัวแปรที่มีชนิดเป็น boolean และ operator ที่ว่านี้ก็คือ < <= > >= == และ != เราจะมาดูถึงการใช้ operator เหล่านี้ในประโยคที่มีการเลือกประมวลผลตามผลลัพธ์ของการเปรียบเทียบนั้น ๆ

ประโยคที่ใช้ if

if เป็นคำสั่งที่เรานำมาใช้ในการเลือกว่าเราจะประมวลผลอะไรตามผลลัพธ์ที่เกิดขึ้น if มีรูปแบบดังนี้ คือ

```
if(expression)
    statement;
```

โดยมีข้อกำหนดว่า expression จะต้องเป็นประโยคที่เมื่อมีการประมวลผลแล้วจะได้ค่าที่เป็น boolean ซึ่งเป็นไปได้เพียงแค่ true หรือ false เท่านั้น เช่น

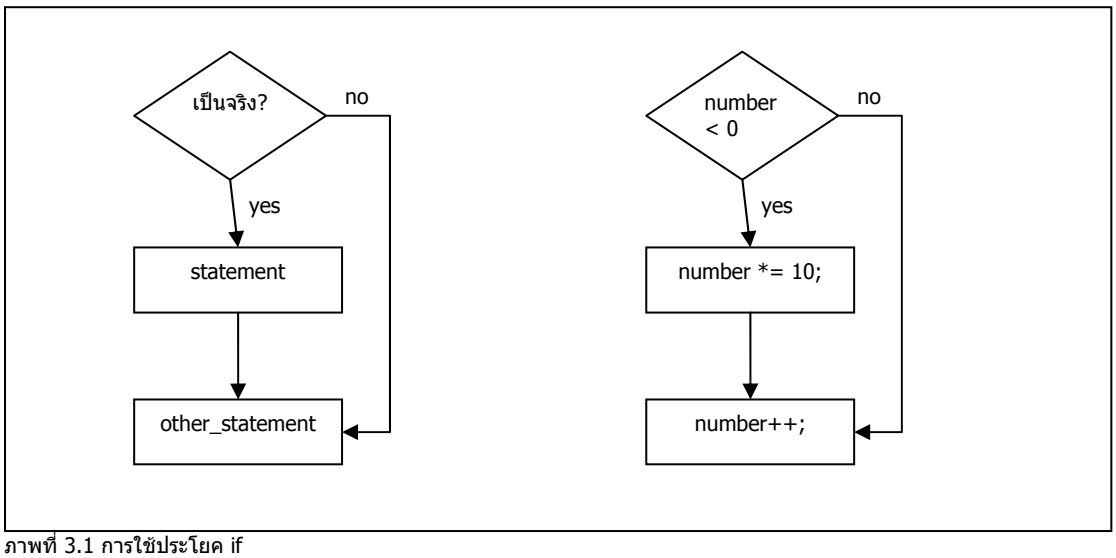
```
if(numberOfStudents >= 500)
    entranceExam = true;
```

หรือ

```
if(number < 0)
    number *= 10;
number++;
```

ถ้าค่าของ expression เป็นจริง ประโยคที่ตามหลัง if จะถูกประมวลผล แต่ถ้าเป็นเท็จก็จะไม่ถูกประมวลผล เช่นตัวอย่างที่เห็นด้านบนนี้ ถ้าค่าของ number < 0 เป็นจริงประโยค number *= 10 ก็จะได้รับการประมวลผล ภาพที่ 3.1 แสดงถึงขั้นตอนของการประมวลผลด้วยการใช้ if กับประโยคในรูปแบบทั่วไป และประโยคการเปรียบเทียบ number ที่ได้กล่าวถึงก่อนหน้านี้

if(expression)
    statement;
other_statement;



ภาพที่ 3.1 การใช้ประโยค if

ในบางครั้งเราก็อาจเขียนประโยค if ให้เป็นรูปแบบที่อยู่ในบรรทัดเดียวกัน เช่น ถ้าเราเปลี่ยนประโยคตัวอย่างก่อนหน้านี้ให้อยู่ในบรรทัดเดียวกันเราก็จะได้

```
if(number < 0) number *= 10;
number++;
```

ซึ่งถูกต้องตามหลักไวยากรณ์ของ Java แต่โดยทั่วไปแล้วการขึ้นบรรทัดใหม่สำหรับประโยคที่ตามมาหลังจากการเปรียบเทียบสิ้นสุดลง จะทำให้ดู code ได้ง่ายขึ้น ลองดูโปรแกรมตัวอย่างการตรวจสอบว่าตัวเลขที่ user ใส่เข้าสู่โปรแกรมเป็นเลขคู่หรือไม่

```
/* IfStatement.java - Using if statement to find out whether a number
 * is odd or even
*/

import java.io.*;
import java.lang.Integer;

class IfStatement {
    public static void main(String[] args) throws IOException {
```

```
        int number;

        //set up buffer stream
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader buffer = new BufferedReader(isr);

        //get input string
        System.out.print("Enter a number (int): ");
        String input = buffer.readLine();
        //convert string to int
        number = Integer.parseInt(input);

        //check if number is even or not
        if(number % 2 == 0)
            System.out.println(number + " is an even number.");
    }
}
```

โปรแกรม IfStatement.java บอก user ให้ใส่ตัวเลขหนึ่งตัว ซึ่งโปรแกรมได้จัดเก็บไว้ในตัวแปรชื่อ number หลังจากนั้นก็ทำการตรวจสอบว่าตัวเลขที่อ่านมาเป็นเลขคู่หรือไม่ด้วยประโยค

```
if(number % 2 == 0)
    System.out.println(number + " is an even number.");
```

ซึ่งถ้านำสองไปหาร number แล้วไม่มีเศษ (หารลงตัว) แสดงว่าตัวเลขที่ว่านี้เป็นเลขคู่ ซึ่งโปรแกรมก็จะแสดงข้อความไปยังหน้าจอบอกผู้ใช้ว่า ตัวเลขที่ใส่เข้ามาเป็นเลขคู่ ดังเช่นผลลัพธ์ของการ run นี้

```
Enter a number (int): 45

Enter a number (int): 20
20 is an even number.
```

โปรแกรมของเรายังไม่ใช่โปรแกรมที่สมบูรณ์ ดังจะเห็นได้จากผลลัพธ์ที่ได้ ถ้าเราใส่ 45 โปรแกรมก็จะไม่แสดงอะไรเลย จะแสดงก็ต่อเมื่อตัวเลขที่ใส่เป็นเลขคู่เท่านั้น เราจะกลับมาดูว่าเราต้องทำอย่างไร ถึงจะสามารถที่จะแสดงข้อความไปยังหน้าจอโดยแบ่งแยกว่าตัวเลขนั้นเป็น เลขคู่หรือเลขคี่

การใช้ if ในประโยคที่มีมากกว่าหนึ่งประโยค (if – block)

การเขียน code ที่มีมากกว่าหนึ่งประโยคนั้น ทำได้ด้วยการใช้เครื่องหมาย {} ล้อมรอบประโยคเหล่านั้น (compound statement) ดังตัวอย่างที่เห็นนี้

```
if(expression) {
    statement 1;
    statement 2;
    statement 3;
    …
    statement n;
}
```

เช่น

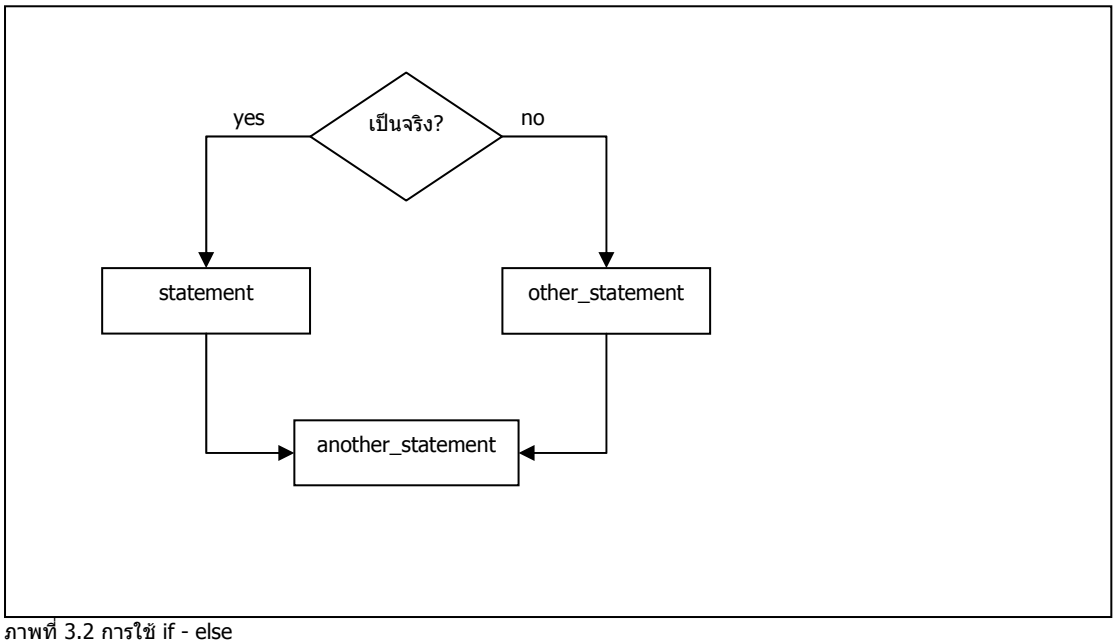
```
if(number % 2 == 0) {
    System.out.println(number + " is an even number.");
    number++;
    System.out.println("But " + number + " is not!");
}
```

ทีนี้เราจะกลับไปยังประเด็นที่เราพูดถึงก่อนหน้านี้ เกี่ยวกับเรื่องของการแสดงผลทั้งที่เป็นจริง และไม่เป็นจริง ในการตรวจสอบด้วยการใช้ if นั้นเราสามารถที่จะบังคับการประมวลผลให้เกิดขึ้นทั้งที่เป็นจริง และเป็นเท็จ อย่างใด อย่างหนึ่งด้วยการใช้ else ตามหลังประโยคที่ตาม if มาอีกทีหนึ่ง

รูปแบบโครงสร้างของประโยค if – else นี้มีดังนี้

```
if(expression)
    statement;
else
    other_statement;
another_statement;
```

ภาพที่ 3.2 แสดงถึงโครงสร้างของ if – else



ภาพที่ 3.2 การใช้ if - else

จากโปรแกรม IfStatement.java ถ้าเราต้องการที่จะแสดงผลถ้าตัวเลขที่ user ใส่เข้าสู่โปรแกรมเป็นเลขคี่ เราก็ต้องเปลี่ยน code ของเราให้เป็นดังนี้

```
/* IfElse.java - Using if - else statement to find out whether a
 * number is odd or even
*/

import java.io.*;
import java.lang.Integer;

class IfElse {
    public static void main(String[] args) throws IOException {
        int number;

        //set up buffer stream
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader buffer = new BufferedReader(isr);

        //get input string
        System.out.print("Enter a number (int): ");
        String input = buffer.readLine();
```

```
        //convert string to int
        number = Integer.parseInt(input);

        //check if number is even or not
        if(number % 2 == 0)
            System.out.println(number + " is an even number.");
        else
            System.out.println(number + " is an odd number.");

        System.out.println("Good Bye.");
    }
}
```

ประโยคที่เราได้เขียนเพิ่มเข้าสู่โปรแกรมของเรา คือ

```
else
    System.out.println(number + " is an odd number.");

System.out.println("Good Bye.");
```

ซึ่งจะทำให้โปรแกรมของเราทำการประมวลผลประโยคที่อยู่ถัดมาจาก else ถ้า number % 2 == 0 เป็นเท็จ และตัวเลขที่จะทำให้ประโยคนี้ถูกประมวลผลก็คือ ตัวเลขที่เป็นเลขคี่ (นำ 2 ไปหารแล้วเหลือเศษ) ลองดูผลลัพธ์ของการ run นี้

```
Enter a number (int): 45
45 is an odd number.
Good Bye.

Enter a number (int): 80
80 is an even number.
Good Bye.
```

จะเห็นว่าการนำเอา if – else มาใช้ทำให้โปรแกรมของเราสนองตอบความต้องการในการแสดงผล เมื่อการตรวจสอบได้ผลที่เป็นจริงทางหนึ่ง และเมื่อผลของการตรวจสอบเป็นเท็จ อีกทางหนึ่ง

เนื่องจาก if – else ก็เป็น statement ตัวหนึ่งดังนั้นเราก็สามารถที่จะนำเอา if – else ไปใส่ไว้ในประโยคที่เป็น if – else อีกประโยคหนึ่งได้ หรือแม้แต่กระทั่งหลาย ๆ ประโยค เราเรียกประโยคที่ใช้ if – else แบบนี้ว่า Nested – if ซึ่งมีรูปแบบโครงสร้าง ที่มีลักษณะคล้าย ๆ กับที่เห็นนี้ (อาจมีประโยค if – else มากกว่า หรือน้อยกว่า)

```
if(expression 1) {
    statement 1;
    statement 2;
}
else
    if(expression 2)
        statement 3;
    else {
        statement 4;
        statement 5;
    }
```

หรือ

```
if(expression 1) {
    statement 1;
    statement 2;
}
else if(expression 2)
    statement 3;
else {
    statement 4;
    statement 5;
}
```

จากโครงสร้างตัวอย่างด้านบนนี้ประโยคที่จะได้รับการประมวลผลถ้า expression 1 เป็นจริงคือ statement 1 และ statement 2 ส่วน statement 3 จะได้รับการประมวลผล ถ้า expression 2 เป็นจริง และ statement 4 และ statement 5 จะได้รับการประมวลผลก็ต่อเมื่อ expression 1 และ expression 2 เป็นเท็จ

ลองมาดูโปรแกรมตัวอย่างการใช้ Nested – if ในโปรแกรม Grades.java

```
/* Grades.java - Using Nested - if finding letter grade
 * from a given score
*/

import java.io.*;
import java.lang.Integer;

class Grades {
    public static void main(String[] args) throws IOException {
        int score;
        char grade;

        //set up buffer stream
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader buffer = new BufferedReader(isr);

        //get input string
        System.out.print("Enter your score: ");
        String input = buffer.readLine();
        //convert string to int
        score = Integer.parseInt(input);

        //determine grade by given score
        if(score >= 90)
            grade = 'A';
        else if(score >= 80)
            grade = 'B';
        else if(score >= 70)
            grade = 'C';
        else if(score >= 60)
            grade = 'D';
        else
            grade = 'F';

        System.out.println("Your grade is " + grade);
    }
}
```

โปรแกรม Grades.java ใช้เกณฑ์การให้เกรดคือ ถ้า score มีค่า

มากกว่าหรือเท่ากับ 90 ได้ A
มากกว่าหรือเท่ากับ 80 ได้ B (อยู่ระหว่าง 80 ถึง 89)
มากกว่าหรือเท่ากับ 70 ได้ C (อยู่ระหว่าง 70 ถึง 79)
มากกว่าหรือเท่ากับ 60 ได้ D (อยู่ระหว่าง 60 ถึง 69)
ต่ำกว่า 60 ได้ F

เราได้ออกแบบให้การตรวจสอบเป็นไปได้ทุกกรณีด้วยการใช้การเปรียบเทียบจาก 90 ลงไปถึง 60 ตามขั้นตอน คือ ถ้า user ใส่ score ที่มีค่าเท่ากับ 75 การเปรียบใน if แรก และ if ที่สองจะเป็นเท็จ แต่จะมาเป็นจริงใน if ที่สามคือ score >= 70 ประโยคที่ตามมา grade = 'C' ก็จะได้รับการประมวลผล ส่วนในกรณีสุดท้ายนั้น ประโยค grade = 'F' จะได้รับการประมวลผลก็ต่อเมื่อ ทุก ๆ กรณีของการตรวจสอบก่อนหน้านี้เป็นเท็จทั้งหมด ผู้อ่านควรตรวจสอบประโยค if – else จากตัวอย่างด้วยค่า score ที่แตกต่างกันเพื่อให้เกิดความเข้าใจมากยิ่งขึ้น

ผลลัพธ์ของการ run ด้วยข้อมูลบางส่วน

```
Enter your score: 87
Your grade is B

Enter your score: 45
Your grade is F

Enter your score: 92
Your grade is A
```

เราสามารถที่จะเขียนโปรแกรม Grades.java ขึ้นใหม่ด้วยการใช้ operator && หรือ || ช่วยในการตรวจสอบถึงสถานะภาพของ score ว่าอยู่ในกลุ่มใด ลองมาดูกัน

```
/* Grades2.java - Using Nested - if finding letter grade
 * from a given score
*/

import java.io.*;
import java.lang.Integer;

class Grades2 {
    public static void main(String[] args) throws IOException {
        int score;
        char grade;

        //set up buffer stream
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader buffer = new BufferedReader(isr);

        //get input string
        System.out.print("Enter your score: ");
        String input = buffer.readLine();
        //convert string to int
        score = Integer.parseInt(input);

        //determine grade by given score
        if(score >= 90)
            grade = 'A';
        else if(score >= 80 && score <= 89)
            grade = 'B';
        else if(score >= 70 && score <= 79)
            grade = 'C';
        else if(score >= 60 && score <= 69)
            grade = 'D';
        else
            grade = 'F';

        System.out.println("Your grade is " + grade);
    }
}
```

เราเปลี่ยน code ในการตรวจสอบใหม่ให้เป็น

```
if(score >= 90)
    grade = 'A';
else if(score >= 80 && score <= 89)
    grade = 'B';
else if(score >= 70 && score <= 79)
```

```
    grade = 'C';
else if(score >= 60 && score <= 69)
    grade = 'D';
else
    grade = 'F';
```

ซึ่งเมื่อได้ลอง run ดูแล้วก็ได้ผลลัพธ์เหมือนกันกับที่ได้จากโปรแกรม Grades.java แต่ขั้นตอนของการตรวจสอบไม่เหมือนกัน เราได้พูดก่อนหน้านี้ว่า ถ้ามีการใช้ && และค่าของประโยคที่อยู่ทางด้านหน้าของ && มีค่าเป็นจริง ประโยคที่อยู่ทางด้านหลังก็จะได้รับการประมวลผลด้วย เช่น สมมติว่า score มีค่าเป็น 85 การประมวลผลใน if แรกจะเป็นเท็จ Java ก็จะประมวลผล if ที่สอง คือ else if(score >= 80 && score <= 89) ซึ่งที่นี่เองที่ Java จะประมวลผลประโยคที่อยู่หลัง && ด้วยเพราะประโยคที่อยู่ทางด้านหน้า คือ score >= 80 นั้นเป็นจริง กรณีอื่น ๆ ที่มีลักษณะคล้ายกันก็จะถูกประมวลผลในแบบเดียวกันนี้ ซึ่งเป็นการประมวลผลที่ซ้ำซ้อน ทั้ง ๆ ที่ไม่มีความจำเป็นที่ต้องทำเลย แต่นี่เป็นเพียงตัวอย่างเดียวเท่านั้น มีกรณีอีกหลายกรณีที่ ต้องใช้ && หรือ || เข้ามาช่วยในการตรวจสอบจริง ๆ เช่นในกรณีของโปรแกรมต่อไปนี้ ซึ่งเป็นโปรแกรมตรวจสอบว่า ตัวอักษรที่กำหนดให้อยู่ในกลุ่มนั้น ๆ หรือไม่

```
/* UpperLower.java - Using && in if - else statement */

import java.io.*;

class UpperLower {
    public static void main(String[] args) throws IOException {
        char ch = 'F';

        if(ch >= 'A' && ch <= 'Z')
            System.out.println(ch + " is an uppercase letter.");
        else if(ch >= 'a' && ch <= 'z')
            System.out.println(ch + " is a lowercase letter.");
    }
}
```

โปรแกรมกำหนดให้ตัวแปร ch มีค่าเป็นอักษร F ตัวใหญ่ และใช้ประโยค if – else ทำการตรวจสอบว่าตัวอักษรที่ว่านี้เป็นอักษรตัวใหญ่หรือตัวเล็ก ซึ่งก็แน่นอนว่าผลลัพธ์ที่ได้จะต้องเป็นตัวใหญ่ แต่เราต้องการที่จะแสดงให้เห็นถึงการใช้ && เข้ามาช่วยในการตรวจสอบค่าของ ch ที่เราได้กำหนดขึ้น

ในการเปรียบเทียบนั้น Java จะใช้ค่า ASCII (Unicode) ในการเปรียบเทียบตัวอักษรทั้งสองตัว โดยในตอนแรกจะเปรียบเทียบ F กับ A (70 กับ 65) ซึ่งผลของการเปรียบเทียบเป็นจริง จึงต้องเปรียบเทียบ F กับ Z (70 กับ 90) ซึ่งก็เป็นจริง ดังนั้นประโยคที่ตามมา System.out.println(ch + " is an uppercase letter."); จึงได้รับการประมวลผล เช่นเดียวกันกับตัวอักษรตัวเล็ก ถ้า ch เก็บอักษรที่เป็นตัวเล็กอยู่การเปรียบเทียบจะเกิดขึ้นในประโยค else – if ที่ตามมา

โปรแกรมต่อไปนี้แสดงการใช้ || ในการตรวจสอบว่าเดือนที่กำหนดให้อยู่ในฤดูอะไร

```
/* Season.java - Using || in if - else statement */

import java.io.*;
import java.lang.Integer;

class Season {
    public static void main(String[] args) throws IOException {
        int month;
        String season, input;
        InputStreamReader in;
        BufferedReader buffer;

        System.out.print("Enter month (1 - 12): ");
        in = new InputStreamReader(System.in);
```

```
        buffer = new BufferedReader(in);
        input = buffer.readLine();

        month = Integer.parseInt(input);
        if(month == 3 || month == 4 || month == 5 || month == 6)
            season = "Summer";
        else if(month == 7 || month == 8 || month == 9 || month == 10)
            season = "Rainy";
        else if(month == 11 || month == 12 ||
                month == 1 || month == 2)
            season = "Winter";
        else
            season = "Never heard of it!";

        System.out.println(month + " is in " + season);
    }
}
```

โปรแกรม Season.java ใช้ || ในการตรวจสอบว่า month ที่ user ใส่เข้าสู่โปรแกรมนั้นอยู่ในฤดูอะไร โดยเรากำหนดให้เดือน 3 – 6 เป็นฤดูร้อน 7 – 10 เป็นฤดูฝน และ 11 – 2 เป็นฤดูหนาว (ซึ่งเดียวนี้ชักจะไม่แน่นอนเสียแล้ว) ลองมาดูผลลัพธ์จากการ run ดีกว่า

```
Enter month (1 - 12): 4
4 is in Summer

Enter month (1 - 12): 8
8 is in Rainy

Enter month (1 - 12): 1
1 is in Winter

Enter month (1 - 12): 20
20 is in Never heard of it!
```

Operator || นั้นเป็นการเปรียบเทียบข้อมูลที่อาจเป็นได้หลาย ๆ อย่าง ดังที่เห็นในโปรแกรม ถ้าเราให้ค่าของ month เป็น 3 4 5 หรือ 6 เราก็กำหนดให้ season มีค่าเป็น Summer ส่วนเดือนอื่น ๆ ก็เช่นเดียวกัน ในทางตรงกันข้ามถ้าเราใช้ && แทนโปรแกรมของเราก็จะไม่ทำงานตามที่เราคาดหวังไว้ กล่าวคือ ค่าของ month ไม่สามารถเป็นไปได้ทั้งสี่ค่า โปรแกรมก็จะประมวลผลประโยคที่ตาม else ตัวสุดท้ายตลอดไป ผู้อ่านควรตรวจสอบว่าที่กล่าวมานี้ เป็นจริงหรือไม่

ในการใช้ Nested if – else นั้นเราจะต้องระวังว่าทุก ๆ if ที่เราใช้จะต้องมี else หรือไม่เพราะ Java จะตรวจสอบว่า else ที่ใช้นั้นว่ามี if อยู่ก่อนหน้าหรือไม่ สมมติว่ามีเราก็ต้องระวังในเรื่องของขั้นตอนการตรวจสอบ ว่าเป็นไปอย่างที่เราตั้งใจหรือไม่ เช่นตัวอย่างนี้

```
/* DanglingElse.java */

import java.io.*;

class DanglingElse {
    public static void main(String[] args) {
        int x = 7, y = 99;

        if(x > 5)
            if(y > 5)
                System.out.println("x and y is greater than 5");
        else
            System.out.println("x is less than 5");
```

```
    }
}
```

โปรแกรม DanglingElse.java ที่เห็นนี้เมื่อ run แล้วผลลัพธ์ที่ได้คือ

```
x and y is greater than 5
```

ซึ่งเป็นไปตามที่เราคิดไว้ แต่ถ้าเราเปลี่ยนค่าของ y ให้เป็น 2 ผลลัพธ์ที่ได้กลับเป็น

```
x is less than 5
```

ซึ่งไม่ใช่สิ่งที่เราต้องการ เพราะเรารู้ว่า x มีค่ามากกว่า 5 สิ่งที่เกิดขึ้นคือ Java จัดให้ else นั้นคู่กับ if ตัวที่สอง ดังนี้

```
if(x > 5)
    if(y > 5)
        System.out.println("x and y is greater than 5");
    else
        System.out.println("x is less than 5");
```

เพราะฉะนั้นถ้าเราต้องการให้การประมวลผลเป็นไปตามที่เราต้องการ เราต้องใช้ {} เป็นตัวช่วย ดังนี้

```
if(x > 5) {
    if(y > 5)
        System.out.println("x and y is greater than 5"):
}
else
    System.out.println("x is greater than 5");
```

ซึ่งเมื่อ run ดูแล้วก็จะได้ผลลัพธ์เป็นไปตามที่เราต้องการ คือ ไม่มีผลลัพธ์ใด ๆ แสดงไปยังหน้าจอถ้า y มีน้อยกว่า หรือเท่ากับ 0

ถึงเวลาที่เราจะกลับไปปรับปรุงโปรแกรม Quadratic.java ที่เราได้พูดถึงในบทที่สอง เพื่อให้มีการตรวจสอบค่าของ determinant ที่อาจเป็น 0 ด้วยการใช้ if ในประโยค

```
//calculate determinant
double d = b * b - 4 * a * c;
if(d <= 0) {
    System.err.println("Error: cannot find square root");
    System.exit(1);
}
double det = Math.sqrt(d);
```

เราได้กำหนดให้มีการตรวจสอบค่าของ d ก่อนการเรียกใช้ Math.sqrt() ถ้าค่าของ d น้อยกว่า หรือเท่ากับ 0 เราจะฟ้องไปยัง user พร้อมกับยุติการทำงานของโปรแกรมทันที ด้วยการเรียกใช้ System.exit(1)

```
/* Quadratic.java - Calculating roots of function */

class Quadratic {
    public static void main(String[] args) {
        double a, b, c;
        //4x^2 + 20x + 16 = 0
        a = 4;
        b = 20;
        c = 16;
        //calculate determimant
        double d = b * b - 4 * a * c;
```

```
        if(d <= 0) {
            System.err.println("Error: cannot find square root");
            System.exit(1);
        }
        double det = Math.sqrt(d);
        //calculate roots
        double x1 = (-b + det) / (2 * a);
        double x2 = (-b - det) / (2 * a);

        System.out.println("Root 1 = " + x1);
        System.out.println("Root 2 = " + x2);
    }
}
```

การใช้ Conditional Operator (? :)

Java มี operator อีกตัวหนึ่งที่เราสามารถนำมาใช้แทน if – else ได้ เราเรียก operator ตัวนี้ว่า Conditional operator หรือ operator ที่มี expression (statement) อยู่สามตัว ซึ่งมีโครงสร้างดังนี้ คือ

(expression) ? (true statement) : (false statement);

เช่น

```
(number % 2 == 0) ? System.out.println("Even number.")
                  : System.out.println("Odd number.");
```

ลองดูโปรแกรมตัวอย่างกัน

```
/* OddEven.java */

import java.io.*;

class OddEven {
    public static void main(String[] args) {
        int num;
        String result;

        num = (int)(Math.random() * 100;
        result = (num % 2 == 0) ? "Even" : "Odd";
        System.out.println(num + " is an " + result + " number.");
    }
}
```

โปรแกรม OddEven.java ใช้ operator ? : ในการตรวจสอบว่า num เป็นเลขคู่หรือ เลขคี่ ซึ่งถ้าเป็นอย่างใดอย่างหนึ่งก็จะเก็บข้อมูลนั้น (Even หรือ Odd) ไว้ในตัวแปร result แล้วจึงแสดงผลไปยังหน้าจอหลังจากนั้น ดังนี้

```
65 is an Odd number.
```

โปรแกรม Min.java ที่เห็นด้านล่างนี้ใช้ operator ? : ในการหาค่าตัวเลขที่น้อยที่สุดระหว่างเลขสองตัวที่กำหนดให้

```
/* Min.java */

import java.io.*;

class Min {
```

```
    public static void main(String[] args) {
        int num1 = 45, num2 = 89;

        int min = (num1 < num2) ? num1 : num2;
        System.out.println("Min number is " + min);
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
Min number is 45
```

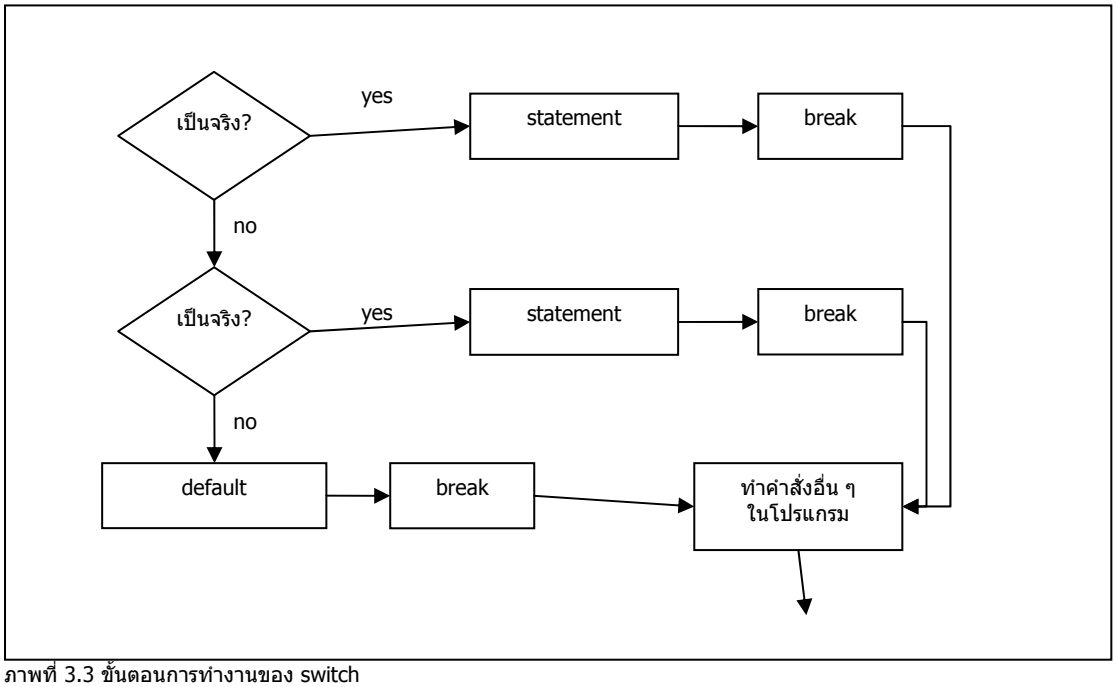
การใช้ operator ? : ก็คือการใช้ if – else นั่นเองดังนั้นก็เป็นเรื่องของผู้เขียนโปรแกรมเองว่าอยากที่จะใช้ operator ตัวไหน เพราะทั้งสองให้ผลการประมวลผลในลักษณะเดียวกัน

การใช้ switch

Java มีคำสั่งอีกตัวหนึ่งที่เราสามารถนำมาใช้แทน if – else ได้ คำสั่งนี้ก็คือ switch ซึ่งมีโครงสร้างดังนี้

```
switch(expression) {
    case value1 :
        statement;
        break;
    case value2 :
        statement;
        break;
    …

    default :
        statement;
        break;
}
```

ค่าที่ได้จาก expression ของ switch นั้นจะต้องเป็นค่าที่มีชนิดเป็น char byte short หรือ int เท่านั้น ถ้าเป็นค่าชนิดอื่น Java จะฟ้องด้วย error ทันที ภาพที่ 3.3 แสดงถึงขั้นตอนการทำงานของ switch



ภาพที่ 3.3 ขั้นตอนการทำงานของ switch

คำสั่ง break ใน switch นั้นสำคัญมาก ๆ ถ้าเราไม่ใส่ การทำงานของ switch ก็จะไม่ได้ผลลัพธ์ดังที่เราได้คาดการณ์ไว้ เราลองมาดูตัวอย่างโปรแกรมการใช้ switch ดู

```
/* Switch.java - Using switch
 * to differentiate input
*/

import java.io.*;
import java.lang.Integer;

class Switch {
    public static void main(String[] args) throws IOException {
        int month;
        String season, input;
        InputStreamReader in;
        BufferedReader buffer;

        System.out.print("Enter month (1 - 12): ");
        in = new InputStreamReader(System.in);
        buffer = new BufferedReader(in);
        input = buffer.readLine();

        month = Integer.parseInt(input);
        switch(month) {
            case 3: case 4: case 5: case 6:
                season = "Summer";
                break;
            case 7: case 8: case 9: case 10:
                season = "Rainy";
                break;
            case 11: case 12: case 1: case 2:
                season = "Winter";
                break;
            default :
                season = "Never heard of it!";
```

```
                break;
        }
        System.out.println(month + " is in " + season);
    }
}
```

เราได้ทำการเปลี่ยนแปลงโปรแกรม Season.java ที่ใช้ if – else ให้มาใช้ switch แทน โดยเราได้รวมเอาเดือนที่อยู่ในฤดูเดียวกันเหมือนที่เราทำกับโปรแกรม Season.java มาอยู่ในกลุ่มของ case เดียวกัน จะเห็นว่าเราเขียน case ต่าง ๆ ที่อยู่ในกลุ่มเดียวกันตามหลังกันไปโดยไม่มีการประมวลผลใด ๆ จนกว่าจะจบ case สุดท้ายในกลุ่มนั้น เช่น

```
case 3: case 4: case 5: case 6:
    season = "Summer";
    break;
```

Java จะประมวลผล case 3 ก่อนแต่เราไม่มีประโยคใด ๆ ใน case นี้ดังนั้น Java จึงประมวลผล case 4 case 5 และจนกระทั่งมาถึง case 6 ซึ่งเป็น case สุดท้ายในกลุ่ม Java จึงประมวลผลประโยค season = "Summer" และ break

การนำเอา case ต่าง ๆ มาต่อกันในลักษณะนี้มีผลเหมือนการประมวลผลด้วยการใช้ operator || ในประโยคที่เรียกใช้ if ผู้อ่านควรสังเกตถึงการใช้ switch ในลักษณะนี้ให้ดี เพราะเป็นวิธีการที่ช่วยลดขั้นตอนการเขียนได้ดีพอสมควร

ผลลัพธ์ที่ได้จากการ run โปรแกรม Switch.java ก็เหมือนกันกับการ run โปรแกรม Season.java

เรามาลองดูตัวอย่างการใช้ switch อีกสักตัวหนึ่ง

```
/* Switch2.java */

import java.io.*;
import java.lang.Integer;

class Switch2 {
    public static void main(String[] args) throws IOException {
        int digit;
        BufferedReader buffer;
        InputStreamReader isr;
        String input;

        System.out.print("Enter a digit: ");
        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
        input = buffer.readLine();

        digit = Integer.parseInt(input);
        switch(digit) {
            case 1: System.out.println("You entered ONE.");
                    break;
            case 2: System.out.println("You entered TWO.");
                    break;
            case 3: System.out.println("You entered THREE.");
                    break;
            case 4: System.out.println("You entered FOUR.");
                    break;
            case 5: System.out.println("You entered FIVE.");
                    break;
            case 6: System.out.println("You entered SIX.");
```

```
                break;
            case 7: System.out.println("You entered SEVEN.");
                break;
            case 8: System.out.println("You entered EIGHT.");
                break;
            case 9: System.out.println("You entered NINE.");
                break;
            case 0: System.out.println("You entered ZERO.");
                break;
            default: System.out.println("Sorry - not a digit!");
                 break;
        }
    }
}
```

โปรแกรม Switch2.java บอก User ให้ใส่ตัวเลขระหว่าง 0 – 9 และจะแสดงตัวหนังสือแทนตัวเลขเหล่านั้นกลับไปยังหน้าจอ ซึ่งเมื่อ run แล้วก็จะได้ผลลัพธ์ดังนี้

```
Enter a digit: 9
You entered NINE.

Enter a digit: 2
You entered TWO.

Enter a digit: 36
Sorry - not a digit!
```

เมื่อเราใส่ 36 โปรแกรมกลับบอกว่าไม่ใช่ตัวเลข ทั้งนี้เพราะโปรแกรมของเราตรวจสอบเฉพาะตัวเลขเพียงตัวเดียวเท่านั้น เราต้องปรับปรุงโปรแกรมของเราใหม่ถ้าเราต้องการที่จะตรวจสอบตัวเลขจริง ๆ ทั้งหมด

การทำงานแบบวน (Loop)

หลาย ๆ ครั้งที่โปรแกรมเมอร์ต้องการที่จะให้โปรแกรมทำการประมวลผลในรูปแบบเดิมซ้ำ ๆ กัน เช่น คูณตัวเลข 1 – 10 ด้วย 2 หรือ อ่านข้อมูลจาก keyboard จนกว่า user จะพอใจ เราสามารถนำเอาชุดคำสั่งที่ Java มีให้มาใช้กับการทำงานที่ว่านี้ได้อย่างไม่ยากนัก ลองดูโปรแกรมที่ทำการแสดงค่าจาก 1 – 10 ไปยังหน้าจอนี้

```
/* ForLoop.java */

import java.io.*;

class ForLoop {
    public static void main(String[] args) {
        int number = 1;

        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
```

```
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
        number++;
        System.out.println("Number now is " + number);
    }
}
```

แน่นอนผลลัพธ์ที่ได้ก็ต้องเป็น

```
Number now is 1
Number now is 2
Number now is 3
Number now is 4
Number now is 5
Number now is 6
Number now is 7
Number now is 8
Number now is 9
Number now is 10
```

การแสดงค่าจาก 1 – 10 นั้นยังไม่มีปัญหาอะไรมากมายนัก ถ้าเราต้องการที่จะแสดงค่าจาก 1 – 100 หรือจาก 1 – 1000 หละ เราคงต้องพิมพ์จนมือเมื่อยแน่เลย ลองมาดูว่าเราจะใช้ loop ช่วยในการทำงานในลักษณะนี้ได้อย่างไร

Java มีโครงสร้างหลายตัวที่เราสามารถนำมาใช้ในการทำงานแบบวนได้ คำสั่งที่ว่านี้ คือ while, do..while และ for ชุดคำสั่งแรกที่เราจะดูกัน คือ for loop

การใช้ for loop

ก่อนที่เราจะไปดูโครงสร้างของ for loop กันเรามาดูกันว่าโปรแกรมก่อนหน้านี้ ถ้าใช้ for loop แล้วจะมีหน้าตาเป็นอย่างไร

```
/* ForLoop2.java */

import java.io.*;

class ForLoop2 {
    public static void main(String[] args) {
        int number;

        for(number = 1; number <= 10; number++)
            System.out.println("Number now is " + number);
    }
}
```

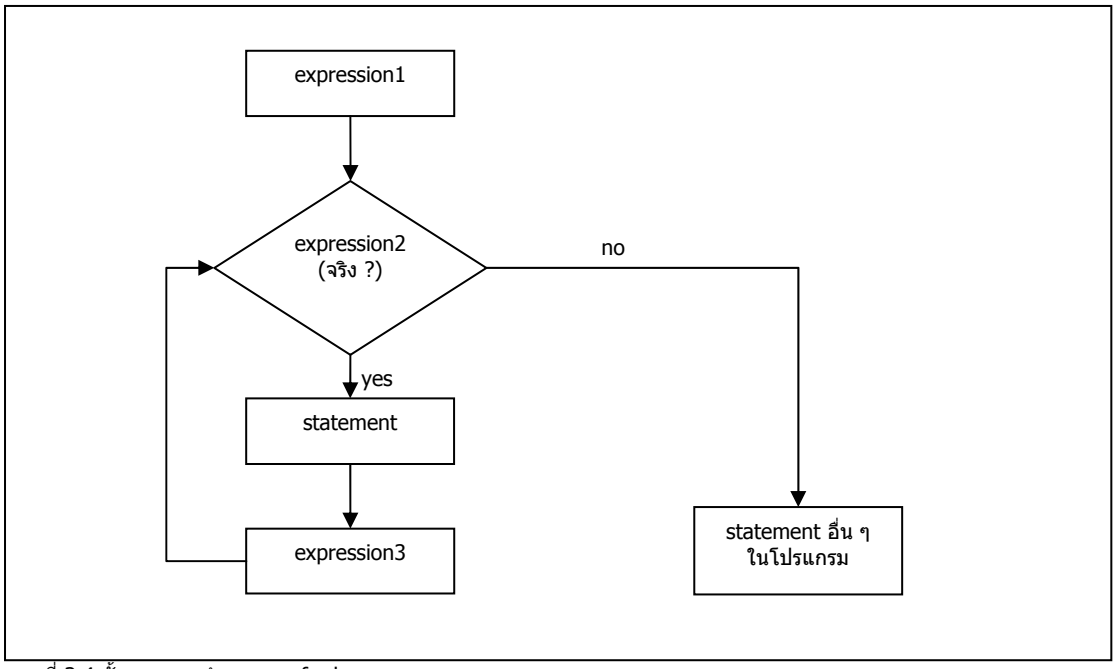
จะเห็นว่าโปรแกรมของเราสั้นลงมาก คำสั่งใน for loop บอกให้ Java รู้ว่าการประมวลผลต้องทำทั้งหมด 10 ครั้งโดยเริ่มต้นการประมวลผล เมื่อ number มีค่าเท่ากับ 1 ไปจนกว่า number จะมีค่าเป็น 10 ซึ่งเมื่อ number มีค่าเป็น 10 Java ก็จะยุติการประมวลผลทั้งหมด และผลลัพธ์ที่ได้ก็เหมือนกันกับโปรแกรม ForLoop.java ที่เราเขียนขึ้นมาก่อนหน้านี้

โครงสร้างของ for loop มีดังนี้คือ

```
for(expression1 ; expression2 ; expression3)
    statement;
```

โดยที่ expression ทั้งสามตัวมีการทำงานที่ต่างกัน expression1 จะถูกประมวลผลเป็นตัวแรกสุด ซึ่งโดยทั่วไป expression นี้มักจะถูกเรียกว่า initialization expression คือทำหน้าที่ในการกำหนดค่าเบื้องต้นให้กับตัวแปรที่จะนำมาใช้ใน for loop หลังจากนั้น expression2 ก็จะถูกประมวลผล ซึ่งถ้าการประมวลผลเป็นจริงตามเงื่อนไข statement ที่อยู่ใน for loop ก็จะถูกประมวลผล และ for loop จะยุติการทำงานทันทีที่การประมวลของ expression2 เป็นเท็จ เมื่อ statement ที่อยู่ใน for loop สิ้นสุดการประมวลผลแล้ว expression3 ก็จะได้รับการประมวลผล และ for loop จะกลับไปประมวลผล expression2 อีกครั้ง ซึ่งถ้าเป็นจริง statement ก็จะถูกประมวลผลอีกครั้ง ทำอย่างนี้ไปจนกว่า expression2 จะเป็นเท็จ เราเรียก expression2 ว่า loop condition และ expression3 ว่า increment expression

ภาพที่ 3.4 แสดงแผนผังการทำงานของ for loop



ภาพที่ 3.4 ขั้นตอนการทำงานของ for loop

เรากลับมาดูโปรแกรม ForLoop2.java ที่เราเขียนก่อนหน้านี้ว่ามีการทำงานอย่างไร เราเริ่มต้นด้วยการเขียน for loop

```
for(number = 1; number <= 10; number++)
    System.out.println("Number now is " + number);
```

สิ่งที่ Java ทำก่อนก็คือ กำหนดค่าตัวแปร number ให้เป็นหนึ่ง หลังจากนั้นก็ทำการตรวจสอบว่าตัวแปร number มีค่าน้อยกว่า หรือเท่ากับสิบหรือไม่ ซึ่งก็เป็นจริง ดังนั้น Java จึงประมวลผลประโยคที่อยู่ใน for loop เมื่อเสร็จแล้วจึงไปประมวลผลประโยค number++ ซึ่งเป็นการเพิ่มค่าให้กับตัวแปร number เสร็จแล้วจึงกลับไปตรวจสอบว่า number <= 10 อีกครั้ง ทำไปเรื่อย ๆ จนกว่า number จะมีค่ามากกว่าสิบจึงจะยุติการทำงาน โดยสรุปแล้วขั้นตอนของ for loop มีดังนี้คือ

1. กำหนดค่าเบื้องต้นให้กับ number
2. ตรวจสอบว่า number <= 10 หรือไม่
   ถ้าเป็นจริง
   ประมวลผลประโยค System.out.println("Number now is " + number);
   ถ้าเป็นเท็จ ยุติการทำงานของ for loop
3. ประมวลผลประโยค number++
4. กลับไปทำคำสั่งในข้อสองใหม่

Expression ที่สำคัญที่สุดที่จะทำให้ for loop ทำงานต่อ หรือยุติการทำงานก็คือ expression ที่สอง ดังนั้นในการใช้ for loop เราจึงต้องระมัดระวังถึงประโยคที่ใช้ในการตรวจสอบว่าสามารถที่จะทำได้ทั้งสองกรณีหรือไม่ เพราะถ้าเราตรวจสอบไม่ดีการทำงานของ for loop อาจทำไปโดยไม่มีการสิ้นสุด หรือที่เรียกว่า Infinite loop ตัวอย่างเช่น ถ้าเราเปลี่ยน for loop ของเราให้เป็นดังนี้

```
for(number = 1; number > 0; number++)
    System.out.println("Number now is " + number);
```

For loop ใหม่ของเราก็จะทำงานโดยไม่มีที่สิ้นสุด เพราะว่าค่าของ number จะมากกว่าศูนย์อยู่ตลอดเวลา ทำให้ประโยคของการตรวจสอบเป็นจริงเสมอ

การเขียน for loop ให้ทำงานตลอดเวลานั้นสามารถเขียนได้หลายแบบ ตัวอย่างเช่น

```
int number = 1;
for(;;)
    System.out.println("Number now is " + number);
```

เราให้ for loop ทำงานไปเรื่อย ๆ ด้วยการยกเลิก expression ทั้งสามตัวที่อยู่ใน loop หรือยกเลิก expression ที่หนึ่งและสาม ดังนี้

```
int number = 1;
for(;number > 0;)
    System.out.println("Number now is " + number);
```

ถ้าเราต้องการ infinite loop ที่ไม่มีประโยคใด ๆ เลยก็ทำได้ ดังนี้

```
for( ; ;) ;
```

เราลองมาดูการใช้ for loop ในการบวกเลขจาก 1 ถึง 100

```
/* Add1To100.java */

import java.io.*;

class Add1To100 {
    public static void main(String[] args) {
        int total = 0;

        for(int i = 1; i <= 100; i++)
            total += i;
        System.out.println("Sum of 1 - 100 is " + total);
    }
}
```

ผลลัพธ์ของการ run คือ

```
Sum of 1 - 100 is 5050
```

ตัวอย่างโปรแกรมต่อไปนี้ทำการคำนวณหาดอกเบี้ยทบต้นในแต่ละปี เป็นจำนวนเท่ากับจำนวนปีที่ user เป็นผู้กำหนดจากเงินฝากเริ่มต้นที่หนึ่งพันบาท (หรือเท่ากับจำนวนที่ user เป็นผู้ใส่เข้าสู่โปรแกรม) โดยใช้สูตรการในการคำนวณ คือ

$amount = p(1 + r)^n$

โดยที่

p คือ จำนวนเงินเริ่มต้น (principal)
r คือ อัตราดอกเบี้ยประจำปี
n คือ จำนวนของปีที่ต้องการหา
amount คือ เงินต้นที่นำฝากของทุกปี

```
/* Interests.java */

import java.io.*;
import java.lang.Integer;

class Interests {
    public static void main(String[] args) throws IOException {
        double amount;    //amount at the end of year
        double principal; //first deposit
        double rate;      //annual interests rate
        int years;        //number of years
        BufferedReader buffer;
        InputStreamReader isr;
        String input;

        //get input from user (principal, rate, years)
        System.out.print("Enter amount to deposit: ");
        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
        input = buffer.readLine();
        principal = Double.parseDouble(input);

        System.out.print("Enter interests rate: ");
        input = buffer.readLine();
        rate = Double.parseDouble(input);

        System.out.print("Enter number of year: ");
        input = buffer.readLine();
        years = Integer.parseInt(input);

        //calculate and display amount
        System.out.println("\nYear\tAmount on deposit");
        for(int y = 1 y <= years; y++) {
            amount = principal * Math.pow(1.0 + rate, y);
            System.out.println(y + "\t" + amount);
        }
        System.out.print("\nTotal interests after " + years);
        System.out.println(" years is " + (amount - principal));
    }
}
```

ผลลัพธ์ของการ run ด้วย principal = 10000 rate = 0.25 และ years = 5

```
Enter amount to deposit: 10000
Enter interests rate: .25
Enter number of year: 5

Year   Amount on deposit
1       12500.0
2       15625.0
3       19531.25
4       24414.0625
5       30517.578125
```

```
Total interests after 5 years is 20517.578125
```

ส่วนประกอบที่สำคัญคือ for loop ที่ทำหน้าที่คำนวณหาจำนวนเงินที่ได้รับในแต่ละปี (พร้อมทั้งดอกเบี้ย) เราใช้ method Math.pow() ในการหาค่าของดอกเบี้ยในแต่ละปี ซึ่งเมื่อหาได้แล้วเราก็แสดงผลลัพธ์ไปยังหน้าจอทันที for loop ของเรายุติการประมวลผลเมื่อค่าของตัวแปร y มีค่ามากกว่าตัวแปร years หลังจากแสดงจำนวนของเงินในแต่ละปีแล้ว เมื่อออกจาก for loop เราก็แสดงจำนวนของดอกเบี้ยทั้งหมดที่ได้จากการฝากในช่วงนั้น

ถ้าหากเราไม่ต้องการใช้สูตรในการคำนวณหาดอกเบี้ย เราก็สามารถทำได้ด้วยการคำนวณหาดอกเบี้ยก่อน แล้วจึงนำเอาดอกเบี้ยที่หาได้ไปบวกกับเงินต้น หลังจากนั้นก็คำนวณหาดอกเบี้ยจากเงินต้นใหม่นี้ ทำการคำนวณจนกว่าจะครบตามจำนวนปีที่ได้กำหนดไว้ ดังที่แสดงด้วย code นี้

```
amount = principal;
System.out.println("\nYear\tAmount on deposit");
for(int y = 1; y <= years; y++) {
    double interests = amount * rate;   //calculate interests
    amount += interests;         //calculate new principal
    System.out.println(y + "\t" + amount);
}
```

เราลองมาดูโปรแกรมตัวอย่างอีกตัวหนึ่งที่ใช้ for loop ในการควบคุมการทำงานของโปรแกรม พร้อมทั้งยอมให้ user ใส่ข้อมูลหลาย ๆ ตัวในบรรทัดเดียวกันได้

```
/* AddNumbers.java - reading numbers from input line
 * user must specify space between those numbers and
 * hit <enter> when done.
*/

import java.io.*;
import java.text.DecimalFormat;

class AddNumbers {
    public static void main(String[] args) throws IOException {
        double number, sum = 0;  //a number read and sum of numbers
        int count = 0;           //total numbers read
        BufferedReader buffer;   //input buffer
        InputStreamReader isr;   //input stream
        StreamTokenizer token;   //token from input stream
        DecimalFormat f = new DecimalFormat("0.00");

        System.out.print("Enter numbers (space between -
                          enter when done): ");
        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
        //get tokens from input buffer
        token = new StreamTokenizer(buffer);
        //set priority to EOL so that we can break out of the loop
        token.eolIsSignificant(true);
        //get each number from tokens
        //calculate the sum until EOL is reached
        for(; token.nextToken() != token.TT_EOL ;) {
            switch(token.ttype) {
                //token is a number
                case StreamTokenizer.TT_NUMBER:
                    number = token.nval;  //save token in number
                    count++;              //number of tokens read
```

```
                    sum += number;        //calculate sum
                    break;
                default:
                    System.out.println("Token is not a number! " +
                                                token.toString());
                    break;
            }
        }
        //calculate average and display result
        double average = sum / count;
        System.out.println("Sum of numbers = " + sum);
        System.out.println("Average of numbers = " +
                            f.format(average));
    }
}
```

โปรแกรม AddNumbers.java ใช้ method จาก class StreamTokenizer และ for loop เป็นตัวช่วยในการดึงเอาข้อมูลแต่ละตัวที่ User ใส่เข้าสู่โปรแกรม เรากำหนดให้ user กดปุ่ม <enter> เมื่อสิ้นสุดการใส่ข้อมูล ดังนั้นเราจึงจำเป็นที่จะต้องบอกให้ Java รู้ว่าปุ่ม <enter> มีความสำคัญในการยุติการทำงานของ loop ซึ่งเราทำได้ด้วยการเรียกใช้ method eolIsSignificant() ด้วยค่าที่เป็นจริงดังนี้

```
eolIsSignificant(true);
```

ถ้าเราไม่เรียก method นี้มาใช้การกดปุ่ม <enter> ของ user ก็จะไม่มีผลใด ๆ ต่อโปรแกรม (โปรแกรมจะไม่ยุติการทำงาน และเราต้องกดปุ่ม <ctrl> + c เพื่อให้โปรแกรมหยุดการทำงาน) เมื่อเรากำหนดความสำคัญของปุ่ม <enter> แล้วเราก็เริ่มอ่าน token แต่ละตัวที่เราได้ดึงมาจาก keyboard จนกว่า token นั้นจะมีค่าเป็น EOL (End Of Line) ซึ่งเป็นตัวกำหนดการสิ้นสุดของข้อมูลในบรรทัดนั้น

เราอ่าน token ที่มีอยู่ทั้งหมดใน buffer มาเก็บไว้ในตัวแปร token ด้วยคำสั่ง

```
token = new StreamTokenizer(buffer);
```

หลังจากนั้นเราก็ตรวจสอบดูว่า token ที่อยู่ถัดไปใน buffer เป็น EOL หรือไม่ ซึ่งถ้าไม่เป็นเราก็เรียกใช้ switch เพื่อตรวจสอบดูว่า token ที่อ่านมานั้นเป็นตัวเลขหรือไม่ ถ้าเป็นเราก็จะทำการบวกตัวเลขนี้เข้ากับตัวแปร sum พร้อมกับเพิ่มค่าให้กับตัวแปร count

```
for(; token.nextToken() != token.TT_EOL ;) {
    switch(token.ttype) {
        //token is a number
        case StreamTokenizer.TT_NUMBER:
            number = token.nval;  //save token in number
            count++;              //number of tokens read
            sum += number;        //calculate sum
            break;
        default:
            System.out.println("Token is not a number! " +
                                token.toString());
            break;
    }
}
```

เราตรวจสอบความเป็นตัวเลขของ token ได้ด้วยการเปรียบเทียบ token.ttype กับ StreamTokenizer.TT_NUMBER และถ้าเป็นจริง token ที่เป็นตัวเลขจะถูกเก็บไว้ในตัวแปร nval (และมีชนิดเป็น double) ดังนั้นตัวแปร number จึงต้องประกาศให้เป็น double และการดึงค่ามาเก็บในตัวแปร number ทำได้ด้วยการกำหนดค่าจากประโยค

```
number = token.nval;
```

เนื่องจากว่าเราสนใจเฉพาะ token ที่เป็นตัวเลขเท่านั้นเราจึงไม่สนใจ token ที่เป็นชนิดอื่น ดังนั้นเราจึงเพียงแต่ส่งค่าของ token กลับคืนไปยังหน้าจอถ้า token นั้นเป็นชนิดอื่นที่ไม่ใช่ตัวเลข

for loop ของเราจะทำงานไปเรื่อย ๆ จนกว่า user จะกดปุ่ม <enter> ซึ่งการกดปุ่ม <enter> นี้จะทำให้ expression ที่มีอยู่เพียงตัวเดียวใน for loop เป็นเท็จ การทำงานของ for loop จึงสิ้นสุดลง ผู้อ่านจะสังเกตได้ว่าเราไม่สนใจกับ expression ที่ 1 และ expression ที่ 3 ของ for loop เลยทั้งนี้ก็เพราะว่าเราไม่ต้องมีการกำหนดค่าเบื้องต้น หรือเพิ่มค่าใด ๆ เลย การยุติการทำงานของ for loop เกิดจากการกดปุ่ม <enter> ของ user เท่านั้น

ผลลัพธ์ของการ run ด้วยค่าสมมติต่าง ๆ

```
Enter numbers (space between - enter when done): 20 30 45 789 12 34
Sum of numbers = 930.0
Average of numbers = 155.0

Enter numbers (space between - enter when done): 45 68 95
Sum of numbers = 208.0
Average of numbers = 69.33
```

Class StreamTokenizer ยังมี field อื่น ๆ ที่เราสามารถเรียกใช้ได้ แต่ไม่ได้กล่าวไว้ในที่นี้เพราะว่าโปรแกรม AddNumbers.java ของเราสนใจแต่เฉพาะการอ่านตัวเลขเท่านั้น ผู้อ่านสามารถหาข้อมูลได้จาก javadoc ของ Sun เอง

ในการใช้ for loop นั้นเราสามารถที่จะใช้ , (comma) เป็นตัวแบ่งการประมวลในแต่ละ expression ได้ ดังตัวอย่างโปรแกรมการหาผลรวมและค่าเฉลี่ยนของเลขจากหนึ่งถึงสิบนี้

```
/* Sum.java */
import java.io.*;

class Sum {
    public static void main(String[] args) {
        int i, sum;

        for(i = 1, sum = 0; i <= 10; sum += i, i++) ;

        System.out.println("Sum = " + sum);
        System.out.println("Average = " + (double)sum / (i-1));
    }
}
```

เรากำหนดให้ expression1 ของเรามีข้อความอยู่สองข้อความคือ i = 0 และ sum = 0 โดยเราใช้ , เป็นตัวแบ่งข้อความทั้งสอง เช่นเดียวกันกับ expression3 เรากำหนดให้มีข้อความ sum += i และ i++ อยู่ และในตัว for loop นั้นเราไม่มีการกำหนดประโยคใด ๆ เลย ทั้งนี้ก็เพราะว่าเราได้กำหนดให้การหาผลรวมนั้นอยู่ใน expression3 แล้ว แต่เราจะต้องใส่ ; (หรือเราอาจใส่ {})เพื่อให้ Java รู้ว่าเราได้จบ for loop ของเราแล้ว

```
for(i = 1, sum = 0; i <= 10; sum += i, i++) ;
```

หรือ

```
for(i = 1, sum = 0; i <= 10; sum += i, i++) { }
```

การทำงานของ for loop ที่ว่านี้ก็เหมือนกันกับการเขียน for loop แบบเดิม ต่างกันตรงที่เราได้ย้ายตำแหน่งของประโยคสองประโยค เท่านั้นเอง

```
sum = 0;
for(i = 1; i <= 10; i++)
    sum += i;
```

ตำแหน่งของข้อความการหาผลรวมก็มีความสำคัญ เราต้องกำหนดให้ประโยค sum += i นั้นอยู่ก่อนประโยค i++ ทั้งนี้ก็เพราะว่า Java จะประมวลผลประโยคที่อยู่ทางซ้ายของเครื่องหมาย , ก่อน แล้วจึงจะประมวลผลประโยคที่อยู่ถัดไปทางขวา ซึ่งถ้าเรากำหนดให้ i++ อยู่ก่อน sum += i การประมวลผลของเราก็จะผิดพลาด ค่าของผลรวมที่ได้คลาดเคลื่อน

ส่วนที่สำคัญอีกส่วนหนึ่งก็คือ ค่าของ i สำหรับการคำนวณหาค่าเฉลี่ย เราต้องหักค่าของ i ออกหนึ่งค่าก่อนนำไปหาร sum ก็เนื่องจากว่าในขณะที่ for loop ยุติการทำงานนั้นค่าของ i มีค่าเป็น 11

การหาผลรวมของ for loop ที่กล่าวมาแล้วนั้น เราสามารถที่จะเขียนได้อีกหลายรูปแบบ เช่น

```
i = ;
sum = 0;
for( ; i <= 10; i++) ;
    sum += i;
```

หรือ

```
i = 1;
sum = 0;
for( ; i <= 10; ) {
    sum += i;
    i++;
}
```

หรือ

```
sum = 0;
for( i = 1; i <= 10; ) {
    sum += i;
    i++;
}
```

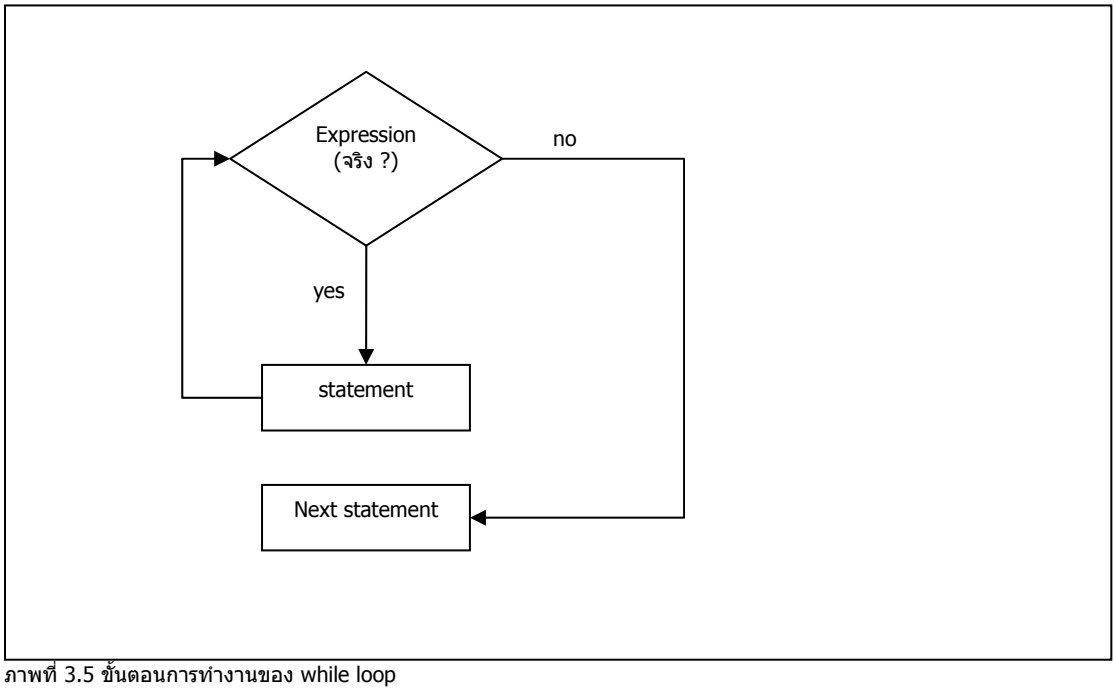
ผู้อ่านควรคำนึงถึงการเขียน code ที่ดูแล้วอ่านได้ง่าย มีโครงสร้างที่มองดูเป็นสากล ดังนั้นถึงแม้ว่า Java จะยอมให้เราเขียน code ได้อย่างอิสระและหลายรูปแบบ เราก็ไม่ควรเขียน code ที่ยากต่อการทำความเข้าใจ ยกเว้นว่าการเขียน code นั้น ๆ มีความจำเป็นที่ต้องใช้รูปแบบนั้น ๆ ในการเขียน (จำเป็นจริงหรือ?)

การใช้ while loop

การใช้ while loop ก็คล้าย ๆ กับการใช้ for loop ต่างกันเฉพาะโครงสร้างบางโครงสร้างเท่านั้นเอง (ตำแหน่งของ expression) เรามาดูโครงสร้างของ while loop กัน

```
while(expression)
    statement;
```

while loop จะตรวจสอบว่าค่าของ expression นั้นเป็นจริงหรือเท็จ ถ้าเป็นจริงก็จะประมวลผลประโยคต่าง ๆ ที่อยู่ภายใน while loop และจะประมวลผลประโยคอื่น ๆ ที่ตามมา (ภายนอก while loop) หลังจากที่หลุดออกจาก while loop แล้ว ภาพที่ 3.5 แสดงขั้นตอนการทำงานของ while loop



ภาพที่ 3.5 ขั้นตอนการทำงานของ while loop

เรามาลองใช้ while loop แทน for loop ที่เราได้ใช้ในโปรแกรมตัวอย่างบางตัวก่อนหน้านี้

```
/* Sum.java */

import java.io.*;

class Sum {
    public static void main(String[] args) {
        int i, sum;

        i = 0;
        sum = 0;
        while(i <= 10) {
            sum += i;
            i++;
        }

        System.out.println("Sum = " + sum);
        System.out.println("Average = " + (double)sum / (i-1));
    }
}
```

สิ่งแรกที่เราต้องทำก็คือการกำหนดค่าเบื้องต้นให้กับตัวแปร i และ sum หลังจากนั้นเราก็ทำการตรวจสอบดูว่าประโยค i <= 10 เป็นจริงหรือไม่ ซึ่งถ้าเป็นจริงเราก็ให้ while loop ของเราทำการบวกค่าของ i เข้ากับตัวแปร sum พร้อมทั้งเพิ่มค่าให้กับ i เมื่อเสร็จแล้ว ถ้าเราลองเปรียบเทียบการใช้ while loop กับ for loop เราจะเห็นว่า ประโยค

```
i = 0;
sum = 0;
```

คือ expression ที่หนึ่งของ for loop และประโยค

i <= 10 ก็คือ expression ที่สองใน for loop ส่วนประโยค i++ ก็คือ expression ที่สามใน for loop นั่นเอง (ถึงแม้ว่าเราได้กำหนดค่าของ i ให้เริ่มต้นที่ 0 ก็ไม่ทำให้การประมวลผลเปลี่ยนไป เพียงแต่ว่า loop ของเราทำงานมากขึ้นอีกหนึ่งครั้ง เท่านั้น)

ทีนี้เรามาลองใช้ while loop แทน for loop ในโปรแกรม AddNumbers.java กัน

```
/* AddNumbers2.java - reading numbers from input line
   user must specify space between those numbers and
   hit <enter> when done. (using while loop)
 */

import java.io.*;
import java.text.DecimalFormat;

class AddNumbers2 {
    public static void main(String[] args) throws IOException {
        double number, sum = 0;  //a number read and sum of numbers
        int count = 0;           //total numbers read
        BufferedReader buffer;   //input buffer
        InputStreamReader isr;   //input stream
        StreamTokenizer token;   //token from input stream
        DecimalFormat f = new DecimalFormat("0.00");

        System.out.print("Enter numbers (space between -
                          enter when done): ");
        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
        //get tokens from input buffer
        token = new StreamTokenizer(buffer);
        //set priority to EOL so that we can break out of the loop
        token.eolIsSignificant(true);
        //get each number from tokens
        //calculate the sum until EOL is reached
        while(token.nextToken() != token.TT_EOL) {
            switch(token.ttype) {
                //token is a number
                case StreamTokenizer.TT_NUMBER:
                    number = token.nval;  //save token in number
                    count++;              //number of tokens read
                    sum += number;        //calculate sum
                    break;
                default:
                    System.out.println("Token is not a number! " +
                                       token.toString());
                    break;
            }
        }
        //calculate average and display result
        double average = sum / count;
        System.out.println("Sum of numbers = " + sum);
        System.out.println("Average of numbers = " +
                           f.format(average));
    }
}
```

เราแทบที่จะไม่ต้องเปลี่ยนอะไรเลย เพียงแต่เปลี่ยนประโยค

```
for(; token.nextToken() != token.TT_EOL ;)
```

ให้เป็น

```
while(token.nextToken() != token.TT_EOL)
```

เท่านั้นเอง ส่วน code อื่น ๆ ก็ยังคงเหมือนเดิม

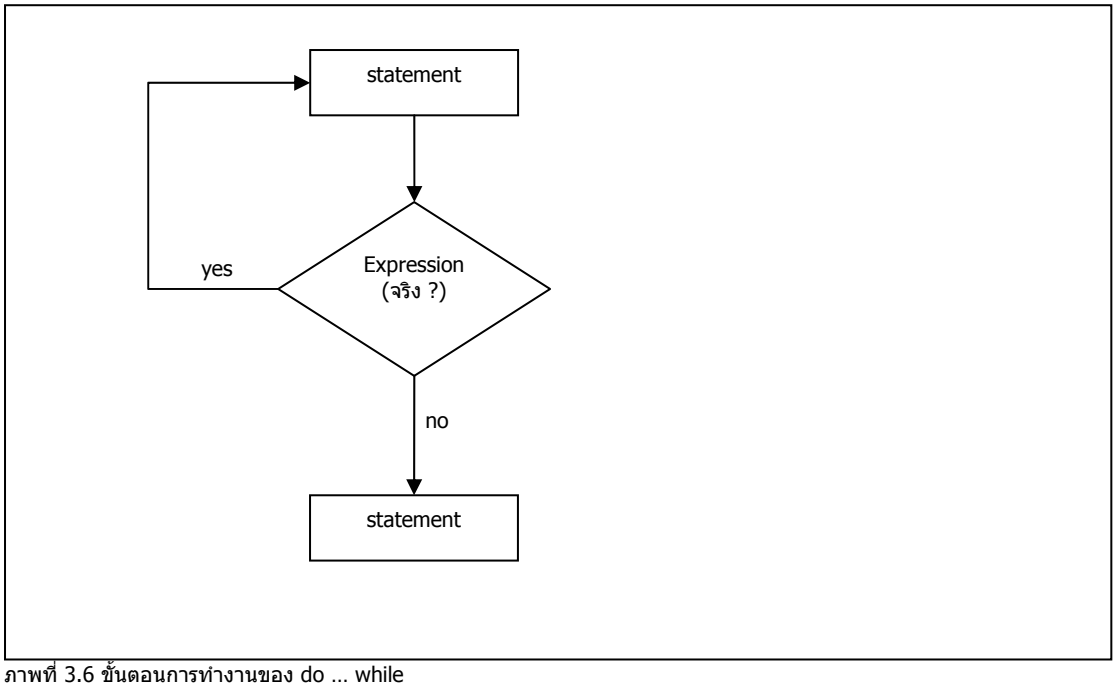
ทั้ง for loop และ while loop เป็นโครงสร้างของการทำงานแบบวนที่ต้องมีการตรวจสอบก่อนว่าประโยคในการควบคุมของ loop ทั้งสองเป็นจริงหรือเท็จ ซึ่งหมายถึงว่าการทำงานของ loop ทั้งสองอาจไม่เกิดขึ้นเลยถ้าการประมวลผลประโยคควบคุมเป็นเท็จ แต่ถ้าเราต้องการที่จะประมวลผลก่อนอย่างน้อยหนึ่งครั้ง เราจะทำอย่างไร

การใช้ do..while

Java มีโครงสร้างการใช้ loop อีกตัวหนึ่งที่ยอมให้มีการทำงานอย่างน้อยก่อนหนึ่งครั้ง แล้วจึงจะทำการประมวลผลประโยคควบคุม นั่นก็คือ do ... while ซึ่งมีโครงสร้างดังนี้

```
do {
     statement
}
while(expression);
```

ภาพที่ 3.6 แสดงขั้นตอนการทำงานของ do ... while loop



ภาพที่ 3.6 ขั้นตอนการทำงานของ do ... while

ถ้าเราต้องการที่จะทำงานอย่างน้อยหนึ่งครั้งก่อนที่จะมีการตรวจสอบเพื่อยุติการทำงาน หรือทำงานต่อไป do ... while loop จะเป็นโครงสร้างที่เหมาะสมที่สุด โปรแกรมตัวอย่างต่อไปนี้แสดงถึงการใช้ do ... while loop ในการตรวจสอบนั้น ๆ

```
/* DoWhile.java - do .. while to repeat the program
   until user enter a letter other than 'y'
 */

import java.io.*;
```

```
import java.lang.Integer;

class DoWhile {
    public static void main(String[] args) throws IOException {
        int score;
        char grade, ch;

        //set up buffer stream
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader buffer = new BufferedReader(isr);

        do {
            //get input string
            System.out.print("Enter your score: ");
            String input = buffer.readLine();
            //convert string to int
            score = Integer.parseInt(input);

            //determine grade by given score
            if(score > 90)
                grade = 'A';
            else if(score > 80)
                grade = 'B';
            else if(score > 70)
                grade = 'C';
            else if(score > 60)
                grade = 'D';
            else
                grade = 'F';

            System.out.println("Your grade is " + grade);
            System.out.print("\nDO you want to continue? <y/n> : ");
            String response = buffer.readLine();
            ch = (char)response.charAt(0);
        }
        while(ch == 'y');
    }
}
```

เราเพียงแต่ดัดแปลงโปรแกรมการหาเกรดของเรา ด้วยการเพิ่ม do … while และประโยค

```
System.out.print("\nDO you want to continue? <y/n> : ");
String response = buffer.readLine();
ch = (char)response.charAt(0);
```

ที่ใช้สำหรับการถาม user ว่าต้องการทำงานต่อหรือไม่ โดยเราทำการดึงเอาตัวอักษรตัวแรกที่อยู่ในบรรทัดนั้นออกมา ทำการเปรียบเทียบกับตัวอักษร y ใน expression ของ do … while loop ซึ่งถ้าการเปรียบเทียบนั้นเป็นจริงเราก็จะทำงานไปเรื่อย ๆ วิธีการเดียวที่จะทำให้ do … while loop ยุติการทำงานก็คือ การใส่ตัวอักษร n (หรือ อื่น ๆ ที่ไม่ใช่ y)

เราใช้ method chatAt() ด้วยค่า parameter ที่เป็นศูนย์เพื่อดึงเอาอักษรตัวแรกสุดใน buffer มาเก็บไว้ในตัวแปร ch สำหรับการเปรียบเทียบประโยคควบคุมของ do … while loop

ผลลัพธ์การ run โปรแกรมด้วยค่าต่าง ๆ

```
Enter your score: 89
Your grade is B
```

```
DO you want to continue? <y/n> : y
Enter your score: 12
Your grade is F

DO you want to continue? <y/n> : y
Enter your score: 75
Your grade is C

DO you want to continue? <y/n> : n
```

เรามาลองดูโปรแกรมตัวอย่างอีกตัวหนึ่งที่ใช้ do … while loop ในการหาผลรวมของเลขระหว่าง 1 – 100

```
/* Sum2.java */

import java.io.*;

class Sum2 {
    public static void main(String[] args) {
        int i, sum;

        i = 0;
        sum = 0;

        do {
            sum += i;
            i++;
        }
        while(i <= 100);

        System.out.println("Sum = " + sum);
        System.out.println("Average = " + (double)sum / (i-1));
    }
}
```

ซึ่งเมื่อ run ดูก็ได้ผลลัพธ์ดังนี้

```
Sum = 5050
Average = 50.5
```

เราไม่ได้เปลี่ยนแปลงอะไรมากมายใน loop ของเราเพียงแต่เปลี่ยนมาใช้ do … while loop เท่านั้นเอง

เพื่อให้เกิดความเข้าใจในการทำงานของ loop ดียิ่งขึ้น เราจึงนำเอาตัวอย่างของ code ที่ใช้ loop ในรูปแบบต่าง ๆ มาให้ดูกัน ผู้อ่านควรตรวจสอบถึงผลลัพธ์ที่ควรจะได้จากการทำงานของ loop เหล่านี้ด้วยมือ ก่อนที่จะเขียนโปรแกรมตรวจสอบ

ตัวอย่างที่ 1

```
class Loops {
    public static void main(String[] args) {
        int index = 1;

        while(index != 9) {
            index += 2;
            System.out.println(index);
            index--;
        }
```

```
    }
}
```

ตัวอย่างที่ 2

```
class Loops1 {
    public static void main(String[] args) {
        int index1, index2;

        index1 = 33;
        index2 = 1;

        while(index1 < index2) {
            index1 = index1 + 10;
            System.out.println(index1);
            index2 = index2 * 2;
        }
    }
}
```

ตัวอย่างที่ 3

```
class Loops2 {
    public static void main(String[] args) {
        int index = 1;

        while(index != 33) {
            index++;
            System.out.println(index);
            index++;
        }
    }
}
```

การใช้ break และ continue

หลาย ๆ ครั้งที่เราต้องการยุติการทำงานของ loop โดยที่ loop ยังไม่สิ้นสุดการทำงานของมันเอง เช่น กำหนดให้ loop หยุดการทำงานเมื่อค่าของ loop index มีค่าตรงตามเงื่อนไขที่ได้กำหนดไว้ ดังแสดงให้เห็นในโปรแกรมตัวอย่างนี้

```
/* Sum3.java */

import java.io.*;

class Sum3 {
    public static void main(String[] args) {
        int i, sum;

        i = 0;
        sum = 0;

        do {
            sum += i;
            i++;
            //break out of loop when i >= 50
            if(i >= 50)
                break;
        }
```

```
        while(i <= 100);

        System.out.println("Sum = " + sum);
        System.out.println("Average = " + (double)sum / (i-1));
    }
}
```

เราใช้โปรแกรม Sum2.java เป็นพื้นฐานของการเปลี่ยนแปลง โดยเรากำหนดให้มีการตรวจสอบค่าของ i ว่ามากกว่าหรือเท่ากันกับ 50 หรือไม่ซึ่งถ้าเป็นจริงเราก็จะ break ออกจาก loop ทันที

```
if(i >= 50)
    break;
```

เมื่อเราลอง run ดูเราได้ผลลัพธ์ดังนี้

```
Sum = 1225
Average = 25.0
```

ลองมาดูโปรแกรมที่ใช้ break ในการหยุดการทำงานของ loop ที่ใช้หาค่าของตัวเลขที่กำหนดให้ว่าเป็นเลขคู่จำนวนสิบตัว

```
/* Break.java */

import java.io.*;

class Break {
    public static void main(String[] args) {

        for(int number = 1; ; number++)
            if(number % 2 == 0) {
                System.out.println("Number is " + number);
                //break out of loop when number >= 10
                if(number >= 10)
                    break;
        }
    }
}
```

เราจะตรวจสอบว่าตัวเลขเป็นเลขคู่ก่อนแล้วจึงทำการแสดงตัวเลขนั้น ๆ ไปยังหน้าจอ เมื่อครบสิบตัวแล้วเราก็จะหยุดการทำงานของ loop ดังที่แสดงให้เห็นจากผลลัพธ์ที่ได้นี้

```
Number is 2
Number is 4
Number is 6
Number is 8
Number is 10
```

ทีนี้เรามาลองใช้ continue ดูบ้าง

```
/* Continue.java */

import java.io.*;

class Continue {
    public static void main(String[] args) {

        for(int number = 1; number <= 10; number++) {
```

```
            //skip if number is 5
            if(number == 5)
                continue;
            System.out.println("Number is " + number);
        }
    }
}
```

การประมวลของ code ที่เหลืออยู่ในโปรแกรมจะถูกประมวลผล ยกเว้นเมื่อค่าของ number มีค่าเท่ากับห้า ดังแสดงให้เห็นจากผลลัพธ์นี้

```
Number is 1
Number is 2
Number is 3
Number is 4
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
```

การใช้ continue นั้นโดยทั่วไปมักจะไม่ค่อยมีที่ใช้เท่าไรนัก เพราะการเขียนโปรแกรมโดยทั่วไปมักจะทำการประมวลผลกลุ่มของคำสั่งทั้งหมดจนเสร็จ หรือไม่ก็ยุติการทำงานเมื่อเงื่อนไขบางอย่างเป็นจริง การกระโดดข้ามการประมวลผลชุดคำสั่งไม่ค่อยจะมีให้เห็นมากมายนัก ดังนั้นก็คงต้องขึ้นอยู่กับการออกแบบโปรแกรมของผู้อ่านเองว่าเหมาะสมอย่างไรต่อการใช่ชุดคำสั่งต่าง ๆ

การใช้ Nested Loop

เราสามารถที่จะนำเอา loop ต่าง ๆ ที่เราได้พูดถึงมาใส่ไว้ใน loop ได้ เช่นเดียวกับที่เราทำกับ if – else เนื่องจากว่าตัว loop เองก็เป็นประโยคอย่างหนึ่ง ดังนั้นการนำประโยคมาวางซ้อนกันในการเขียนโปรแกรมก็ย่อมที่จะทำได้ เรามาลองดูตัวอย่างการใช้ nested loop กัน

```
/* NestedLoop.java */

import java.io.*;

class NestedLoop {
    public static void main(String[] args) {

        for(int row = 1; row <= 10; row++) {
            for(int column = 1; column <= 10; column++) {
                if(column > row)
                    continue;
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

โปรแกรม NestedLoop.java ของเราใช้ for loop สองตัวในการแสดงจำนวนของ '*' ในแต่ละแถว ดังที่แสดงให้เห็นจากผลลัพธ์นี้ (หน้าถัดไป)

```
*
**
***
****
*****
******
*******
********
*********
**********
```

ใน for loop ตัวแรกของเรา เราใช้ตัวแปร row เป็น loop index ซึ่งจะประมวลผลเป็นจำนวนเท่ากับสิบครั้ง และในแต่ละครั้งที่ row มีการประมวลผล for loop ตัวที่สองที่มี column เป็น loop index จะประมวลผลเป็นจำนวนเท่ากับสิบครั้ง เพราะฉะนั้นการประมวลผลของ for loop ทั้งสองตัวจึงมีค่าเป็นจำนวนเท่ากับหนึ่งร้อยครั้ง

ทุก ๆ การประมวลผลของ for loop ตัวที่สองเราจะตรวจสอบว่าค่าของ column มากกว่าค่าของ row หรือไม่ ซึ่งถ้ามากกว่าเราจะหยุดการแสดงค่า '*' ไปยังหน้าจอทันทีด้วยการเรียกใช้ continue ที่เราได้พูดถึงก่อนหน้านี้

ในขณะที่ row และ column มีค่าเท่ากับหนึ่งนั้นเราก็ส่ง '*' ไปยังหน้าจอ หลังจากนั้น column ก็มีค่าเป็นสอง ซึ่งมีค่ามากกว่า row การกระโดดข้ามจึงเกิดขึ้น ซึ่งจะเป็นไปจนกระทั่ง column มีค่าเท่ากับสิบเอ็ด การประมวลผลของ for loop ด้านในจึงยุติลง หลังจากนั้น row ก็มีค่าเป็นสอง เหตุการณ์เดิมก็เกิดขึ้นอีก แต่ตอนนี้การแสดงผล '*' จะมีเพิ่มขึ้นจนกว่าค่าของ column จะมากกว่า row เป็นเช่นนี้ไปเรื่อย ๆ จนกระทั่งค่าของ row มีค่าเป็นสิบเอ็ดการประมวลผลของ for loop ด้านนอกจึงยุติลง

โดยความเป็นจริงแล้วเราสามารถที่จะใช้ break แทนการใช้ continue ได้ ซึ่งก็จะได้ผลลัพธ์เหมือนกัน ทั้งนี้เพราะการใช้ break จะทำให้เราหลุดออกจาก for loop ด้านในทันที ซึ่งเป็นการประมวลผลที่ดีกว่าการใช้ continue เนื่องจากว่าเราประมวลผล for loop ด้านในด้วยจำนวนครั้งที่เท่ากับค่าของ row ที่เป็นอยู่ขณะนั้น (ผู้อ่านควรตรวจสอบดูว่าการประมวลผลของ for loop ด้านในมีจำนวนเท่ากับกี่ครั้ง ในทุก ๆ การประมวลผลของ for loop ด้านนอก - ทุกค่าของ row)

ตัวอย่างการใช้ nested-loop ในรูปแบบต่าง ๆ (ผู้อ่านควรตรวจสอบผลลัพธ์ด้วยมือก่อนการเขียนโปรแกรมทดสอบ)

ตัวอย่างที่ 1

```
class Loops3 {
    public static void main(String[] args) {
        int j, k = 0;

        while(k < 3) {
            j = 5;
            while(j > 0) {
                System.out.println(j);
                j -= 2;
            }
            System.out.println("k = " + k);
            k++;
        }
    }
}
```

ตัวอย่างที่ 2

```
class Loops4 {
    public static void main(String[] args) {
        int j, k = 1;

        while(k < 7) {
            j = 5;
            while(j > 0) {
                System.out.println("j = " + j);
                System.out.println("k = " + k);
                j--;
            }
            k *= 2;
        }
    }
}
```

ตัวอย่างที่ 3

```
class Loops5 {
    public static void main(String[] args) {
        int j = 10, k = 1;

        while(k < 8) {
            while(j > 0) {
                System.out.println("j = " + j);
                System.out.println("k = " + k);
                j--;
            }
            k = k * 2;
        }
    }
}
```

การใช้ Labeled continue และ Labeled break ใน for loop

Java ยอมให้มีการใช้ label ในการหยุดและเริ่มการประมวลผลใหม่ ซึ่งโปรแกรมเมอร์หลาย ๆ ท่านไม่ค่อยจะเห็นด้วยกับการใช้การประมวลผลในลักษณะนี้เท่าใดนัก เรามาดูตัวอย่างกัน

```
/* LabledContinue.java */

import java.io.*;

class LabeledContinue {
    public static void main(String[] args) {

        label: for(int row = 1; row <= 10; row++) {
                   System.out.println();
                   for(int column = 1; column <= 10; column++) {
                       if(column > row)
                           continue label;   //goto label
                       System.out.print("*");
                   }
               }
    }
}
```

เราได้ดัดแปลงโปรแกรม Continue.java ของเราสำหรับการแสดงการใช้ labeled continue ในโปรแกรม LabeledContinue.java เรากำหนดให้มี identifier ที่มีชื่อว่า label อยู่ก่อนหน้า for loop ด้านนอกของเรา ซึ่งต้องมีเครื่องหมาย ':' ตามหลังเสมอ และในการใช้ continue ใน for loop ด้านในนั้นเราจะต้องมี identifier ที่เราใช้ตามหลัง continue เสมอ

การประมวลด้วย labeled continue ก็ไม่ยากนักที่จะทำความเข้าใจ เมื่อค่าของ column มากกว่าค่าของ row การทำงานใน for loop ด้านในก็สิ้นสุดลงและไปเริ่มต้นใหม่ที่ for loop ด้านนอก เพราะนั่นเป็นที่ที่มี label อยู่ การทำงานแบบนี้ก็เหมือนกับการกระโดดไปยังตำแหน่งที่เราต้องการจะไปนั่นเอง ทั้งนี้ก็ต้องขึ้นอยู่กับเงื่อนไขที่เราได้ตั้งไว้

ถ้าเราเปลี่ยน continue ให้เป็น break เราจะได้ผลลัพธ์อะไร เรามาลองดูโปรแกรมที่ใช้ labeled break กันดู

```
/* LabledContinue.java */
import java.io.*;

class LabeledContinue {
    public static void main(String[] args) {

        System.out.println("Statement before first label.");
        first:
        for(int i = 1; i <= 10; i++) {
            for(int j = 1; j <= 10; j++) {
                if(j > i)
                    break first;
                System.out.println('*');
            }
        }
        System.out.println("Statement after first label.");
    }
}
```

เราได้เพิ่มประโยคทั้งก่อนและหลัง label ที่ชื่อว่า first ของเราเพื่อจะได้ดูว่าการประมวลผลเกิดขึ้นที่ใดบ้าง ก่อนที่เราจะประมวลผลของประโยคที่ตามหลัง first เราส่งข้อความไปยังหน้าจอว่า

```
Statement before first label.
```

และ for loop ของเราทั้งสองตัวส่งตัวอักษรเพียงตัวเดียวไปยังหน้าจอคือ

```
*
```

หลังจากนั้นก็ส่งข้อความ

```
Statement after first label.
```

ไปยังหน้าจอ จะเห็นว่าการใช้ labeled break นั้นทำให้การประมวลผลของเรายุติลง เพราะว่าประโยคที่มีอยู่ใน block ของ first นั้นมีเพียงประโยคเดียวดังนั้น Java จะประมวลผลประโยคที่อยู่ถัดไป หลังจาก block ของ first ซึ่งก็คือประโยค

```
System.out.println("Statement after first label.");
```

เราสามารถที่จะมี label block กี่ block ก็ได้ในโปรแกรมของเรา การใช้ labeled break ถ้าใช้ได้อย่างเหมาะสมแล้วจะทำให้การเขียน code เป็นไปได้อย่างสะดวกยิ่งขึ้น ทั้งนี้ก็เพราะว่าเราสามารถที่จะกระโดดไปยัง block ต่าง ๆ ได้อย่างง่าย ๆ

ลองมาดูโปรแกรมตัวอย่างอีกตัวหนึ่งที่ใช้ labeled break ในการหาผลคูณของเลขระหว่าง 1 – 5

```
/* LabledBreak.java */

import java.io.*;

class LabeledBreak {
    public static void main(String[] args) {
        int prod = 1;
        int i = 1;

        outer:
            while(true) {
                prod *= i;
                i++;
                if(i > 5)
                    break outer;
            }
        System.out.println("Product = " + prod);
    }
}
```

เราใช้ while loop ที่ทำงานอยู่ตลอดเวลาโดยไม่มีการกำหนดให้หยุด (ใช้ true ใน expression) แต่เรากำหนดให้มีการหยุดจากการใช้ labeled break ใน while loop เอง ตัวโปรแกรมจะคำนวณหาผลคูณไปจนกว่าค่าของ i จะเป็นห้าก็จะ break ออกจาก block ที่ชื่อ outer ทันที และจะไปประมวลผลประโยคที่อยู่ถัดมาจาก block ซึ่งก็คือ System.out.println("Product = " + prod)

Loop เป็นโครงสร้างที่ใช้มากในการเขียนโปรแกรมต่าง ๆ ทั้งนี้เพราะ loop เอื้ออำนวยให้ผู้เขียนเรียกใช้ชุดคำสั่งต่าง ๆ ซ้ำกันได้ ทำให้การเขียนโปรแกรมลดขั้นตอนการทำงานลง และการทำงานของโปรแกรมก็กระชับยิ่งขึ้น เราจะกลับมาพูดถึง loop อีกหลาย ๆ ครั้งในบทต่อ ๆ ไป โดยเฉพาะในเรื่องของ Array และ String

สรุป

ในบทนี้เราได้พูดถึงการใช้ประโยคของการเปรียบเทียบ และการตัดสินใจ การใช้ประโยคควบคุมการทำงาน การใช้ loop ในการประมวลผลติดต่อกัน โดยรวมแล้วเราได้พูดถึง

- ✓ การใช้ if และ if-else
- ✓ การใช้ nested-if
- ✓ การใช้ Tertiary operator - ? :
- ✓ การใช้ switch
- ✓ การใช้ loop ต่าง ๆ เช่น for-loop while-loop และ do-while-loop
- ✓ การใช้ break continue และการใช้ประโยคที่ใช้ label

แบบฝึกหัด

1. จงเขียนโปรแกรมที่รับ int จาก keyboard จำนวน 10 ตัว หลังจากนั้นให้คำนวณหาผลรวม และค่าเฉลี่ยของข้อมูลทั้ง 10 ตัว ให้ส่งผลลัพธ์ออกทางหน้าจอ

2. จงเขียนโปรแกรมที่สร้างตัวเลขแบบสุ่มจำนวน 100 ตัว เสร็จแล้วให้แสดงผลลัพธ์ออกทางหน้าจอเฉพาะตัวเลขที่เป็นเลขคี่เท่านั้น โดยกำหนดให้การแสดงผลเป็น 10 ตัวต่อหนึ่งบรรทัด

3. จากโจทย์ในข้อ 2 แทนที่จะแสดงผลเป็นเลขคี่ให้แสดงผลเฉพาะที่เป็นเลขคู่เท่านั้น โดยแสดงเฉพาะเลขคู่ที่มากกว่า 20 เท่านั้น

4. จงเขียนโปรแกรมที่แสดงตัวอักษรตั้งแต่ A ไปจนถึง Z ด้วยการใช้ while loop โดยไม่มีการรับค่าใด ๆ จาก keyboard

5. จงเขียนโปรแกรมที่คำนวณหาค่าที่เล็กที่สุดในกลุ่มตัวอักษรที่อ่านเข้ามาจาก keyboard ทีละตัว (ตัวอักษรที่นำเข้าควรมีจำนวนไม่น้อยกว่า 10 ตัว)

6. จงเขียนโปรแกรมที่อ่านข้อมูล 2 ตัวจาก keyboard โดยที่ตัวแรกเป็น หมายเลขของสินค้า (product number) และตัวที่สองเป็น จำนวนที่ขายไปในแต่ละวัน (quantity sold) กำหนดให้มีสินค้าอยู่ 5 ชนิดที่มีราคาต่างกัน ดังนี้

   สินค้าที่ 1 ราคา 500.0 บาท
   สินค้าที่ 2 ราคา 750.0 บาท
   สินค้าที่ 3 ราคา 850.0 บาท
   สินค้าที่ 4 ราคา 950.0 บาท
   สินค้าที่ 5 ราคา 1000.0 บาท

   โปรแกรมจะคำนวณหายอดเงินที่ขายได้ของสินค้าทุกตัว ในอาทิตย์ที่ผ่านมา โดยที่จำนวนของสินค้าแต่ละชนิดที่ขายไปจะนำเข้ามาจาก keyboard โปรแกรมที่เขียนขึ้นต้องใช้ switch ในการแยกแยะราคาของสินค้าแต่ละชนิดที่เข้ามา ให้แสดงผลลัพธ์ออกทางหน้าจอ

7. จงเขียนโปรแกรมที่คำนวณหาดอกเบี้ยทบต้นจากเงินต้นที่อ่านเข้ามาจาก keyboard ด้วยสูตรที่กำหนดให้ ต่อไปนี้ (คล้าย ๆ กับตัวอย่างในบทนี้)

   $a = p(1 + r)^n$

   p หมายถึงเงินลงทุนครั้งแรก
   r หมายถึงอัตราดอกเบี้ยต่อปี
   n หมายถึงจำนวนปีที่ต้องการฝากเงิน
   a จำนวนเงินที่ฝากเข้าทุก ๆ สิ้นปี

   โปรแกรมควรแสดงรายการออกเป็นสอง column โดยที่ column แรกเป็นปีที่ฝาก และ column ที่สองเป็นดอกเบี้ยที่ได้ในแต่ละปี ให้ทดลอง run โปรแกรมด้วยอัตราดอกเบี้ย 5% ต่อปี และเงินต้นที่ 1000 บาท เป็นเวลา 10 ปี

8. จงเขียนโปรแกรมที่แสดงผลลัพธ์ ดังที่เห็นนี้ ด้วยการใช้ loop เท่านั้น

```
**********      **********
*********        *********
********          ********
*******            *******
******              ******
*****                *****
****                  ****
***                    ***
**                      **
*                        *
```

9. จงเขียนโปรแกรมที่รับค่า int หนึ่งตัว (n) ที่มากกว่าศูนย์ หลังจากนั้นให้แสดงผล ดังนี้

   1 2 3 … n-1 n
   1 2 3 … n-1
   …
   1 2 3
   1 2
   1

10. จงเขียนโปรแกรมที่รับค่าเป็น วัน เดือน ปี จาก keyboard หลังจากนั้นให้แสดงตำแหน่งของวันที่นั้นในปี เช่น 12 29 2002 จะแสดงเป็น 363

11. จงเขียนโปรแกรมที่รับค่าต่าง ๆ จาก keyboard เฉพาะที่เป็น 0 มากกว่าศูนย์ และ น้อยกว่าศูนย์ ให้นับจำนวนของค่าทั้งสามว่ามีอย่างละกี่ตัว โปรแกรมจะหยุดทำงานได้ก็ต่อเมื่อ user ใส่ข้อมูลที่ไม่ใช่ตัวเลข

12. จงเขียนโปรแกรมที่รับค่าระหว่าง 1 – 100 หลังจากนั้นให้แสดงค่าที่รับเข้ามาในรูปแบบของตัวหนังสือ เช่น ถ้าข้อมูลนำเข้าเป็น 11 ผลลัพธ์ที่ได้คือ สิบเอ็ด

13. จงเขียนโปรแกรมที่ทำการเข้ารหัสของข้อมูลที่นำเข้าจาก keyboard โดยนำเอาตัวอักษรที่อ่านเข้าบวกกับค่าของตัวอักษรที่อยู่ถัดไป 2 ตัวอักษร mod ด้วยจำนวนของตัวอักษรที่มีอยู่ (26 ในภาษาอังกฤษ) เช่น ถ้าค่านำเข้าเป็น a ผลลัพธ์ที่ได้จากการเข้ารหัสจะเป็น a + c ซึ่งจะมีค่าเท่ากับ 97 + 99 = 196 % 26 = 14 ให้ใช้ค่านี้เป็นตัวบอกตำแหน่งของตัวอักษรที่จะแสดงไปยังหน้าจอ ซึ่งตัวอักษร ณ ตำแหน่งที่ 14 นี้คือ n

14. จงเขียนโปรแกรมที่รับค่าเป็นประโยคทางคณิตศาสตร์ เช่น 9 + 5 – 2 * 4 + 2 / 3 หลังจากนั้นให้ทำการประมวลผลประโยคดังกล่าว ตามลำดับของข้อมูลที่นำเข้า

15. จงเขียนโปรแกรมที่รับค่าเดือนที่เป็นตัวเลขระหว่าง 1 – 12 หลังจากนั้นให้แสดงผลเป็นจำนวนของวันที่มีอยู่ในเดือนดังกล่าว

16. จงหาผลลัพธ์ของตัวแปร sum จาก code ที่ให้

```
class Loops6 {
 public static void main(String[] args) {
    int j = 0, k, m, n, sum = 0;

    while(j != 20) {
        k = 0;
        while(k != 100) {
            m = 0;
            while(m != 50) {
                n = 0;
                while(n != 10) {
                    sum++;
                    n++;
                }
                m++;
            }
            k++;
        }
        j++;
    }
 }
}
```

17. จงเขียนโปรแกรมที่คำนวณหา ตัวหารร่วมมาก (GCD – Greatest Common Divisor) ของตัวเลขสองตัวที่รับเข้ามาจาก keyboard เช่น gcd ของ 24 และ 15 คือ 3

    note: gcd ของ x และ y คือ integer ที่ใหญ่ที่สุดที่หาร x และ y ลงตัว (เหลือเศษ 0)

18. จงเขียนโปรแกรมที่คำนวณหา ตัวคูณร่วมน้อย (LCD – Least Common Multiple) ของตัวเลขสองตัวที่รับเข้ามาจาก keyboard เช่น lcd ของ 24 และ 15 คือ 120

    note: lcd ของ x และ y คือ integer ที่น้อยที่สุดที่ทั้ง x และ y หารลงตัว

19. จงเขียนโปรแกรมที่หา factor ของตัวเลขที่รับเข้ามาจาก keyboard เช่น ถ้าข้อมูลเท่ากับ 18 factor ที่ได้คือ 2 3 และ 3 (8 = 2 * 3 * 3)

4
การใช้ Array และ String

ในบทนี้เราจะมาดูการสร้างและใช้ array ซึ่งเป็นโครงสร้างอีกตัวหนึ่งที่ทำให้เราสามารถเก็บข้อมูลที่เป็นชนิดเดียวกันหลาย ๆ ตัวไว้ในตัวแปรตัวเดียว รวมไปถึงการสร้างและใช้ string ซึ่งจะเป็นการแนะนำให้ผู้อ่านได้รู้จักกับ object ในภาษา Java แบบคร่าว ๆ

หลังจากจบบทเรียนนี้แล้ว ผู้อ่านจะได้ทราบถึง

- การประกาศและกำหนดค่าให้ array
- การเข้าหาข้อมูลในตำแหน่งต่าง ๆ ที่อยู่ใน array
- การสร้าง array ที่ใช้เก็บ array
- การสร้าง object ที่เป็น string
- การสร้างและใช้ array ที่เก็บ object
- การใช้กระบวนการต่าง ๆ กับ object ที่เป็น string

การใช้ array

Array เป็นโครงสร้างชนิดหนึ่งที่ประกอบไปด้วยข้อมูลชนิดเดียวกันที่อาจมีมากกว่าหนึ่งตัว การเข้าหาข้อมูลแต่ละตัวที่อยู่ใน array จะต้องทำผ่านทาง ชื่อของ array และ index ที่มีค่าเป็น integer (int) ประกอบกัน Java ได้กำหนดให้ index ตัวแรกสุดที่เก็บข้อมูลมีค่าเป็น 0 และจะเพิ่มขึ้นทีละหนึ่งค่าไปเรื่อย ๆ จนกว่าจะหมดข้อมูลที่มีอยู่ใน array นั้น ๆ การประกาศตัวแปรที่เป็น array ทำได้ดังนี้

datatype[] nameOfArray;

เช่น

```
int[] listOfNumbers;
double[] prices;
char[] vowels;
```

ในการประกาศ array นั้นเราจะต้องกำหนดจำนวนของข้อมูลสูงสุดที่ array นั้นสามารถเก็บได้ ชนิดของข้อมูลที่จะเก็บไว้ใน array นั้น เครื่องหมาย [] เป็นตัวบอกให้ Java รู้ว่าตัวแปรที่ถูกสร้างขึ้นมานี้เป็น array และเครื่องหมายนี้ก็จะเป็นตัวที่ใช้ในการเข้าหาข้อมูลแต่ละตัวใน array หนังสือหลาย ๆ เล่มเลือกที่จะประกาศ array ในรูปแบบที่เห็นด้านล่างนี้ ซึ่งมีความหมายเหมือนกันกับที่เราได้ประกาศก่อนหน้านี้

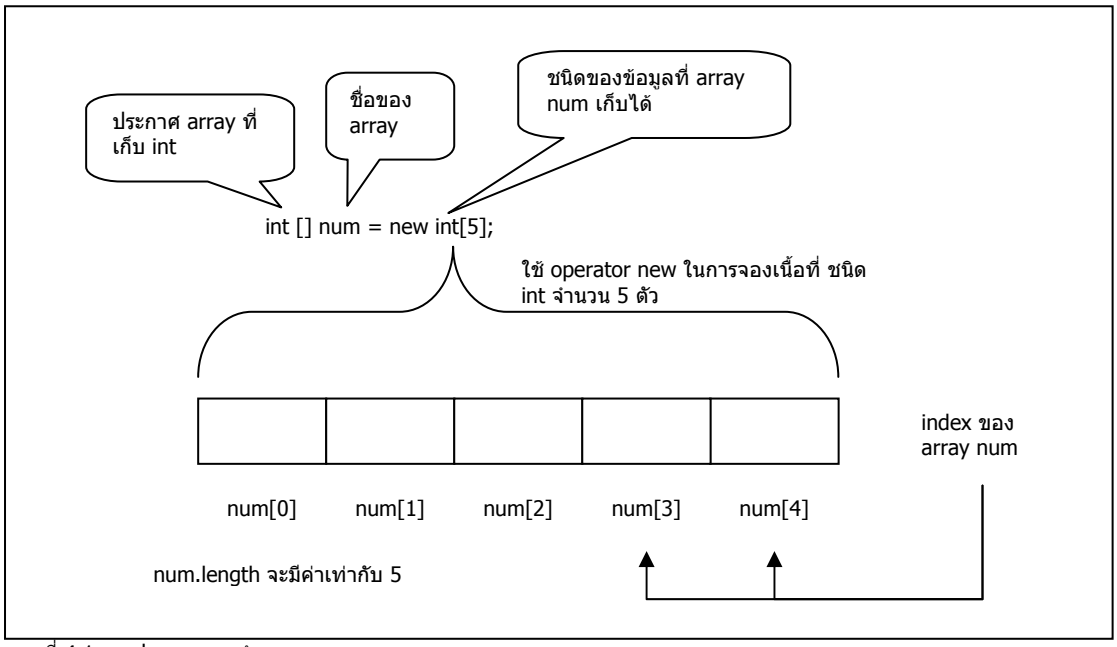
```
int listOfNumbers[];
double prices[];
char vowels[];
```

สำหรับหนังสือเล่มนี้เราจะใช้การประกาศในแบบที่หนึ่ง

หลังจากที่เราได้ประกาศให้ตัวแปรเป็น array แล้วเราก็ต้องกำหนดความจุให้กับ array นั้น ดังนี้

```
listOfNumbers = new int[5];
```

ประโยคที่เราได้เขียนขึ้นนั้นบอกให้ Java รู้ว่าเราต้องการให้ Java จองเนื้อที่ในการเก็บ int เป็นจำนวนเท่ากับ 10 ตัวให้กับตัวแปร listOfNumbers ซึ่งตัวแปรนี้จะเป็นตัวอ้างอิง (reference) ถึงข้อมูลทั้งหมดที่ได้ถูกเก็บไว้ใน array นี้ ภาพที่ 4.1 แสดงถึงสิ่งต่าง ๆ ที่เกิดขึ้นเมื่อเราประกาศการใช้ array



ภาพที่ 4.1 การประกาศและกำหนดขนาดของ array

เมื่อมีการประกาศใช้ array เช่น

```
int [] num;
```

นั้น array ที่เราได้ประกาศนั้นยังไม่มีเนื้อที่ในหน่วยความจำเลย เราต้องกำหนดขนาดเพื่อให้เกิดการจองเนื้อที่ในหน่วยความจำด้วยการใช้ operator new รวมไปถึงจำนวนของเนื้อที่ที่ต้องการใช้ เช่น

```
int [] num = new int[5];
```

และทุกครั้งของการจองเนื้อที่ Java จะกำหนดให้ข้อมูลในทุก ๆ ตำแหน่งมีค่าเป็น 0 เสมอ

ในการเข้าหาข้อมูลแต่ละตัวนั้น เราก็เพียงแต่กำหนด index ของข้อมูลที่เราต้องการเข้าหา เช่น

```
num[3]
num[2]
```

เราไม่สามารถเข้าหาข้อมูล ณ ตำแหน่งที่มีค่า index เกิน 4 ได้ทั้งนี้เพราะเรากำหนดให้ array num มีความจุสูงสุดเท่ากับ 5 ดังนั้นค่าของ index ที่เป็นไปได้ คือ 0 1 2 3 และ 4 ถ้าหากว่าเราเข้าหาข้อมูลที่ไม่อยู่ในตำแหน่งที่กล่าวนี้ Java จะฟ้องด้วย error ทันที (array index out of bound)

การกำหนดค่าเบื้องต้นให้กับ array

เรารู้ว่าหลังจากการประกาศและจองเนื้อที่ให้กับ array นั้นถ้าเป็น array ที่เก็บ int ทุก ๆ หน่วยความจำของ array จะมีค่าเป็น 0 แต่ถ้าเราต้องการให้เป็นค่าอื่นเราก็สามารถทำได้ ดังโปรแกรมตัวอย่างนี้

```
/* Array1.java */

import java.io.*;

class Array1 {
```

```
    public static void main(String[] args) {
        int[] list;           //declare array of int

        list = new int[10];   //allocate space for 10 ints
        for(int i = 0; i < list.length; i++)
            System.out.println("list[" + i +"] = " + list[i]);
    }
}
```

โปรแกรม Array1.java ประกาศให้ตัวแปร list เป็น array ที่เก็บ int เป็นจำนวนเท่ากับ 10 ตัว เมื่อประกาศเสร็จแล้ว เราก็ใช้ for loop ในการเข้าหาข้อมูลในแต่ละตัว array โดยแสดงถึงค่าที่ Java กำหนดให้กับ array ของเราไปยังหน้าจอ ซึ่งมีค่าดังนี้

```
list[0] = 0
list[1] = 0
list[2] = 0
list[3] = 0
list[4] = 0
list[5] = 0
list[6] = 0
list[7] = 0
list[8] = 0
list[9] = 0
```

การใช้ loop เป็นวิธีการที่ง่ายที่สุดในการเข้าหาข้อมูลทุกตัวที่อยู่ใน array ยกเว้นว่าเรารู้ตำแหน่งที่แน่นอนของข้อมูลที่เราต้องการใน array นั้น เช่น ตำแหน่งที่ 5 หรือตำแหน่งที่ 3 เราก็เข้าหาได้โดยตรงดังนี้ list[5] หรือ list[3] เป็นต้น

Array ที่สร้างขึ้นใน Java นั้นเราสามารถที่จะดึงเอาขนาดของ array ออกมาได้ด้วยการเรียกใช้ค่าคงที่ที่ได้ถูกสร้างขึ้นหลังจากการประกาศและจองเนื้อที่ให้กับ array ดังนี้

```
list.length;
```

ทุก array ที่ได้ถูกสร้างขึ้นจะมีขนาดของ array เก็บไว้ในตัวแปรชื่อ length และเราเรียกขนาดออกมาได้ผ่านทางชื่อของ array ตามด้วย . และตัวแปร length

เรามาลองกำหนดค่าในตำแหน่งต่าง ๆ ให้กับ array ของเรา

```
/* Array2.java */

import java.io.*;

class Array2 {
    public static void main(String[] args) {
        int[] list;           //declare array of int

        list = new int[10];   //allocate space for 10 ints
        list[5] = 8;
        list[3] = 2;
        list[9] = 4;
        for(int i = 0; i < list.length; i++)
            System.out.println("list[" + i +"] = " + list[i]);
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
list[0] = 0
list[1] = 0
list[2] = 0
list[3] = 2
list[4] = 0
list[5] = 8
list[6] = 0
list[7] = 0
list[8] = 0
list[9] = 4
```

เราสามารถที่จะกำหนดค่าให้กับ array โดยที่ไม่ต้องกำหนดขนาดให้กับ array โดยตรง แต่ให้ Java จัดการเกี่ยวกับเรื่องเนื้อที่ให้เรา ดังโปรแกรมตัวอย่างต่อไปนี้

```
/* Array3.java */

import java.io.*;

class Array3 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};

        for(int i = 0; i < list.length; i++)
            System.out.println("list[" + i +"] = " + list[i]);
    }
}
```

เรากำหนดให้ list เป็น array ที่เก็บ int และกำหนดค่าเบื้องต้นเป็น 2 3 1 56 และ 90 ตามลำดับ เมื่อ Java เห็นการประกาศในลักษณะนี้ก็จะทำการจองเนื้อที่ให้กับ array โดยอัตโนมัติพร้อมทั้งนำค่าต่าง ๆ ที่ได้ถูกกำหนดจาก user ไปใส่ไว้ใน array list และเมื่อ run ดูเราก็ได้ผลลัพธ์ดังที่เห็นนี้

```
list[0] = 2
list[1] = 3
list[2] = 1
list[3] = 56
list[4] = 90
```

โปรแกรมตัวอย่างต่อไปนี้แสดงถึงการเก็บข้อมูลที่ user ใส่เข้ามาจาก keyboard เข้าสู่ array

```
/* ArrayOfNumbers.java */

import java.io.*;
import java.text.DecimalFormat;
import java.lang.Integer;

class ArrayOfNumbers {
    public static void main(String[] args) throws IOException {
        double number, sum = 0;  //a number read and sum of numbers
        double[] list;           //array of double
        int count;               //number of items in array
        String input;            //input string
        BufferedReader buffer;   //input buffer
        InputStreamReader isr;   //input stream
        DecimalFormat f = new DecimalFormat("0.00");

        System.out.print("How many numbers? : ");
        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
```

```
        input = buffer.readLine();

        count = Integer.parseInt(input); //total items in list
        list = new double[count];        //allocate space forlist
        //reading each double into list
        for(int i = 0; i < list.length; i++) {
            System.out.print("Enter number: ");
            input = buffer.readLine();
            list[i] = Double.parseDouble(input);
        }

        //add all doubles in list
        int n = list.length - 1; //n is the last index of list
        while(n >= 0) {          //looping until n equals to 0
            sum += list[n--];    //add list[n] into sum and
        }                        //decrement n

        //calculate average and display result
        double average = sum / count;
        System.out.println("Sum of numbers = " + sum);
        System.out.println("Average of numbers = " +
                           f.format(average));
    }
}
```

โปรแกรม ArrayOfNumbers.java ถาม user ถึงจำนวนของข้อมูลสูงสุดที่ user จะใส่เข้าสู่โปรแกรม ซึ่งเมื่อได้แล้วโปรแกรมจะใช้จำนวนที่อ่านได้เป็นตัวกำหนดขนาดให้กับ array

```
list = new double[count];
```

หลังจากนั้นก็จะอ่านข้อมูลทีละตัวจาก user นำไปเก็บไว้ใน array list

```
for(int i = 0; i < list.length; i++) {
    System.out.print("Enter number: ");
    input = buffer.readLine();
    list[i] = Double.parseDouble(input);
}
```

ในขณะที่อ่านข้อมูลเข้าสู่ array นั้นเราอาจหาผลรวมของเลขทุกตัวพร้อม ๆ กันไปด้วยก็ได้ แต่ที่โปรแกรมนี้แยกการหาผลรวมก็เพื่อที่จะแสดงให้เห็นถึงการใช้ loop ในการเข้าหา array อีกครั้งหนึ่ง โดยเรากำหนดให้ตัวแปร n เป็น index ตัวสุดท้ายของ array list ด้วยประโยค

```
int n = list.length - 1;
```

เราต้องไม่ลืมว่า list.length ให้จำนวนของข้อมูลทั้งหมดที่อยู่ใน array ดังนั้น index ตัวสุดท้ายก็คือ list.length – 1 หลังจากนั้นเราจะใช้ while loop เป็นตัวเข้าหาข้อมูลใน array แต่ละตัวจากข้างหลัง และบวกข้อมูลทุก ๆ ตัวเข้ากับตัวแปร sum

```
while(n >= 0) {
    sum += list[n--];
}
```

ประโยค sum += list[n--] นั้นจะบวกข้อมูลในตำแหน่ง n เข้ากับตัวแปร sum ก่อนที่จะลดค่าของ n ลงหนึ่งค่า และเมื่อค่า n น้อยกว่า 0 เราก็หลุดออกจาก while loop ทันที

การอ้างถึง array ด้วยการ clone และการ copy array

เมื่อเราสร้าง array ขึ้นมาแล้วเราสามารถที่จะอ้างถึงตัวแปรตัวนี้ด้วยตัวแปรอื่นก็ได้ แต่ไม่ได้หมายความว่าเราได้สร้าง array ตัวใหม่จากตัวแปรตัวใหม่นี้ เราเพียงแต่ใช้ตัวแปรตัวใหม่ในการอ้างถึง array ตัวเดิม เพราะฉะนั้นการกระทำใด ๆ ผ่านตัวแปรตัวใหม่ก็จะมีผลกระทบกับข้อมูลใน array ตัวเดิมนั้น ดังโปรแกรมตัวอย่างนี้

```
/* Array4.java */

import java.io.*;

class Array4 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};

        for(int i = 0; i < list.length; i++)
            System.out.println("list[" + i +"] = " + list[i]);

        int[] newList = list; //cloning
        newList[0] = 11;      //changing data at list[0]
        for(int i = 0; i < list.length; i++)
            System.out.println("list[" + i +"] = " + list[i]);
    }
}
```

เราได้ดัดแปลงโปรแกรม Array3.java ให้มีการอ้างถึง (reference) array list ด้วยตัวแปรตัวใหม่ที่ชื่อ newList

```
int[] newList = list;
```

และเราก็ได้ทำการเปลี่ยนข้อมูล ณ ตำแหน่ง 0 ให้เป็น 11 ผ่านทาง newList เมื่อลอง run ดูเราก็จะได้ผลลัพธ์ดังที่คาดไว้

```
list[0] = 2
list[1] = 3
list[2] = 1
list[3] = 56
list[4] = 90

list[0] = 11
list[1] = 3
list[2] = 1
list[3] = 56
list[4] = 90
```

ส่วนวิธีการ copy ข้อมูลจาก array ตัวหนึ่งไปยัง array อีกตัวหนึ่งนั้นมีวิธีการอยู่สามวิธีที่ทำได้ วิธีที่หนึ่งนั้นเราต้องสร้าง array ใหม่ขึ้นมาแล้วก็ copy ข้อมูลทุกตัวจาก array ตัวเดิมเข้าสู่ array ตัวใหม่ ดังนี้

```
/* Array5.java */

import java.io.*;

class Array5 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};         //original array
        int[] newList = new int[list.length];   //new array
```

```
        //copy each item from list into newList
        for(int i = 0; i < list.length; i++)
            newList[i] = list[i];

        for(int i = 0; i < list.length; i++)
            System.out.println("newList[" + i +"] = " + newList[i]);

    }
}
```

โปรแกรม Array5.java สร้าง newList ด้วยคำสั่ง

```
int[] newList = new int[list.length];
```

เรากำหนดขนาดของ newList ด้วยขนาดที่มาจาก list หลังจากนั้นเราก็ copy ข้อมูลทุกตัวด้วยการใช้ for – loop

```
for(int i = 0; i < list.length; i++)
    newList[i] = list[i];
```

และเมื่อ run ดูเราก็ได้ผลลัพธ์ดังที่คาดไว้ คือ

```
newList[0] = 2
newList[1] = 3
newList[2] = 1
newList[3] = 56
newList[4] = 90
```

ทีนี้เรามาดูวิธีที่สองกัน

```
/* Array6.java */

import java.io.*;

class Array6 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};         //original array
        int[] newList = new int[list.length];   //new array

        //copy each item from list into newList
        System.arraycopy(list, 0, newList, 0, list.length);

        for(int i = 0; i < newList.length; i++)
            System.out.println("newList[" + i +"] = " + newList[i]);

    }
}
```

ถึงแม้ว่าเรายังไม่ได้พูดถึงเรื่องของ class หรือ method แต่เราก็ต้องมาทำความเข้าใจถึงการเรียกใช้ method ที่ Java มีให้ เพราะว่าวิธีที่สองนี้เราใช้ method arraycopy() ทำการสร้างและ copy ข้อมูลให้ ตอนนี้เราคงต้องจำวิธีการเรียกใช้มากกว่าการทำความเข้าใจอย่างละเอียด (ซึ่งเราจะพูดถึงในบทต่อ ๆ ไป) เรา copy ด้วยการใช้คำสั่ง

```
System.arraycopy(list, 0, newList, 0, list.length);
```

โดยเราต้องใส่ข้อมูล หรือ เงื่อนไข (parameter) ตามที่ Java กำหนดไว้คือ

ตัวแรกเป็น array ตัวเดิมที่ต้องการ copy (list)
ตัวที่สองเป็น index เริ่มต้นของ array ตัวเดิม (0)
ตัวที่สามเป็น array ตัวใหม่ที่ต้องการสร้าง (newList)
ตัวที่สี่เป็น index เริ่มต้นของ array ตัวใหม่ (0)
ตัวที่ห้าเป็น จำนวนของข้อมูลที่ต้องการ copy (list.length)

วิธีที่สามก็คล้าย ๆ กับวิธีที่สองเพียงแต่เราเรียกใช้ method clone() แทน ซึ่งทำได้ดังนี้

```
/* Array7.java */

import java.io.*;

class Array7 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};         //original array

        //cloning process
        int[] newList = (int[])list.clone();

        for(int i = 0; i < list.length; i++)
            System.out.println("newList[" + i +"] = " + newList[i]);

    }
}
```

เราเพียงแต่เรียกใช้ method clone() ดังที่เห็นในโปรแกรม Array7.java เราก็สามารถที่จะมี array ตัวใหม่สำหรับการใช้งาน ด้วยประโยค

```
int[] newList = (int[])list.clone();
```

เราจำเป็นที่จะต้องทำการ cast ให้กับ array ของเราเสียก่อนเพื่อให้ชนิดของ array ที่ถูก clone ขึ้นมานั้นเป็นชนิดเดียวกัน ซึ่งทำได้ด้วยการนำเอา (int[]) ไปใส่ไว้หน้าประโยค list.clone() ทั้งนี้ต้องกำหนดชื่อ array ตัวเดิมให้ถูกต้อง (list) เพื่อไม่ให้เกิดการผิดพลาดใด ๆ

วิธีที่สามที่ได้กล่าวไว้จะ copy ข้อมูลใน array เดิมที่มีอยู่ทั้งหมดไปเก็บไว้ใน array ตัวใหม่ ดังนั้น ถ้าหากว่าเราต้องการที่จะสร้าง array ใหม่จากข้อมูลบางส่วนของ array ตัวเดิมเราก็ต้องใช้วิธีที่สอง ดังตัวอย่างนี้

```
/* Array8.java */

import java.io.*;

class Array8 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};         //original array
        int[] newList = new int[list.length-2]; //new array

        //copy item at index 2, 3, and 4 from list into newList
        System.arraycopy(list, 2, newList, 0, list.length - 2);

        for(int i = 0; i < newList.length; i++)
            System.out.println("newList[" + i +"] = " + newList[i]);
    }
}
```

โปรแกรม Array8.java ของเราต้องการที่จะ copy ข้อมูล ณ ตำแหน่งที่ 2 3 และ 4 จาก array list เข้าสู่ array newList ในตำแหน่ง 0 1 และ 2 ตามลำดับ เพราะฉะนั้นเราต้องเรียกใช้ method arraycopy() ดังนี้

```
System.arraycopy(list, 2, newList, 0, list.length - 2);
```

ผลลัพธ์ของการ run คือ

```
newList[0] = 1
newList[1] = 56
newList[2] = 90
```

การใช้ array แบบยืดหยุ่น (Dynamic Array)

เราสามารถที่จะใช้เทคนิคที่เราพูดมาก่อนหน้านี้ในการสร้าง array แบบไม่จำกัดจำนวนของข้อมูลที่สามารถจัดเก็บได้ใน array นั้น ๆ เราเรียกการจัดเก็บแบบนี้ว่าการจัดเก็บแบบยืดหยุ่น แต่ก่อนที่เราจะสร้างได้นั้นเราต้องกำหนดให้ array เริ่มต้นของเรามีขนาดเบื้องต้นก่อน หลังจากนั้นจึงจะสามารถขยายขนาดของ array ได้ ดังโปรแกรมตัวอย่างที่ได้ดัดแปลงมาจากโปรแกรม AddNumbers.java ที่เขียนขึ้นก่อนหน้านี้

```
/* DynamicArray.java */

import java.io.*;
import java.text.DecimalFormat;

class DynamicArray {
    public static void main(String[] args) throws IOException {
        double number, sum = 0;  //a number read and sum of numbers
        double []list = new double[2];   //array of double
        int count = 0;                   //total numbers read
        BufferedReader buffer;           //input buffer
        InputStreamReader isr;           //input stream
        StreamTokenizer token;           //token from input stream
        DecimalFormat f = new DecimalFormat("0.00");

        System.out.print("Enter numbers (space between -
                          enter when done): ");
        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
        //get tokens from input buffer
        token = new StreamTokenizer(buffer);
        //set priority to EOL so that we can break out of the loop
        token.eolIsSignificant(true);
        //get each number from tokens
        //calculate the sum until EOL is reached
        while(token.nextToken() != token.TT_EOL) {
            switch(token.ttype) {
                //token is a number
                case StreamTokenizer.TT_NUMBER:
                    //double the size of list
                    if(count > list.length-1) {
                        double []newList = new double[list.length*2];
                        System.arraycopy(list, 0, newList, 0,
                                         list.length);
                        list = newList;       //point to new list
                    }
                    list[count] = token.nval;//save token into list
```

```
                    sum += list[count];       //calculate sum
                    count++;                  //number of tokens read
                    break;
                default:
                    System.out.println("Token is not a number! ");
                    break;
            }
        }
        //calculate average and display result
        double average = sum / count;
        System.out.println("Sum of numbers = " + sum);
        System.out.println("Average of numbers = " +
                           f.format(average));
    }
}
```

เราประกาศตัวแปร list ให้เป็น array ที่เก็บ double เริ่มต้นด้วยจำนวนเท่ากับ 2ตัว และในการนำข้อมูลเข้าสู่ array นั้นเราจะตรวจสอบดูว่าจำนวนของ index ที่เราใช้ในการเข้าหาข้อมูลแต่ละตัวมีค่าเกินจำนวนของข้อมูลที่เป็นอยู่ขณะนั้นหรือไม่ ซึ่งถ้าเป็นจริงเราก็จะทำการขยายขนาดให้กับ array list เป็นจำนวนสองเท่าของขนาดเดิม และทำการ copy ข้อมูลจาก array list ไปสู่ array ใหม่นี้ (newList) หลังจากนั้นเราก็อ้างถึง array ตัวใหม่นี้ด้วยตัวแปรเดิมที่เราใช้ในการเก็บข้อมูลครั้งแรก (list) เราไม่ต้องกังวลถึงหน่วยความจำที่เราได้ใช้ไปในการสร้าง array ตัวเดิมเพราะ Java จะทำการตรวจสอบถึงหน่วยความจำที่ไม่มีการใช้งาน และจะคืนหน่วยความจำนี้ให้กับระบบโดยอัตโนมัติ

```
if(count > list.length-1) {
    double []newList = new double[list.length*2];
    System.arraycopy(list, 0, newList, 0, list.length);
    list = newList;
}
```

วิธีการขยายขนาดของ array แบบนี้ทำให้เราไม่ต้องกำหนดขนาดของ array ให้มีขนาดใหญ่มากมายเกินไปตอนที่เราสร้าง array ในครั้งแรก เราสนใจเฉพาะเมื่อจำเป็นต้องขยายขนาดของ array จริง ๆ เท่านั้น ผู้อ่านอาจเปลี่ยนแปลงการขยายขนาดของ array ด้วยการเพิ่มทีละกี่ตัวก็ได้ ทั้งนี้ก็ขึ้นอยู่กับความจำเป็นและเหมาะสมกับงานนั้น ๆ

ผลลัพธ์ของการ run โปรแกรม DynamicArray.java

```
Enter numbers (space between - enter when done): 15 5 20 15 10 15 20
Sum of numbers = 100.0
Average of numbers = 14.29
```

โปรแกรมตัวอย่างการใช้ array ในการนับจำนวนครั้งของลูกเต๋าที่ออกซ้ำกันจากการโยนหลาย ๆ ครั้ง

```
/* DiceRolling.java */

import java.io.*;

class DiceRolling {
    public static void main(String[] args) {
        int face, roll;
        int[] frequency = new int[7];//number of times each face occur

        //rolling the die for 100 times
        roll = 1;
        while(roll <= 100){
            //calculate face of die
            face = (int)(Math.random() * 6) + 1;
```

```
            //save each time the same face occur by using
            //indeces as faces of die
            frequency[face]++;
            roll++;
        }

        System.out.println("Face\tFrequency");
        for(int i = 1; i < frequency.length; i++)
            System.out.println(i + "\t" + frequency[i]);

    }
}
```

โปรแกรม DiceRolling.java ใช้ตัว index ของ array เป็นหน้าของลูกเต๋าที่ถูกโยนโดยการใช้การสุ่ม ผ่านทาง method random() ซึ่งจะบังคับให้ผลลัพธ์ของการโยนนั้นอยู่ระหว่าง 1 – 6 ด้วยการคูณ 6 เข้ากับผลลัพธ์ที่ได้จากการสุ่ม บวกเข้ากับ 1 การนับจำนวนครั้งก็ทำได้ด้วยการเพิ่มค่าให้กับข้อมูลที่ index นั้น ๆ ของลูกเต๋า เช่น ถ้าการโยนได้ค่าเท่ากับ 6 เราก็เพิ่มค่าด้วย frequency[6]++ ซึ่งจะทำให้ค่าที่อยู่ ณ ตำแหน่งนั้นเพิ่มขึ้นอีกหนึ่งค่า วิธีการใช้ index ของ array ในลักษณะนี้ช่วยลดขั้นตอนการนับจำนวนของหน้าลูกเต๋าที่เกิดขึ้น ได้ดีกว่าวิธีอื่น ๆ เช่น การใช้ if – else มาช่วยในการตรวจสอบว่าการโยนให้ค่าอะไร แล้วจึงเพิ่มค่าให้กับตัวแปรที่เก็บการนับของหน้านั้น ๆ

```
if(face == 1)
    count1++;
else if(face == 2)
    count2++;
else if(face == 3)
    count3++;
…
…

else if(face == 6)
    count6++;
```

ซึ่งเป็นวิธีที่ต้องเขียน code มากกว่าและใช้ตัวแปรเยอะกว่า การใช้ frequency[face]++

ผลลัพธ์ของการ run

```
Face    Frequency
1       12
2       25
3       24
4       7
5       12
6       19
```

การค้นหาข้อมูลใน array

การค้นหาข้อมูลใน array นั้นทำได้อยู่สองวิธีใหญ่ ๆ คือ การค้นหาแบบตามลำดับ (sequential search) และ การค้นหาแบบหารสอง (binary search) เราจะมาดูถึงวิธีการค้นหาแบบตามลำดับก่อน เพราะเป็นการค้นหาที่ง่ายที่สุด

การค้นหาแบบตามลำดับ (Sequential search)

ในการค้นหาแบบนี้นั้นเราจะเริ่มต้นจากข้อมูลตัวแรกสุดที่อยู่ใน array ทำการเปรียบเทียบข้อมูลตัวนี้กับข้อมูลที่ต้องการค้นหาว่าเป็นตัวเดียวกันหรือไม่ ถ้าใช่เราก็หยุดการค้นหา ถ้าเราค้นไปจนถึงตัวสุดท้ายแต่ก็ยังไม่ใช่ เราก็รู้ว่าข้อมูลที่ต้องการค้นหาไม่ได้อยู่ใน array ลองดูโปรแกรมตัวอย่างกัน

```
/* SeqSearch.java */

import java.io.*;

class SeqSearch {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};;

        int number = 90;      //number to search
        int i = 0;            //loop index
        boolean notFound = true; //loop control variable
        while(notFound) {
            if(number == list[i]) //if found
                notFound = false; //break out of loop
            i++;                  //move to next number
        }                         //in the list

        if(!notFound)
            System.out.println(number + " is in the list.");
        else
            System.out.println(number + " is not in the list.");
    }
}
```

เราใช้ while loop ในการเข้าหาข้อมูลทุกตัวที่อยู่ใน array list โดยเรากำหนดให้ตัวแปร notFound ซึ่งมีชนิดเป็น boolean เป็นตัวควบคุมการทำงานของ loop ซึ่งถ้าการค้นหาประสพผลสำเร็จ ตัวแปร notFound ก็จะถูกกำหนดให้เป็น false และ while loop ของเราก็จะยุติการทำงานทันที หลังจากนั้นเราก็แสดงผลของการค้นหาไปยังหน้าจอ ดังตัวอย่างของการค้นหาเลข 90 จากโปรแกรมของเรานี้

```
90 is in the list.
```

ถ้าหากว่าเราไม่ต้องการใช้ boolean เป็นตัวกำหนดการทำงานของ loop เราก็อาจกำหนดให้ loop ทำงานไปเรื่อย ๆ และใช้คำสั่ง break เป็นตัวกำหนดการหยุดก็ได้ โดยการเปลี่ยน code ให้เป็น

```
/* SeqSearch2.java */

import java.io.*;

class SeqSearch2 {
    public static void main(String[] args) {
        int[] list = {2, 3, 1, 56, 90};

        int number = 90;      //number to search
        int i = 0;            //loop index
        while(true) {
            if(number == list[i]) //if found
                break;            //break out of loop
            i++;                  //move to next number
        }                         //in the list

        if(i >= list.length)
            System.out.println(number + " is not in the list.");
        else
            System.out.println(number + " is in the list.");
    }
}
```

เรารู้ว่าถ้าข้อมูลที่ต้องการค้นหาไม่อยู่ใน list ตัวแปร i จะมีค่าเกินกว่าจำนวนของข้อมูลที่มีอยู่ใน array ดังนั้นการแสดงผลจะต้องใช้การเปรียบเทียบค่าของ i กับจำนวนของข้อมูลที่มีอยู่ใน array (แทนการเปรียบเทียบด้วย boolean ก่อนหน้านี้) วิธีการตรวจสอบแบบนี้ลดขั้นตอนการทำงานของ loop ลงไปได้หนึ่งครั้ง เพราะเราจะ break ออกจาก loop ทันที ที่เราค้นหาข้อมูลเจอ

การค้นหาแบบนี้ไม่ค่อยนิยมใช้กันมากเท่าไร ทั้งนี้ก็เพราะว่าจะใช้เวลาในการค้นหามาก ถ้าข้อมูลที่ต้องการค้นหาอยู่ด้านหลังสุดของ array ยิ่งถ้า array เก็บข้อมูลไว้เยอะการค้นหาก็จะใช้เวลามากยิ่งขึ้น

การค้นหาแบบหารสอง (Binary search)

การค้นหาในรูปแบบของ binary search จะทำการแบ่งข้อมูลทั้งหมดออกเป็นสองส่วนทุกครั้งที่มีการเปรียบเทียบข้อมูลที่ต้องการค้นหา (key) การค้นหาเริ่มต้นด้วยการเปรียบเทียบ key กับข้อมูล ณ ตำแหน่งที่อยู่กึ่งกลางของ array ถ้า key มีค่าน้อยกว่าข้อมูล ณ ตำแหน่งนี้การค้นหาก็ย้ายไปค้นทางซ้ายของข้อมูลนี้ และจะค้นทางขวาถ้า key มีค่ามากกว่าข้อมูลที่ตำแหน่งนี้ การค้นหาจะดำเนินไปจนกว่าจะค้นพบ นั่นก็คือ key มีค่าเท่ากับข้อมูลที่ตำแหน่งนั้น หรือไม่ก็ยุติการค้นถ้าไม่มีข้อมูลนั้นใน array

สิ่งที่สำคัญในการค้นหาแบบนี้ก็คือ ข้อมูลจะต้องมีการจัดเก็บแบบจัดเรียงจากน้อยไปหามาก เพราะถ้าไม่อยู่ในรูปแบบนี้แล้วการค้นหาก็จะทำไม่ได้

วิธีการค้นหาด้วย binary search จะลดจำนวนครั้งของการเปรียบเทียบลงไปทีละครึ่ง ทำให้การค้นหาใช้เวลาน้อยลง ตัวอย่างเช่น การค้นหาใน array ที่มีข้อมูลเท่ากับ 1024 ตัวจะใช้การเปรียบเทียบเพียง 10 ครั้ง ครั้งแรกลดลงเหลือ 512 ครั้งที่สองเหลือ 256 และครั้งต่อ ๆ ไปเหลือ 128 64 32 16 8 4 2 และ 1 จะเห็นว่าลดลงมาก ซึ่งเป็นจุดแข็งของการค้นหาของ binary search โปรแกรม BinSearch.java แสดงการค้นหาด้วยการใช้ binary search

```
/* BinSearch.java */

import java.io.*;

class BinSearch {
    public static void main(String[] args) {
        int[] list = {1, 2, 3, 4, 5, 6, 7, 8 ,9, 10};

        //generate key to search
        int key = (int)(Math.random() * 10 + 1);
        int low = 0;                 //lower index
        int high = list.length - 1;  //higher index
        int middle;                  //number at the middle (pivot)
        boolean found = false;       //indicate search result

        while(low <= high) {
            middle = (low + high) / 2;   //calculate pivot
            if(key == list[middle]) {    //if found - stop searching
                found = true;
                break;
            }
            else if(key < list[middle])  //key is in lower half
                high = middle - 1;
            else
                low = middle + 1;        //key is in upper half
        }
        if(found)
            System.out.println("Found " + key + "!");
        else
            System.out.println(key + " not found!");
    }
}
```

โปรแกรม BinSearch.java ของเราเริ่มต้นด้วยการกำหนดให้ array list มีข้อมูลเป็นจำนวนเท่ากับ 10 ตัว และกำหนดให้ข้อมูลที่ต้องการค้นหา (key) มาจากการสุ่ม หลังจากนั้นก็กำหนดให้ index หัว (low) และท้าย (high) ของ array เป็น 0 และ list.length – 1 ตามลำดับ เราจะใช้ while loop เป็นตัวเข้าหาข้อมูลใน array list ของเราด้วยการเปรียบเทียบค่าของ low กับ high ซึ่งถ้าเมื่อใดก็ตามที่ค่าของ low มากกว่าหรือเท่ากับ high ก็หมายความว่าการค้นหาของเราไม่ประสพผลสำเร็จ คือไม่มีข้อมูลนั้นอยู่ใน list เราก็จะยุติการทำงานทันที เราหาข้อมูลที่อยู่ตรงกลางด้วย

```
middle = (low + high) / 2;
```

และเปรียบเทียบการค้นหาที่ประสพผลสำเร็จด้วยประโยค

```
if(key == list[middle]) {
    found = true;
    break;
}
```

เราจะเปรียบเทียบและแบ่งข้อมูลออกเป็นสองส่วนด้วย

```
else if(key < list[middle])
    high = middle - 1;
else
    low = middle + 1;
```

เราจะย้าย index ที่อยู่หลังสุดให้เป็น middle – 1 ถ้าข้อมูลที่เป็น key น้อยกว่าข้อมูลที่อยู่ ณ ตำแหน่ง middle และจะย้าย low ไปอยู่ที่ middle + 1 ถ้า key มีค่ามากกว่าข้อมูล ณ ตำแหน่ง middle เราจะทำไปจนกว่าจะเจอข้อมูลที่ตำแหน่ง middle หรือไม่ก็หยุดการทำงานโดยไม่พบข้อมูลที่ต้องการค้นหาเลย

การเรียงลำดับ (sort) ข้อมูลใน array

มีวิธีการหลากหลายวิธีที่สามารถนำมาใช้ในการจัดเรียงข้อมูลใน array แต่เนื่องจากว่าเรายังต้องศึกษาถึงส่วนประกอบอื่น ๆ ของ Java ก่อนที่เราจะนำเอาวิธีการเหล่านั้นมาใช้ได้ ซึ่งหนังสือส่วนมากก็จะนำเอาวิธีการจัดเรียงข้อมูลไปไว้ในวิชาโครงสร้างข้อมูล (ซึ่งก็เป็นหนังสือเล่มต่อไปจากหนังสือเล่มนี้) ดังนั้นเราจะพูดถึงเฉพาะการนำเอา method การ sort ที่ Java มีให้มาใช้โดยตรงเพื่อให้การทำงานเป็นไปได้รวดเร็วยิ่งขึ้น

เราเพียงแต่ส่ง array ที่เราต้องการ sort ไปให้ method sort() ที่ Java มีให้ ดังนี้

```
Arrays.sort(list)
```

โดยที่ list เป็น array ที่เราต้องการ sort โปรแกรม Sort.java แสดงวิธีการเรียกใช้ sort จาก class Arrays

```
//Sort.java - recursive version

import java.lang.Integer;
import java.util.Arrays;

class Sort {
    public static void main(String[] args) {
        int[] list = new int[100];   //array with 100 ints
        int i;                       //loop index

        //populate list with random ints
        for(i = 0; i < 100; i++)
            list[i] = (int)(Math.random() * 100 + 1);
```

```
        //display 100 ints to screen - 20 per line
        for(i = 0; i < list.length; i++) {
            if(i % 20 == 0)
                System.out.println();
            System.out.print(list[i] + "\t");
        }
        System.out.println();

        Arrays.sort(list);    //calling sort from class Arrays

        //display again
        for(int j =0 ; j < list.length; j++) {
            if(j % 20 == 0)
                System.out.println();
            System.out.print(list[j] + "\t");
        }
        System.out.println();
    }
}
```

ยังมี method อีกหลายตัวใน class Arrays ที่เราสามารถเรียกใช้ได้ ทั้งนี้ก็ต้องดูว่าเรากำลังทำอะไรอยู่ ในที่นี้จะขอพูดอีกเพียง method เดียวคือ binarySearch() เพราะว่าเราได้พูดถึงวิธีการเขียน code ของ binary search มาก่อนหน้านี้แล้ว โปรแกรม BinarySearch.java แสดงถึงวิธีการเรียกใช้ method binarySearch() ที่ว่านี้

```
//BinarySearch.java - recursive version

import java.util.Arrays;

class BinarySearch {
    public static void main(String[] args) {
        int[] list = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

        int key = (int)(Math.random() * 10 + 1);//key to search for
        int found = Arrays.binarySearch(list, key);

        if(found >= 0)
            System.out.println("Found " + key + " at index " + found);
        else
            System.out.println(key + " is not in the list!");
    }
}
```

เราเรียก method binarySearch() ด้วย parameter 2 ตัวคือ (1) array ที่มีข้อมูลอยู่ – ในที่นี้คือ list และ (2) ข้อมูลที่ต้องการค้นหา – ซึ่งก็คือ key ในโปรแกรมด้านบนนี้ ประโยค

```
int found = Arrays.binarySearch(list, key)
```

ถ้าข้อมูลที่ต้องการค้นหาอยู่ใน list การเรียก method binarySearch() ก็จะได้ค่าของ index ที่ข้อมูลนั้นอยู่ส่งกลับมายังตัวแปร found แต่ถ้าหาไม่เจอค่าที่ส่งกลับจะมีค่าน้อยกว่า 0 เสมอ ดังนั้นการแสดงผลจึงต้องเปรียบเทียบค่าของ found กับค่าของ 0

เมื่อทดลอง run ดูผลลัพธ์ที่ได้คือ

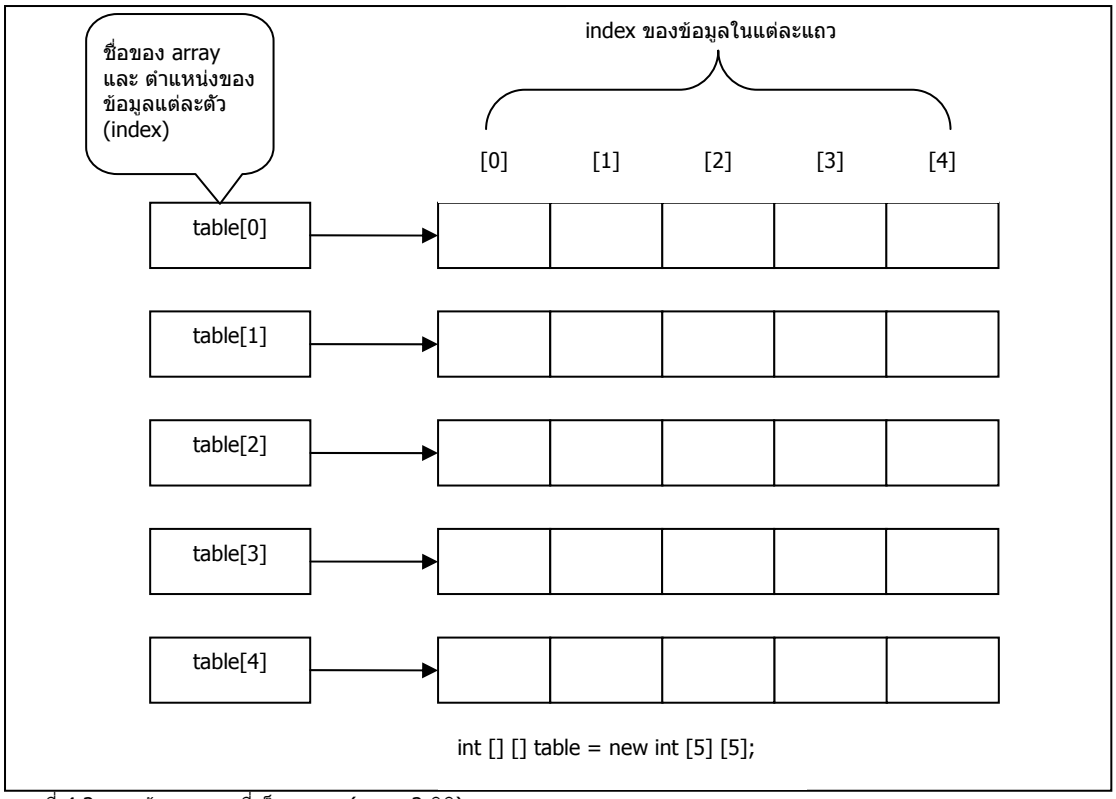
```
Found 3 at index 2
Found 1 at index 0
```

การสร้าง array ที่มีข้อมูลเป็น array (Array of arrays)

เราได้ใช้ array ที่มีข้อมูลเป็น primitive type จากตัวอย่างก่อนหน้านี้หลาย ๆ ตัวอย่าง เรามาลองดูการสร้าง array ที่มีข้อมูลเป็น array ดู เราจะเริ่มต้นด้วยการประกาศ ดังนี้

```
int[][] table = new int[5][5];
```

ซึ่งหมายถึงการประกาศให้ตัวแปร table เป็น array ที่มีข้อมูล 5 ตัวโดยที่แต่ละตัวนั้นเป็น array ที่เก็บข้อมูลที่เป็น int อยู่ 5 ตัว เราจะใช้ [][] สองตัวเป็นตัวบอกให้ Java รู้ว่าตัวแปรตัวนี้คือ array ที่ใช้เก็บ array โดยทั่วไปเรามักจะเรียก array ในลักษณะนี้ว่า array 2 มิติ ภาพที่ 4.2 แสดงถึงเนื้อที่ของ array table



ภาพที่ 4.2 การสร้าง array ที่เก็บ array (array 2 มิติ)

โปรแกรม Array9.java ที่เห็นด้านล่างนี้แสดงการใช้ array 2 มิติ โดยกำหนดให้มีขนาด 5 x 5 และกำหนดให้ข้อมูลมีค่าอยู่ระหว่าง 1 ถึง 10 ด้วยการสุ่มจาก random()

ความยาวหรือขนาดของ table หาได้จากค่าคงที่ที่เก็บไว้ length เช่นเดียวกันกับ array 1 มิติ ส่วนขนาดของข้อมูลที่อยู่ในแต่ละแถวหาได้จากค่าคงที่ในแถวนั้น เช่น

```
table[0].length
table[3].length
```

โปรแกรม Array9.java ใช้ table[i].length เป็นตัวกำหนดขนาดในแต่ละแถวโดยที่ i เป็น index ของแถวนั้น ๆ

```
/* Array9.java */

import java.io.*;
```

```
class Array9 {
    public static void main(String[] args) {
        int[][] table = new int[5][5];

        for(int i = 0; i < table.length; i++) {
            for(int j = 0; j < table[i].length; j++)
                table[i][j] = (int)(1 + Math.random() * 10);
        }

        for(int i = 0; i < table.length; i++) {
            for(int j = 0; j < table[i].length; j++)
                System.out.print(table[i][j] + "\t");
            System.out.println();
        }

    }
}
```

ผลลัพธ์ของการ run โปรแกรม Array9.java คือ

```
7   3   2   8   7
5   8   5   4   8
7   6   8   6   2
7   10  9   7   5
10  4   4   2   2
```

เรามาดูโปรแกรมตัวอย่างอีกตัวหนึ่งที่ประกาศและกำหนดค่าให้กับ array 2 มิติด้วยการกำหนดแบบอัตโนมัติจาก Java เอง

```
/* Array10.java */

import java.io.*;

class Array10 {
    public static void main(String[] args) {
        //declare 2-D array with specific data
        int[][] table = { {1, 2, 3},
                          {5, 6, 8},
                          {7, 4, 6},
                          {4, 8, 9} };

        for(int i =0 ; i < table.length; i++) {
            for(int j = 0; j < table[i].length; j++)
                System.out.print(table[i][j] + "\t");
            System.out.println();
        }

    }
}
```

เรากำหนดให้ table เป็น array 2 มิติที่มีจำนวนแถวเท่ากับ 4 และจำนวนข้อมูลในแต่ละแถวเท่ากับ 3 โดยการกำหนดนั้นเราต้องใส่ข้อมูลในแต่ละแถวในเครื่องหมาย {} และแบ่งข้อมูลที่อยู่แถวในแต่ละแถวด้วยเครื่องหมาย , ดังนี้

```
int[][] table = { {1, 2, 3},     /* แถว 0 */
                  {5, 6, 8},     /* แถว 1 */
                  {7, 4, 6},     /* แถว 2 */
```

```
                {4, 8, 9}       /* แถว 3 */
            };
```

เราสามารถที่จะให้ข้อมูลอยู่ในบรรทัดเดียวกันได้ เช่น

```
int[][] table = {{1, 2, 3}, {5, 6, 8}, {7, 4, 6}, {4, 8, 9}};
```

หลังจาก run โปรแกรมแล้ว ผลลัพธ์ที่ได้คือ

```
1   2   3
5   6   8
7   4   6
4   8   9
```

การใช้ array ที่มีจำนวนของข้อมูลในแต่ละแถวไม่เท่ากัน

เราสามารถที่จะกำหนดให้ข้อมูลในแต่ละแถวของ array มีจำนวนไม่เท่ากันได้ ดังตัวอย่างด้านล่างนี้

```
double [][] list = new double[5][];
```

ซึ่งเป็นการประกาศ array 2 มิติที่มีจำนวนแถวเท่ากับ 5 แต่จำนวนของข้อมูลในแต่ละแถวยังไม่ได้ถูกกำหนด เราอาจกำหนดให้ข้อมูลในแถว 0 มีจำนวนเท่ากับ 4 และข้อมูลในแถว 4 มีจำนวนท่ากับ 10 เป็นต้น ดังนี้

```
list[0] = new double[4];
list[4] = new double[10];
```

ส่วนข้อมูลในแถวที่เหลืออยู่ก็แล้วแต่ว่าเราอยากที่จะให้มีข้อมูลกี่ตัว หรือเราอาจกำหนดให้ข้อมูลในแต่ละแถวมีจำนวนเท่ากับค่าที่เพิ่มขึ้นตามค่าของแถว (index) เช่น

```
int m = 0;
while(m < 5) {
    list[m] = new double[m + 1];
    m++;
}
```

โปรแกรม Array11.java สร้าง array 2 มิติที่มีข้อมูลในแต่ละแถวเท่ากับค่าของ index ในแถวนั้น ๆ

```
/* Array11.java */

import java.io.*;

class Array11 {
    public static void main(String[] args) {
        double [][]list   = new double[5][];

        //allocate space for each row
        int m = 0;
        while(m < 5) {
            list[m] = new double[m + 1];
            m++;
        }

        for(int i = 0; i < list.length; i++) {
            for(int j = 0; j < list[i].length; j++)
                System.out.print(list[i][j] + "\t");
            System.out.println();
```

```
        }
    }
}
```

เมื่อ run ดูผลลัพธ์ที่ได้คือ

```
0.0
0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
```

การกำหนดขนาดของ array แบบนี้ทำให้การใช้เนื้อที่ในการเก็บข้อมูลในแต่ละแถวมีประสิทธิภาพมากขึ้น ทั้งนี้ก็เพราะว่าเราใช้เฉพาะเนื้อที่ที่จำเป็นต้องใช้เท่านั้น ถ้าเราใช้การกำหนดแบบเก่าเราต้องใช้หน่วยความจำเท่ากับ 8 x 25 = 200 byte แต่ถ้าเราใช้การกำหนดตามเท่าที่ใช้ เราก็จะใช้หน่วยความจำเพียงแค่ 8 x 15 = 120 byte

การสร้าง array ที่มีความจุของข้อมูลในแต่ละแถวไม่เท่ากันเอื้ออำนวยความสะดวกในการจองเนื้อที่ ทำให้การใช้หน่วยความจำมีประสิทธิภาพมากยิ่งขึ้น แต่การทำงานจะต้องมีการระมัดระวังเป็นพิเศษ เพราะจำนวนของข้อมูลในแต่ละแถวมีจำนวนไม่ท่ากัน ผู้อ่านต้องใช้ความละเอียดในการออกแบบ ถ้าต้องการใช้ array แบบนี้

ตัวอย่างการใช้ Array of arrays ในการหาค่า max ค่า min และค่าเฉลี่ย

```
//ArrayOfArrays.java

import java.io.*;

class ArrayOfArrays {
    public static void main(String[] args) {
        //inititial scores
        int [][]score = { {45, 87, 68, 57},
                          {98, 70, 58, 69},
                          {85, 42, 68, 82} };
        double average;           //average of scores
        int max, min, total = 0; //maximum, minimum, total of scores
        int count = 0;            //total scores

        max = score[0][0];        //assuming max score is at [0][0]
        min = score[0][0];        //assuming min score is at [0][0]
        //looping for 3 rows
        for(int i = 0; i < score.length; i++) {
            //looping for 4 columns
            for(int j = 0; j < score[i].length; j++) {
                if(score[i][j] > max) //find max score
                    max = score[i][j];
                if(score[i][j] < min) //find mim score
                    min = score[i][j];
                total += score[i][j]; //calculate total
                count++;              //number of scores
            }
        }
        //calculate average of scores
        average = (double)total / count;

        System.out.println("Maximum score is " + max);
        System.out.println("Minimun score is " + min);
        System.out.println("Average score is " + average);
```

```
    }
}
```

โปรแกรม ArrayOfArrays.java กำหนดให้ score เป็น array ที่เก็บข้อมูลเป็นจำนวนเท่ากับ 3 แถว แถวละ 4 ตัว เราจะหา score ที่เล็กที่สุด score ที่ใหญ่ที่สุด รวมไปถึงการคำนวณหาค่าเฉลี่ยของ score ทั้งหมด เราใช้ for – loop 2 ตัวในการเข้าหา array ของเราเหมือนเดิม พร้อมทั้งตรวจสอบว่า score ตัวไหนเป็น max และ score ตัวไหนเป็น min ด้วยการกำหนดให้ทั้งสองตัวแปรมีค่าเป็นข้อมูลตัวแรกสุดที่อยู่ใน array และจะทำการเปรียบเทียบข้อมูล ณ ตำแหน่งทั้งหมดใน array กับข้อมูลนี้ ประโยค

```
if(score[i][j] > max)
    max = score[i][j];
```

จะเปลี่ยนค่าของ max ให้มีค่าเท่ากับ score[i][j] ถ้าข้อมูลที่ตำแหน่งนี้มากกว่าค่าของ max และประโยค

```
if(score[i][j] < min)
    min = score[i][j];
```

จะเปลี่ยนค่าของ min ให้มีค่าเท่ากับ score[i][j] ถ้าข้อมูลที่ตำแหน่งนี้น้อยกว่าค่าของ min วิธีการหาค่าของ max และ min แบบนี้สามารถนำไปใช้ได้กับการหาค่า max และ min โดยทั่วไปได้โดยไม่ต้องสนใจว่าข้อมูลที่อยู่ภายใน array จะมี range อยู่ช่วงไหน ทั้งนี้เพราะเราใช้ข้อมูลตัวแรกสุดเป็นตัวตรวจสอบก่อน ทำให้การหาค่าของทั้ง max และ min เป็นไปได้โดยไม่ต้องกำหนดค่าให้กับตัวแปร max หรือ min ใด ๆ ทั้งสิ้น (ถ้าเราไม่ต้องการตรวจสอบเอง เราก็สามารถที่จะเรียกใช้ method max() และ min() ที่มีอยู่ใน class Math ได้ – ดูในบทก่อน ๆ) ในส่วนของการหาค่าเฉลี่ยก็เหมือนกับที่เราได้เคยทำมาแล้วก่อนหน้านี้

การใช้ array of arrays นั้นก็เหมือนกับที่เราใช้ตาราง หรือ table ในการเก็บข้อมูลต่าง ๆ หนังสือหลาย ๆ เล่มเรียก array of arrays ที่มีแค่ row และ column ว่า array 2 มิติ (two dimensional array) ส่วน array of arrays ที่มีข้อมูลเป็น array อีกเราก็มักจะเรียกว่า array 3 มิติ เป็นดังนี้ทุกครั้งที่มีการเพิ่มมิติให้กับ array แต่การมองเห็นหรือการใช้ก็คงมีน้อย ถ้าหากว่าจำนวนมิติที่ใช้มีมากกว่า 3 มิติ เพราะว่าคนเราชินกับการมองเห็นแบบ 3 มิติตลอดเวลา (เราจะทิ้งให้ผู้อ่านศึกษาในเรื่องของ array 3 มิติเอง)

การใช้ String

การใช้ string นั้นสามารถทำได้หลาย ๆ วิธี และวิธีแรกที่เราจะทำความรู้จักนั้นเป็นการใช้ array เป็นหลักในการทำงานกับ string นั้น ๆ เราคงไม่สามารถที่จะเรียก string ในลักษณะนี้ได้เต็มปากเต็มคำมากนัก เพราะว่าโดยความเป็นจริงแล้ว array ที่เราใช้ในการทำงานที่เกี่ยวข้องกับ String นั้นไม่ใช่ string โดยตรงแต่เป็น array ที่เก็บ char หลาย ๆ ตัวไว้ หรือที่เรียกกันว่า array of characters ซึ่งหนังสือหลาย ๆ เล่มก็มักจะเรียก string ว่า array of characters เราจะตรวจสอบวิธีการใช้ string แบบนี้พอสังเขป แล้วเราจะกลับมาพูดถึง String ของ Java อย่างละเอียดอีกทีหนึ่ง

Array of characters

การประกาศ string แบบนี้ทำได้ง่าย ๆ เหมือนกับที่เราประกาศใช้ array ที่เก็บข้อมูลชนิดอื่น เพียงแต่เราเปลี่ยนให้ array เก็บ char แทนเท่านั้น เช่น

```
char [] string = new char[20];
```

การประกาศด้านบนนี้เป็นการสร้าง array ที่ใช้เก็บ char จำนวนเท่ากับ 20 ตัว โปรแกรมตัวอย่างด้านล่างแสดงการใช้ array of characters

```
//ArrayOfChars.java

import java.io.*;

class ArrayOfChars {
    public static void main(String[] args) {
```

```
        //declare array of char
        char[] str = new char[26];
        int i = 0;

        //populate str with characters a to z
        for(char c = 'a'; c <= 'z'; c++) {
            str[i] = c;
            i++;
        }
        //print all characters to screen
        for(i = 0; i < 26; i++)
            System.out.print(str[i]);
        System.out.println();
    }
}
```

โปรแกรม ArrayOfChars.java เริ่มต้นด้วยการสร้าง array of chars จำนวน 27 ตัวจากประโยค

```
char[] str = new char[26];
```

หลังจากนั้นก็ใช้ for – loop ในการกำหนดค่าให้กับทุก ๆ ตำแหน่งของ str ด้วยการใช้ char เป็น index ของ loop เนื่องจากว่าการเพิ่มค่าให้กับ char นั้นที่จริงแล้วเป็นการเพิ่มค่าให้กับตัวเลขที่เป็นตัวแทนของตัวอักษรนั้น ๆ เช่น ถ้าเรากำหนดให้ char c = 'f' การเพิ่มค่าให้กับ c หนึ่งครั้งจะทำให้ c มีค่าเป็น 'g' ซึ่งเป็นตัวอักษรที่อยู่ถัดไปในตาราง ASCII เพราะฉะนั้นการใช้ for – loop นี้ก็ทำให้เกิดการกำหนดค่าให้กับ str ทุกตำแหน่ง

```
for(char c = 'a'; c <= 'z'; c++) {
    str[i] = c;
    i++;
}
```

ผู้อ่านคงเดาได้ว่าผลลัพธ์ของโปรแกรมตัวนี้คืออะไร → abcdefghijklmnopqrstuvwxyz

การใช้ array of characters นั้นไม่ค่อยจะสะดวกมากนัก ทั้งนี้เพราะเป็นการทำงานที่เกี่ยวกับ array เราอยากที่จะมี string จริง ๆ ที่เราสามารถที่จะทำกระบวนการต่าง ๆ ได้ เช่น บวก string 2 ตัวเข้าด้วยกัน หรือตรวจสอบความเท่ากัน ความเหมือนกันของ string อย่างนี้เป็นต้น และ Java ก็มี class String ให้เราเรียกใช้งานที่เกี่ยวข้องกับ string โดยตรง

String ใน Java

String ใน Java นั้นเป็น object ดังนั้นเราจึงจำเป็นต้องเรียกใช้ class String ในการสร้าง string เพื่อการใช้งานของเรา เช่น เราอาจต้องการเก็บชื่อของ user ไว้ในโปรแกรม หรือส่งข้อความที่เป็น string ไปให้ user อย่างนี้เป็นต้น

การประกาศใช้ String ก็ไม่ยาก ซึ่งทำได้ดังนี้

```
String str = new String("Java is fun");
```

หรือแบบนี้

```
String str = "Java is fun";
```

โดยเฉพาะการประกาศในประโยคที่สองนี้ มีผลเท่ากับประโยค 2 ประโยคต่อไปนี้

```
char[] str = {'J', 'a', 'v', 'a', ' ', 'i', 's', ' ', 'f', 'u', 'n'};
String str = new String(str);
```

จะเห็นได้ว่า string ก็คือ array of chars นั่นเอง บางครั้งการประกาศตัวแปร string โดยที่ไม่มีการกำหนดค่าให้กับ string นั้นเราก็ทำได้ด้วยการประกาศเหมือนกับการประกาศตัวแปรชนิดอื่น เช่น

```
String name;
```

ซึ่งมีความหมายว่าตัวแปร name เป็น string ที่ไม่ได้อ้างถึงค่าใด ๆ เลย และถ้าเรา compile โปรแกรมที่มีการประกาศแบบนี้จะมี error เกิดขึ้น วิธีการแก้ไขเพื่อไม่ให้ Java ฟ้องตอน compile เราต้องกำหนดค่า null ให้กับ string name ดังนี้

```
String name = null;
```

ซึ่งบอกให้ Java รู้ว่า string name ไม่ได้อ้างถึงข้อมูลใด ๆ เลย เราสามารถที่จะอ้างถึงข้อมูลใด ๆ ก็ได้หลังจากนั้น

ลองมาดูโปรแกรมตัวอย่างการทำงานกับ string

```
//StringOps.java

import java.io.*;

class StringOps {
    public static void main(String[] args) {
        String first = "Java ";
        String second = "Programming";
        String third = null;
        int percent = 100;

        third = first + second;
        System.out.println(third + " is " + percent + "% fun!");
    }
}
```

โปรแกรม StringOps.java ประกาศตัวแปรที่เป็น String 3 ตัวคือ first second และ third โดยกำหนดให้ first มีค่าเป็น "Java " second มีค่าเป็น "Programming" และ third ไม่มีค่าอ้างอิงถึง string ใด ๆ พร้อมกันนี้เราได้กำหนดให้ตัวแปร percent มีค่าเป็น 100 ประโยค

```
third = first + second;
```

ทำการรวม first และ second เข้าด้วยกัน ซึ่งทำให้ third มีค่าเป็น "Java Programming" และในประโยคที่ทำการแสดงผลไปหน้าจอนั้นเราได้เพิ่มการแสดงผลค่าของตัวแปร percent และ string คงที่ตัวอื่นเข้าไปด้วยเพื่อให้ผลลัพธ์เป็น

```
Java Programming is 100% fun!
```

เราสามารถที่จะรวมค่าของตัวแปรอื่นที่ไม่ใช่ string เข้ากับ string ได้ดังนี้ สมมติว่าเราเพิ่มตัวแปรต่อไปนี้

```
int percent = 100;
String is = " is ";
String fun = "%fun!";
```

และเรารวม String และ int เข้าด้วยกัน ด้วยประโยค

```
third = first + second + is + percent + fun;
```

เราก็จะได้ผลลัพธ์เหมือนกันกับที่เราได้ทำก่อนหน้านี้ สาเหตุที่ทำได้เพราะ Java จะเปลี่ยน int ให้เป็น string ก่อนที่จะรวมเอา string ทั้งหมดเข้าด้วยกัน

การเปรียบเทียบ String

เราไม่สามารถที่จะเปรียบเทียบ string 2 ตัวด้วยการใช้ == ได้เหมือนกับที่เราทำกับตัวแปรที่เป็น primitive type ได้ เช่น

```
first == third
```

เพราะว่าประโยคดังกล่าวจะเปรียบเทียบว่า string ทั้งสองตัวอ้างถึงที่เดียวกันหรือไม่ (ที่ที่เก็บ string ใด string หนึ่ง) เช่น ถ้าเรามี

```
String myString = "Motorcycle";
String yourString = myString;
```

ซึ่งทำให้ตัวแปร myString และ yourString อ้างถึงข้อมูลเดียวกัน (และจะให้ค่า true ถ้ามีการเปรียบเทียบด้วย ==) ไม่ใช่การเปรียบเทียบถึงข้อมูลที่ตัวแปรทั้งสองอ้างอยู่ เช่นถ้าเรามี string ต่อไปนี้

```
String keyboard = "from Thailand";
String kb = "from Thailand";
```

ถ้าเราเปรียบเทียบด้วยการใช้

```
if(keyboard == kb) …
```

เราจะได้ค่า false ถึงแม้ว่าตัวแปรทั้งสองจะมีค่าที่เหมือนกัน ทั้งนี้เพราะว่าเครื่องหมาย == จะเปรียบเทียบค่าอ้างอิง (ที่อยู่ หรือ address ในหน่วยความจำ) ของตัวแปรทั้งสองไม่ใช่ค่าที่ทั้งสองตัวแปรเก็บไว้ เราต้องใช้ method อื่นที่ Java มีให้ใน class String

โปรแกรมตัวอย่างของการเปรียบเทียบ string

```
//StringCompare.java

import java.io.*;

class StringCompare {
    public static void main(String[] args) {
        String first = "Java ";
        String second = "Programming";
        String third = "Java Programming";
        String forth;

        forth = first + second;  //forth is "Java Programming"
        if(third == forth)
            System.out.println("third and forth refer to
                                the same string");
        else
            System.out.println("third and forth do not refer to
                                the same string");

        first = forth; //first and forth refer to the same string
        System.out.println("first is " + first);
        System.out.println("forth is " + forth);
        if(first == forth)
            System.out.println("first == forth is true");
        else
```

```
            System.out.println("first == forth is false");
    }
}
```

โปรแกรม StringCompare.java แสดงถึงการเปรียบเทียบ string ด้วยการใช้เครื่องหมาย == ถ้าเราสังเกตให้ดีจะเห็นว่า string third และ forth มีค่าที่เหมือนกัน (string ที่ประกอบไปด้วย char ที่เหมือนกันหมด) เมื่อ run ดูผลลัพธ์ที่ได้คือ

```
third and forth do not refer to the same string
```

ซึ่งบอกให้เรารู้ว่าการใช้เครื่องหมาย == เปรียบเทียบความเหมือนหรือความแตกต่างของ string ไม่ได้ เช่นเดียวกันกับประโยคเปรียบเทียบถัดมาในโปรแกรม เรากำหนดให้ first อ้างถึงข้อมูลที่เดียวกันกับ forth และเมื่อเรา run ผลลัพธ์ที่ได้ก็เหมือนกับที่เราได้คาดหวังไว้คือ

```
first is Java Programming
forth is Java Programming
first == forth is true
```

ถ้าเราต้องการที่จะเปรียบเทียบ string จริง ๆ เราต้องใช้ method equals() และ method compareTo() ดังโปรแกรมตัวอย่างต่อไปนี้

```
//StringCompare1.java

import java.io.*;

class StringCompare1 {
    public static void main(String[] args) {
        String first = "Java ";
        String second = "Programming";
        String third = "Java ProgramminG";
        String forth;

        forth = first + second;  //forth is "Java Programming"
        if(third.equals(forth))
            System.out.println("third equals forth");
        else
            System.out.println("third not equals forth");

        first = forth; //first and forth refer to the same string
        if(first.equals(forth))
            System.out.println("first equals forth");
        else
            System.out.println("first not equals forth");
    }
}
```

เราได้เปลี่ยนให้ third มีค่าเป็น "Java ProgramminG" เพื่อที่จะแสดงให้เห็นว่าในการเปรียบเทียบนั้น char ทุกตัวจะต้องเหมือนกัน ผลลัพธ์ที่เราได้จากการ run คือ

```
third not equals forth
first equals forth
```

เราใช้ประโยค third.equals(forth) ในการเปรียบ third และ forth ว่ามีค่าที่เหมือนกัน (เท่ากัน) ซึ่งถ้าเท่ากันจริงผลลัพธ์ที่ได้จากการเปรียบจะมีค่าเป็น true และจะให้ค่า false ถ้าไม่เท่ากัน

การเปรียบเทียบ string 2 ตัวเพื่อตรวจสอบความเท่ากันนั้นเราจะเอาตัวไหนอยู่ด้านหน้าก็ได้ เช่น

third.equals(forth) หรือ forth.equals(third) ก็ได้

method equals() จะตรวจสอบตัวอักษรทุกตัวที่อยู่ใน string โดยให้ความสนใจในเรื่องของตัวอักษรว่าเป็นตัวอักษรตัวเล็กหรือตัวใหญ่ การตรวจสอบจะให้ค่าที่เป็นจริงถ้าเหมือนกันทั้งรูปร่างและขนาด เช่น g หรือ G ไม่ใช่ตัวอักษรตัวเดียวกัน การตรวจสอบจะเป็น false

ในการเปรียบเทียบบางครั้งเราก็ไม่สนใจว่าตัวอักษรที่อยู่ภายใน string นั้นเป็นตัวเล็กหรือตัวใหญ่ ซึ่งถ้าเราต้องการเพียงแต่ตรวจสอบถึงความเหมือนทางรูปร่างเราก็ต้องใช้ method equalsIgnoreCase() ในการเปรียบเทียบ string นั้น ๆ

ถ้าเราเปลี่ยนการเปรียบเทียบในโปรแกรมตัวอย่างก่อนหน้านี้จากการใช้ equals() มาเป็น equalsIgnoreCase() ดังนี้

```
if(third.equalsIgnoreCase(forth))
    System.out.println("third equals forth");
else
    System.out.println("thrid not equals forth");
```

เราก็จะได้ผลลัพธ์เป็น

```
third equals forth
```

การเปรียบเทียบความไม่เท่ากัน (มากกว่า หรือ น้อยกว่า) ของ string

หลาย ๆ ครั้งเราต้องตรวจสอบถึงความมากกว่า หรือ น้อยกว่าของ string เพื่อให้การจัดเก็บ string นั้น ๆ อยู่ในรูปแบบที่มีการจัดเรียง ไม่ว่าจะเป็นจากน้อยไปหามาก หรือมากไปหาน้อย ในการตรวจสอบว่า string ตัวไหนมากกว่าหรือน้อยกว่ากันนั้น เราจะใช้ method compareTo() เช่น

```
myString.compareTo(yourString);
```

สมมติว่า myString มีค่าเป็น "lovely bike" และ yourString มีค่าเป็น "lovely kite" การเปรียบเทียบด้วย compareTo() จะให้ค่าที่เป็นบวก ทั้งนี้ก็เพราะว่า ตัวอักษร 'k' ใน yourString นั้นมีค่ามากกว่าตัวอักษร 'b' ใน myString เรามาดูโปรแกรมตัวอย่างการใช้ method compareTo() กันดีกว่า

```
//StringCompareTo.java

import java.io.*;

class StringCompareTo {
    public static void main(String[] args) {
        String first = "lovely bike";
        String second = "lovely kite";
        String third = "lovely";

        //compare first and second
        if(first.compareTo(second) > 0)
            System.out.println("first is greater than second");
        else
            System.out.println("second is greater than first");

        //compare first and third
        if(first.compareTo(third) > 0)
            System.out.println("first is greater than third");
        else
            System.out.println("third is greater than first");

        String a = "A";
```

```
        String z = "Z";
        String str1 = "AAA";
        String str2 = "AAA";

        //compare a and z
        if(a.compareTo(z) < 0)
            System.out.println(a + " is less than " + z);

        //compare str1 and str2
        if(str1.compareTo(str2) == 0)
            System.out.println(str1 + " equals " + str2);
    }
}
```

ผลลัพธ์ที่เราได้จากการ run คือ

```
second is greater than first
first is greater than third
A is less than Z
AAA equals AAA
```

โปรแกรมของเราเปรียบเทียบ first กับ second ด้วยการตรวจสอบค่าที่ได้จากการเปรียบเทียบกับ 0 ซึ่งถ้า first มากกว่า second ประโยคนี้ก็จะเป็น true แต่เนื่องจากว่า first นั้นไม่มากกว่า second การเปรียบเทียบของเราจึงให้ผลลัพธ์เป็น false Java จึงประมวลผลประโยคที่อยู่ใน else (ส่งผลลัพธ์ไปหน้าจอ) – การประมวลผลประโยคอื่น ๆ ก็คล้ายกัน

โดยทั่วไปเราจะเปรียบเทียบค่าที่ได้จาก method compareTo() กับ 0 เสมอเพราะจะทำให้เราได้ค่า true หรือ false ที่สามารถนำไปใช้ในการเลือกประมวลผลประโยคต่าง ๆ ได้ตามที่เราต้องการ

การเปรียบเทียบความเหมือนและเท่ากันก็ทำได้ด้วยการใช้เครื่องหมาย == ดังเช่นประโยคสุดท้ายของโปรแกรม

```
if(str1.compareTo(str2) == 0)
    System.out.println(str1 + " equals " + str2);
```

ในการเปรียบเทียบด้วย method compareTo() นี้ค่าที่ส่งกลับมาเป็นไปได้เพียงแค่ 3 ค่า คือ -1, 0, และ 1 เท่านั้น โดยที่

-1 หมายถึง string ที่อยู่ทางซ้าย น้อยกว่า string ที่อยู่ทางขวา (string ที่เป็น parameter)
0 หมายถึง string ทั้งสองนั้นเท่ากัน
1 หมายถึง string ที่อยู่ทางซ้าย มากกว่า string ที่อยู่ทางขวา

เช่น ถ้าเรามีประโยค string1.compareTo(string2);

string1 หมายถึง string ที่อยู่ทางซ้าย
string2 หมายถึง string ที่อยู่ทางขวา หรือ string ที่เป็น parameter

การเข้าหาตัวอักษรที่อยู่ใน string

เราสามารถที่จะเข้าหาตัวอักษรต่าง ๆ ที่อยู่ใน string ได้ด้วยการใช้ method ต่าง ๆ ดังที่ได้แสดงไว้เป็นตัวอย่างในโปรแกรม StringChars.java

```
//StringChars.java

import java.io.*;
```

```
import java.lang.Character;

class StringChars {
    public static void main(String[] args) {
        //setting up constant string
        String string = "Learn to write programs in Java doesn't take"
        + " as long as I think it would. But to fully understand"
        + " it is a time consuming task.";

        int charCount = 0;        //count of letters
        int vowelCount = 0;       //count of vowels
        int lowerCaseCount = 0;   //count of lowercase letters
        int upperCaseCount = 0;   //count of uppercase letters
        int spaceCount = 0;       //count of white spaces

        //check all letters
        for(int i = 0; i < string.length(); i++) {
            char ch = string.charAt(i);
            //check if it's a letter
            if(Character.isLetter(ch))
                charCount++;
            //check if it's an uppercase
            if(Character.isUpperCase(ch))
                upperCaseCount++;
            //check if it's a lowercase
            if(Character.isLowerCase(ch))
                lowerCaseCount++;
            //check if it's a white space
            if(Character.isWhitespace(ch))
                spaceCount++;

            //convert it to lowercase
            ch = Character.toLowerCase(ch);
            //check if it's a vowel
            if(ch == 'a' || ch == 'e' || ch == 'i' ||
                            ch == 'o' || ch == 'u')
                vowelCount++;
        }

        System.out.println(charCount + " characters counted.");
        System.out.println(upperCaseCount +
                           " uppercase letters counted.");
        System.out.println(lowerCaseCount +
                           " lowercase letters counted.");
        System.out.println(vowelCount + " vowels counted.");
        System.out.println(charCount - vowelCount +
                           " consonants counted.");
        System.out.println(spaceCount + " spaces counted.");
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
99 characters counted.
4 uppercase letters counted.
95 lowercase letters counted.
37 vowels counted.
62 consonants counted.
```

```
24 spaces counted.
```

การทำงานของโปรแกรมเริ่มต้นด้วยการกำหนดให้ตัวแปร string มีค่าเป็นค่าคงที่ที่อยู่ใน " " เราจะเข้าหาตัวอักษรแต่ละตัวด้วย for – loop และดึงเอาตัวอักษรออกมาจาก string แต่ละตัวด้วยคำสั่ง

```
char ch = string.charAt(i)
```

ซึ่งจะดึงเอาตัวอักษร ณ ตำแหน่ง index i นั้น ๆ มาเก็บไว้ที่ตัวแปร ch หลังจากนั้นก็ตรวจสอบด้วย method ต่าง ๆ ที่มีอยู่ใน class Character เช่น

Character.isLetter(ch)	ให้ค่า true ถ้าเป็นตัวอักษร
Character.isUpperCase(ch)	ให้ค่า true ถ้าเป็นตัวอักษรตัวใหญ่
Character.isLowerCase(ch)	ให้ค่า true ถ้าเป็นตัวอักษรตัวเล็ก
Character.isWhitespace(ch)	ให้ค่า true ถ้าเป็นช่องว่าง (เคาะแป้น หรือ tab)

พร้อมทั้งเพิ่มค่าให้กับตัวแปรทั้งหลายที่เป็นที่เก็บจำนวนครั้งของตัวอักษรต่าง ๆ ที่ปรากฏใน string หลังจากนั้นก็เปลี่ยนตัวอักษรให้เป็นตัวเล็กเพื่อใช้ตรวจสอบว่าเป็นสระในภาษาอังกฤษหรือไม่ สาเหตุที่เปลี่ยนเพราะเราไม่อยากที่จะเสียเวลาในการตรวจสอบสระตัวเล็กทีหนึ่ง และตัวใหญ่อีกทีหนึ่ง อย่างไรก็ตามทั้งสองก็เป็นสระอยู่เสมอ

```
ch = Character.toLowerCase(ch);
if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
    vowelCount++;
```

ยังมี method อีกมากมายใน class Character และ class String ที่เราสามารถนำมาใช้ในการทำงานที่เกี่ยวข้องกับ string ตัวอย่างต่อไปนี้เป็นการค้นหาคำ (sub string) ที่อยู่ใน string ด้วยการใช้ method indexOf()

```
//IndexOf.java

import java.io.*;
import java.lang.Character;

class IndexOf {
    public static void main(String[] args) {
        //setting up constant string
        String string = "Learn to write programs in Java doesn't take"
          + " as long as I think it would. But to fully understand"
          + " it is a time consuming task.";

        int itCount = 0;  //count of it
        String it = "it"; //string to seach for

        //search for "it"
        int index = string.indexOf(it);
        while(index >= 0) {
            itCount++;
            //move to first letter after last "it"
            index += it.length();
            //look for next "it"
            index = string.indexOf(it, index);
        }

        System.out.println("it appeared " + itCount + " times.");
    }
}
```

ผลลัพธ์ของโปรแกรมคือ

```
it appeared 3 times.
```

โปรแกรมของเรากำหนดให้ index มีค่าเป็นตำแหน่งที่ "it" ปรากฏอยู่ใน string ด้วยคำสั่ง

```
int index = string.indexOf(it)
```

เมื่อการค้นหาประสพผลสำเร็จเราก็จะใช้ while – loop ในการค้นหาต่อไป แต่ก่อนที่เราจะดำเนินการนั้น เราจะเพิ่มค่าหนึ่งค่าให้กับ itCount ก่อน หลังจากนั้นเราจะเลื่อน index ไปอยู่ในตำแหน่งของตัวอักษรตัวแรกที่อยู่หลัง "it" ด้วยคำสั่ง index += it.length() เนื่องจากว่า method indexOf() จะส่งค่าของตำแหน่งของตัวอักษรตัวแรกของ "it" มาให้ เสร็จแล้วก็ทำการค้นหาต่อไปด้วยคำสั่ง

```
index = string.indexOf(it, index)
```

ทำการค้นหาไปจนกระทั่งหมด string เราก็ออกจาก while – loop

ในการใช้ indexOf() ค้นหานั้นเราต้องเปรียบเทียบค่าที่ส่งกลับมากับ 0 เพราะถ้าหาไม่เจอค่าของ index ก็จะเป็น -1 ทำให้เราหลุดออกจาก while – loop ได้

Method indexOf() นั้นมีอยู่สองตัวที่รับเงื่อนไข (parameter) ไม่เหมือนกัน ดังที่เห็นในโปรแกรม ตัวแรกรับเพียงค่าเดียวคือ sub string ที่ต้องการค้นหา ส่วนตัวที่สองรับค่าสองค่าคือ sub string ที่ต้องการค้นหา และ จุด (index) เริ่มต้นของการค้นหา ซึ่งถ้าสังเกตให้ดีเราใช้ indexOf() ตัวแรกในการค้นหาครั้งแรก และตัวที่สองในการค้นหาตัวต่อ ๆ ไป ทั้งนี้ก็เพราะว่าครั้งแรกสุดเราไม่รู้ว่าค่าของ index แต่ในครั้งที่สองนั้นเราได้ค่าของ index แล้ว

ผู้อ่านต้องจำไว้เสมอว่า indexOf() จะส่งค่า -1 มาให้ถ้าไม่มี sub string ใน string นั้น ๆและจะส่งค่าของตำแหน่งที่ sub string นั้นอยู่ถ้ามี sub string ที่ว่า

ต่อไปเราลองมาดูโปรแกรมที่ทำการดึงเอากลุ่มของตัวอักษร (คำ) หรือที่เรียกว่า sub string ออกจาก string ดู

```
//SubString.java

import java.io.*;
import java.lang.Character;

class SubString {
    public static void main(String[] args) {
        //setting up constant string
        String string = "Learning Java";

        String str1 = string.substring(0);      //the whole string
        String str2 = string.substring(9);      //string: "Java"
        String str3 = string.substring(0, 8);   //string: "Learning"

        System.out.println("str1 > " + str1);
        System.out.println("str2 > " + str2);
        System.out.println("str3 > " + str3);
    }
}
```

Java มี method substring() ให้เราใช้อยู่สองตัว คือ substring() ที่ไม่มี parameter และ substring() ที่มี parameter อยู่สองตัว โดยที่ตัวแรกเป็น index เริ่มต้นของ sub string และ parameter ตัวที่สองเป็น index ตัวสุดท้ายที่อยู่ใน sub string นั้นแต่ไม่นับตัวมันเอง ดังที่ได้แสดงไว้ในโปรแกรม

```
String str2 = string.substring(9)
```

หมายความถึงการเริ่มนับที่ index = 9 ไปจนถึงตัวสุดท้ายใน string นั้น คือ เริ่มที่ตัวอักษร 'J' ไปจบที่ตัวอักษร 'a' ซึ่งคือคำว่า "Java" ส่วน

```
String str3 = string.substring(0, 8)
```

หมายถึงการเริ่มนับที่ index = 0 ไปจนถึงตัวที่ 8 คือตั้งแต่ตัวอักษร 'L' ไปจนถึงตัวอักษร ' ' (space) แต่ไม่นับ space ดังนั้น sub string ที่เราได้ก็คือ "Learning"

เราสามารถที่จะดึงเอา sub string ออกจาก string ที่เป็นค่าคงที่ได้ เช่น

```
String sub = "Basic idea about Java".substring(6)
```
จะให้ `"idea about Java"`

หรือ

```
String sub = "Basic idea about Java".substring(6, 10)
```
จะให้ `"idea"`

ตารางที่ 4.1 แสดงถึง method ต่าง ๆ ของ class String

ตาราง 4.1 method ของ class String	
method	ความหมาย
length()	ความยาวของ String (จำนวนตัวอักษร)
indexOf(String)	ตำแหน่งเริ่มต้นที่ String ตัวนี้อยู่ ถ้าไม่มีจะเป็น -1
indexof(String, start index)	ตำแหน่งเริ่มต้นที่ String ตัวนี้อยู่ตามจุดที่กำหนด ถ้าไม่มีจะเป็น -1
lastIndexof(String)	ตำแหน่งสุดท้ายของ String ตัวนี้ เท่ากับ -1 ถ้าไม่มี
lastIndexof(String, start index)	ตำแหน่งสุดท้ายของ String ตัวนี้ตามจุดที่กำหนด เท่ากับ -1 ถ้าไม่มี
trim()	String ที่ไม่มี space อยู่ทั้งด้านหน้า และหลัง
substring(start index)	ค่าของ String ณ ตำแหน่งที่กำหนดจนจบ
substring(start index, end index)	ค่าของ String ณ ตำแหน่งที่กำหนดทั้งสอง
replace(old char, new char)	เปลี่ยน char ทุกตัวด้วย char ตัวใหม่ที่กำหนด
charAt(index)	ค่าตัวอักษร ณ ตำแหน่งที่กำหนด
equals(String)	true ถ้า String ทั้งสองตัวเท่ากัน
equalsIgnoreCase(String)	true ถ้า String ทั้งสองตัวเท่ากัน ไม่สนใจ case
startsWith(String)	true ถ้า String เริ่มต้นด้วย String ที่กำหนด
startsWith(String, start index)	true ถ้า String เริ่มต้นด้วย String ที่กำหนด ณ ตำแหน่งที่กำหนด
endsWith(String)	true ถ้า String จบด้วย String ที่กำหนด

การใช้ class StringBuffer ในการสร้าง string

แทนที่จะใช้ class String เราก็สามารถที่จะใช้ class StringBuffer แทนได้ การทำงานก็ไม่แตกต่างกันมากมายนัก เช่น สมมติว่าเราจะสร้าง string สักตัวหนึ่งเราก็ทำได้ ดังนี้

```
StringBuffer string = new StringBuffer("Basic String Operations");
```

เราสร้าง object หนึ่งตัวชื่อ string จาก class StringBuffer โดยกำหนดค่าเบื้องต้นเป็น "Basic String Operations" เราไม่สามารถที่จะสร้าง string เหมือนกับที่เราสร้างด้วย class String ได้ เราต้องจองเนื้อที่ให้กับตัวแปรด้วยคำสั่ง new ก่อน หลังจากนั้นก็กำหนดค่าให้กับตัวแปรนี้ เราอาจประกาศด้วยการประกาศแบบนี้

```
StringBuffer string = null;
string = new StringBuffer("Just another way to declare");
```

แต่ไม่ใช่แบบนี้

```
StringBuffer string = "Just another way to declare";
```

ข้อดีของการใช้ StringBuffer ก็คือ เราสามารถที่จะเปลี่ยนแปลง string ที่เราสร้างขึ้นได้ (object จาก class String ที่เราใช้ก่อนหน้านี้ทำไม่ได้) และเรายังสามารถที่จะกำหนดขนาดของ string จาก class StringBuffer ได้ตามความพึงพอใจของเรา ลองมาดูโปรแกรมตัวอย่างการใช้ object จาก class StringBuffer กัน

```
//StringBufferTest.java

import java.io.*;

class StringBufferTest {
    public static void main(String[] args) {
        //setting up constant string
        StringBuffer string = new StringBuffer("fish");
        System.out.println("string is " + string);

        //append "er" after "fish"
        string.append("er");
        System.out.println("string is " + string);

        //append "man" after "fisher" using method insert()
        string.insert(string.length(), "man");
        System.out.println("string is " + string);
    }
}
```

โปรแกรม StringBufferTest.java แสดงการสร้าง object จาก class StringBuffer ด้วยคำสั่ง

```
StringBuffer string = new StringBuffer("fish")
```

จะทำให้ค่าของ `string` เป็น

f	i	s	h

หลังจากที่เราแสดงข้อมูลที่อยู่ใน string แล้วเราก็นำเอา string "er" มาเชื่อมเข้ากับ string "fish" ด้วยคำสั่ง

```
string.append("er")
```

จะทำให้ค่าของ `string` เป็น

f	i	s	h	e	r

ทำให้ข้อความที่อยู่ใน string เปลี่ยนเป็น "fisher" และหลังจากที่เราแสดงผลที่ได้ใหม่นี้ไปยังหน้าจอ เราก็เปลี่ยนข้อมูลใน string ใหม่ด้วยการนำเอา string "man" มาเชื่อมเข้ากับ string ที่เราได้ก่อนหน้านี้ด้วยการใช้ method insert() ดังนี้

```
string.insert(string.length(), "man")
```

จะทำให้ค่าของ `string` เป็น

f	i	s	h	e	r	m	a	n

เราเรียก insert() ด้วย parameter 2 ตัวคือ 1). ความยาวของ string เดิมซึ่งก็คือจุดเริ่มต้นของการเชื่อม และ 2). string ตัวใหม่ที่เราต้องการเชื่อม คือ "man" เมื่อ run ดูแล้วผลลัพธ์ที่ได้คือ

```
string is fish
string is fisher
string is fisherman
```

เราสามารถที่จะเปลี่ยนขนาดของ object ที่มาจาก class StringBuffer ได้ดังนี้

```
string.setLength(6)
```

ซึ่งจะทำให้ข้อมูลที่อยู่ใน string เปลี่ยนไป หลังจากที่เราเปลี่ยนขนาดของ string และเพิ่มประโยคสำหรับการแสดงผลในโปรแกรมของเราให้เป็น

```
System.out.println("string after set length is " + string)
```

ผลลัพธ์ที่เราจะได้จากการ run ก็จะกลายเป็น

```
string after set length is fisher
```

เราไม่ได้ถูกจำกัดให้ใส่ข้อมูลที่เป็น string เท่านั้นใน object ที่มาจาก class StringBuffer เราสามารถที่จะเรียกใช้ append() หรือ insert() กับข้อมูลชนิดอื่น ๆ ได้ เช่นโปรแกรมตัวอย่างต่อไปนี้

```
//StringBufferTest1.java

import java.io.*;

class StringBufferTest1 {
    public static void main(String[] args) {
        //setting up constant string
        StringBuffer string = new StringBuffer();
        string.append(2);
        System.out.println("string is " + string);

        //append " fisher" after 2
        string.append(" fisher");
        System.out.println("string is " + string);

        //append "men" after "fisher" using method insert()
        string.insert(string.length(), "men");
        System.out.println("string is " + string);

        string.append(" with ");
        System.out.println("string is " + string);

        string.append(100);
        string.append(" fish.");
        System.out.println("string is " + string);
    }
}
```

เราไม่ได้ทำอะไรมากมายนักเพียงแต่เรียกใช้ method append() ด้วยค่าที่เป็น int เช่น string.append(2) หรือ string.append(100) เป็นต้น Java ยังยอมให้เราเชื่อมข้อมูลที่มีความหลากหลายของชนิดของข้อมูลเข้าสู่ object ที่มาจาก class StringBuffer ได้โดยไม่จำกัดจำนวน ผลลัพธ์ของโปรแกรมหลังจากการปรับปรุงคือ

```
string is 2
string is 2 fisher
string is 2 fishermen
string is 2 fishermen with
string is 2 fishermen with 100 fish.
```

ส่วนหนึ่งของ class StringBuffer ที่น่าสนใจก็คือ การกำหนดขนาดให้กับ buffer ที่รองรับข้อมูลใน object ต่าง ๆ ที่เกิดจาก class นี้ นั่นก็คือ Java จะกำหนดขนาดของ buffer ให้เท่ากับข้อมูลที่มีอยู่ใน buffer บวกกับอีก 16 ตัวอักษรเสมอ เช่น ถ้าเราสร้าง object

```
StringBuffer name = new StringBuffer("Chiang Mai");
```

ซึ่งมีจำนวนตัวอักษรทั้งหมด (นับช่องว่างด้วย) เท่ากับ 10 ตัว Java จะจองเนื้อที่ให้มีความจุเท่ากับ 26 ตัว เราสามารถตรวจสอบความจุของ buffer ด้วยการเรียกใช้ method capacity() ได้ เช่น ถ้าเราสร้างประโยคต่อไปนี้

```
StringBuffer s = new StringBuffer("Chaing Mai");
System.out.println("Length of " + s + " is " + s.length());
System.out.println("Capacity of " + s + " is " + s.capacity());
```

หลังจากการประมวลผล เราจะได้ผลลัพธ์ดังนี้

```
Length of Chaing Mai is 10
Capacity of Chaing Mai is 26
```

ตัวอย่างต่อไปนี้คงเป็นตัวอย่างสุดท้ายที่เราจะแสดงให้เห็นถึงการใช้ method อีกตัวหนึ่งในการเปลี่ยนแปลงข้อมูลที่อยู่ใน string ให้อยู่ในรูปแบบของการกลับหัวกลับหาง (reverse)

```
//ReverseString.java

import java.io.*;

class ReverseString {
    public static void main(String[] args) {
        //setting up a constant string
        StringBuffer string = new StringBuffer("I love Chaing Mai");
        //create an empty string
        StringBuffer revStr = new StringBuffer();

        revStr.append(string);   //append string to revStr
        revStr.reverse();     //reverse it

        System.out.println("Reverse of [" + string + "] is ["
                                          + revStr + "]");
    }
}
```

ผลลัพธ์ของการ run คือ

```
Reverse of [I love Chaing Mai] is [iaM gniahC evol I]
```

หลังจากที่สร้าง string ด้วยค่า "I love Chiang Mai" แล้วเราก็สร้าง object อีกตัวหนึ่งโดยไม่มีการอ้างถึงค่าใด ๆ ทั้งสิ้นด้วยคำสั่ง StringBuffer revStr = new StringBuffer() หลังจากนั้นเราเชื่อม string เข้ากับ revStr ด้วยการใช้ method append() เช่นเดียวกับตัวอย่างอื่น ๆ ที่เราได้ทำมาก่อนหน้านี้ เสร็จจากการเชื่อมเราก็ใช้คำสั่ง revStr.reverse() เพื่อทำการ reverse ข้อมูลที่อยู่ใน revStr ที่เหลืออยู่ก็คือการส่งผลลัพธ์ของการ reverse ไปยังหน้าจอ

ผู้อ่านอาจสงสัยว่าทำไมเราไม่สร้าง object ด้วยการส่ง string ไปให้กับ revStr ในตอนแรกที่เราสร้าง revStr เช่น

```
StringBuffer revStr = new StringBuffer(string)
```

เราทำไม่ได้เพราะ Java ได้สร้าง method ในการสร้าง object จาก class StringBuffer ไว้เพียงสามตัว คือ

1. StringBuffer() โดยไม่ใส่เงื่อนไขใด ๆ
2. StringBuffer(int len) ต้องใส่ขนาดของ buffer
3. StringBuffer(String str) ต้องใส่ object จาก class String เท่านั้น

ยังมี method อีกมากมายที่เราสามารถนำมาใช้ในงานที่เกี่ยวข้องกับ string แต่เราคงไม่สามารถพูดในที่นี้ได้หมด จึงเพียงแต่หวังว่าผู้อ่านจะทำการค้นคว้าต่อไปเพื่อความเข้าใจในเรื่องที่เกี่ยวข้องกับ string ต่อไป ตารางที่ 4.2 แสดงถึง method ต่าง ๆ ของ class StringBuffer

ตาราง 4.2 method ของ class StringBuffer

method	ความหมาย
capacity()	ความจุของ StringBuffer ตัวนี้
length()	ความยาวของ StringBuffer ตัวนี้
setLength(new length)	กำหนดความยาวของ StringBuffer ด้วยความยาวใหม่
append(value)	เชื่อมค่าเข้าทางด้านหลังของ StringBuffer ตัวนี้
insert(index, value)	เพิ่มค่าเข้าสู่ StringBuffer ณ ตำแหน่งที่กำหนด
replace(start index, end index, String)	เปลี่ยนค่าตามจุดที่กำหนด ด้วย String
delete(start index, end index)	ลบค่าออกตามจุดที่กำหนด
deleteCharAt(index)	ลบตัวอักษรตามจุดที่กำหนด
setCharAt(index, character)	เปลี่ยนตัวอักษร ณ ตำแหน่งที่กำหนด
charAt(index)	ค่าตัวอักษร ณ ตำแหน่งที่กำหนด
substring(index)	ค่า substring ตามจุดเริ่มต้นที่กำหนดจนจบ String
substring(start index, end index)	ค่า substring ตามจุดที่กำหนดเท่านั้น
toString()	object ที่มีค่าของ String ที่อยู่ใน StringBuffer
reverse()	สลับตัวอักษรทุกตัวใน StringBuffer

การกำหนดให้ข้อมูลของ array เป็น string (Array of Strings)

เนื่องจากว่า string เป็น object ดังนั้นเราจึงสามารถที่จะเก็บ string ไว้ใน array ได้ การจัดเก็บ string ไว้ใน array ก็ไม่ยุ่งยาก ก็คล้าย ๆ กับที่เราเก็บข้อมูลชนิดอื่นไว้ใน array นั่นเอง มาถึงตอนนี้ถ้าเราลองมองย้อนกลับไปดูการประกาศ method main() ของเรา เราจะเห็นการประกาศแบบนี้

```
public static void main(String[] args) { …
```

ภายในเครื่องหมาย () เราจะเห็นการประกาศ String [] args ซึ่งเป็นการประกาศให้ตัวแปร args เป็นตัวแปรที่ใช้เก็บ array of strings ซึ่งการประกาศนี้ก็เหมือนกับการประกาศ array โดยทั่วไปเพียงแต่เราเปลี่ยนให้การจัดเก็บข้อมูลมาเป็น string เท่านั้นเอง ลองมาดูโปรแกรมตัวอย่างการสร้าง array of strings กันดู

```
//ArrayOfStrings.java

import java.io.*;
```

```
class ArrayOfStrings {
    public static void main(String[] args) {
        //declare array of 5 strings
        String[] names = new String[5];

        //initialize array
        names[0] = "Kafe";
        names[1] = "River Side";
        names[2] = "Good View";
        names[3] = "Cottage";
        names[4] = "Club G";

        String word = names[2].substring(0, 4);
        System.out.println("First word from names[2] is " + word);

        for(int i = 0; i < 5; i++) {
            System.out.println(names[i]);
        }
    }
}
```

เราได้ประกาศให้ names เป็น array of Strings ที่สามารถเก็บข้อมูลได้สูงสุด 5 ตัว ด้วยคำสั่ง

```
String[] names = new String[5];
```

หลังจากนั้นเราก็ใส่ข้อมูลให้กับตำแหน่งต่าง ๆ ของ array names และเมื่อใส่ครบแล้ว เราดึงเอาคำแรกที่อยู่ใน names[2] มาแสดง เมื่อเสร็จแล้วเราก็แสดงข้อมูลทั้งหมดไปยังหน้าจอ ด้วยการใช้ for – loop ผลลัพธ์ของการ run คือ

```
First word from names[2] is Good
Kafe
River Side
Good View
Cottage
Club G
```

โปรแกรมตัวอย่างที่เห็นเป็นโปรแกรมอย่างง่าย ที่เขียนขึ้นมาเพื่อแสดงการใช้ array of Strings กระบวนการต่าง ๆ ที่สามารถทำได้กับ array โดยทั่วไปก็สามารถทำได้กับ array of Strings ดังนั้นผู้อ่านควรย้อนกลับไปดูเรื่องของ array ที่เราได้พูดถึง และลองประยุกต์ใช้กระบวนการต่าง ๆ กับ array of Strings ดูเพื่อให้เกิดความเข้าใจมากยิ่งขึ้น

สรุป

ในบทนี้เราได้พูดถึงการเก็บข้อมูลด้วย array การใช้ String และ StringBuffer รวมไปถึงกระบวนการต่าง ๆ ที่เราสามารถทำได้กับตัวแปร หรือ object ต่าง ๆ ที่เกิดจาก class String และ class StringBuffer ผู้อ่านควรทำความเข้าใจกับ method ต่าง ๆ ที่เราได้พูดถึงเพื่อการนำไปใช้อย่างมีประสิทธิภาพ จุดหลัก ๆ ที่เราได้พูดถึง คือ

- ✓ การสร้าง array ในการเก็บข้อมูลชนิดเดียวกันหลายตัวไว้ในตัวแปรตัวเดียว
- ✓ การเข้าหาข้อมูลแต่ละตัวใน array ด้วยการใช้ index
- ✓ ขนาดของ array สามารถดึงออกมาจากตัวแปรคงที่ length
- ✓ Array สามารถที่จะเก็บข้อมูลที่เป็น array ได้
- ✓ เราสร้าง object ที่เป็น string จาก class String พร้อมทั้งกำหนดค่าที่ไม่สามารถเปลี่ยนแปลงได้
- ✓ ขนาดของ string ต้องดึงออกมาจาก method length()
- ✓ String มี method หลาย ๆ ตัวให้เราใช้ในการจัดการกับ string
- ✓ เราสร้าง string ที่สามารถเปลี่ยนแปลงได้จาก class StringBuffer

- ✓ StringBuffer มี method หลายตัวที่เราเรียกใช้ในการจัดการกับ string ต่าง ๆ
- ✓ ขนาดของ string จาก class StringBuffer หาได้จาก method length() และความจุของ object ที่มาจาก StringBufer หาได้ด้วยการเรียกใช้ method capacity()

แบบฝึกหัด

1. จงเขียนโปรแกรมที่รับข้อมูลทีเป็น int จาก keyboard จำนวน 10 ตัว เก็บไว้ใน array พร้อมทั้งหาค่าสูงสุด ค่าต่ำสุด จำนวนของข้อมูลที่เป็นเลขคี่ทั้งหมด และจำนวนของข้อมูลที่เป็นเลขคู่ทั้งหมด

2. จงเขียนโปรแกรมที่ใช้ Math.random() ในการสร้างข้อมูลจำนวน 100 ตัว เก็บไว้ใน array ให้แสดงข้อมูลทั้งหมดไปยังหน้าจอโดยให้มีจำนวนข้อมูลเป็น 5 แถว ๆ ละ 20 ตัว

3. จงเขียนโปรแกรมที่สร้าง array 2 มิติขนาด 3 x 3 จำนวน 2 ตัวที่มีข้อมูลเป็น int ให้ใส่ข้อมูลใน array ด้วยการใช้ Math.random() ที่กำหนดให้ค่าของข้อมูลที่สร้างขึ้นอยู่ระหว่าง 1 – 9 หลังจากนั้นให้นำเอา array 2 ตัวนี้มาบวกกัน (Matrix addition) เก็บผลลัพธ์ทั้งหมดที่ได้ไว้ใน array ตัวที่สาม พร้อมทั้งแสดง array ทั้งสามตัวไปยังหน้าจอ

4. จงเขียนโปรแกรมที่รับ object จาก class String จำนวนสามตัวจาก keyboard ให้ตรวจสอบว่า string ตัวไหนใหญ่ที่สุด และ string ตัวไหนเล็กที่สุด แสดงผลลัพธ์ไปยังหน้าจอ

5. จงเขียนโปรแกรมที่ใช้ array เก็บ string จำนวน 10 ตัวที่อยู่ในรูปแบบของ day/month/year เช่น 02/10/00 ให้โปรแกรมตรวจสอบข้อมูลที่อยู่ใน string ทุกตัว เสร็จแล้วให้ส่งผลลัพธ์ทั้งหมดในรูปแบบของ 2 October 2000 ไปยังหน้าจอ

6. จงเขียนโปรแกรมที่ใช้ array ในการเก็บ char จำนวน 10 ตัว ให้ทำการ reverse ข้อมูลที่อยู่ใน array นี้ ผลลัพธ์ที่ได้ให้เก็บไว้ใน array ตัวเดิม

7. จงเขียนโปรแกรมที่ใช้ StringBuffer เก็บ string จำนวน 10 ตัว ให้นำ string ทั้งหมดไปเก็บไว้ใน string ตัวใหม่โดยให้ string ที่มีจำนวนของ char น้อยอยู่ทางด้านหน้า และ string ที่มีจำนวนของ char มากอยู่ทางด้านหลัง (เรียงจากน้อยไปหามาก)

8. จงเขียนโปรแกรมที่ใช้ array ในการเก็บชื่อของลูกค้าจำนวน 20 ชื่อ ให้ทำการค้นหาชื่อของลูกค้าที่มีตัวอักษรขึ้นต้น ตามที่ user เป็นผู้กำหนดจาก keyboard ให้แสดงชื่อทุกตัวที่มีอักษรขึ้นต้นดังกล่าวออกทางหน้าจอ

9. กำหนดให้ array1 และ array2 เป็น array ที่เก็บ int จำนวนเท่ากับ 10 ตัวโดยที่ ข้อมูลของแต่ละ array ได้ถูกจัดเรียงให้อยู่ในรูปของ มากไปหาน้อย จงเขียนโปรแกรมที่นำเอาข้อมูลของ array ทั้งสองตัวมารวมกันโดยที่ยังรักษาคุณสมบัติของการเรียงจากมากไปหาน้อย เก็บไว้ใน array ตัวใหม่ แสดงผลลัพธ์ของทั้งสาม array ออกทางหน้าจอ

10. จงเขียนโปรแกรมที่รับ String 1 ตัวที่ประกอบไปด้วย ชื่อต้น และนามสกุลที่ถูกแบ่งด้วยช่องว่าง (space) เช่น Micha Sapporo หลังจากนั้นให้แบ่ง String ตัวนี้ออกเป็น 2 ตัวให้เป็น ชื่อต้น 1 ตัว และนามสกุล 1 ตัว

11. จงปรับปรุงโปรแกรมในข้อ 10 ให้ทำหน้าที่ลบช่องว่างที่อาจนำหน้า และตามหลัง ชื่อที่นำเข้าจาก keyboard

12. จงเขียนโปรแกรมที่รับ String 2 ตัวจาก keyboard ให้ทำการค้นหาว่า String ตัวที่สองมีอยู่ใน String ตัวแรกหรือไม่ ถ้ามี มีอยู่กี่ตัว อยู่ที่ตำแหน่งใดบ้าง

13. จงเขียนโปรแกรมที่รับ String 2 ตัวจาก keyboard ให้นำเอา String ตัวที่สองไปใส่ไว้ในตำแหน่งที่ถูกกำหนดจาก user ถ้าตำแหน่งที่ user กำหนดเป็นไปไม่ได้ให้นำ String นั้นไปใส่ไว้ด้านหลังของ String ตัวแรก

14. จงเขียนโปรแกรมที่ทำหน้าที่ดังนี้

รับ String ในรูปแบบของ 112 Newell Street, Walla Walla, WA 99210
ทำการตัด String ให้อยู่ในรูปแบบของ

No. 112
Street: Newell Street
City: Walla Walla
State: WA
Zip: 99210

15. จงเขียนโปรแกรมที่ทำการสลับคำที่มีอยู่ใน String เช่น ถ้า String เป็น I love Chiang Mai ผลลัพธ์ที่ได้จากการสลับจะเป็น Mai Chiang love I

5
Objects และ Classes

เราได้เรียนรู้ถึงโครงสร้างหลักของการเขียนโปรแกรมโดยทั่วไปจากบทก่อน ๆ เราได้เรียกใช้ class ต่าง ๆ (บาง class) ที่ Java มีให้ในการเขียนโปรแกรม รวมไปถึงการสร้าง object จาก class String และ class StringBuffer ในบทนี้เราจะมาทำความเข้าใจในเรื่องของการสร้าง class การสร้าง object จาก class ที่เราสร้างขึ้น การนำเอา class มาช่วยแก้โจทย์และปัญหาทางด้านคอมพิวเตอร์

หลังจากจบบทนี้แล้ว ผู้อ่านจะได้รับทราบในเรื่องของ

- ความหมายของ class และการสร้าง class
- การสร้าง constructor
- การสร้าง method
- การ overload method
- การสร้าง object จาก class
- การใช้ attribute ต่าง ๆ ของ class
- การสร้าง nested class
- การสร้าง และการเรียกใช้ package

Class และ การสร้าง class

เราได้เห็นจากบทก่อน ๆ ว่าเราต้องกำหนดโปรแกรมที่เราเขียนขึ้นด้วยคำว่า class เสมอ แต่เราไม่ได้พูดถึงเลยว่า class คืออะไร โดยทั่วไปการเขียนโปรแกรมด้วยภาษาที่เรียกกันว่า Object-Oriented Programming (OOP) นั้นคำว่า class มักจะหมายถึงคุณลักษณะ หรือคุณสมบัติ (prescription) ของวัตถุใดวัตถุหนึ่ง (ที่เกิดจาก class นั้น ๆ) ซึ่งถ้าพูดแบบภาษาชาวบ้านทั่ว ๆ ไปเราอาจเรียก class ดังกล่าวว่าเป็นแม่แบบของ class (อื่น ๆ ที่ตามมาก็ได้) หรือว่าเป็นตัวกำหนดคุณสมบัติของ object ที่ได้ถูกสร้างขึ้น

ถ้าเรามองไปรอบ ๆ ตัวเรา เราจะเห็นวัตถุหลากหลายชนิด เช่น เก้าอี้ keyboard ถ้วยกาแฟ และอะไรอื่น ๆ อีกมากมาย ซึ่งวัตถุหลายชิ้นมีรูปร่างหน้าตาเหมือนกัน มีการใช้สอยที่คล้าย หรือ เหมือนกัน เช่น เก้าอี้มีสี่ขา หรือสามขา มีพนักพิง หรือไม่มีพนักพิง แต่การใช้งานเหมือนกันคือ เอาไว้นั่ง หรืออีกตัวอย่างหนึ่ง เช่น รถยนต์ มีสี่ล้อ ขับเคลื่อนไปได้ทั้งข้างหน้า และข้างหลัง ต้องใช้น้ำมันเป็นเชื้อเพลิง มีสีภายนอกที่ต่างกัน แต่มีคุณสมบัติที่เหมือนกันคือเอาไว้เดินทางไปยังที่ต่าง ๆ และโดยทั่วไปแล้วผู้ขับขี่ส่วนใหญ่จะไม่รู้ถึงการทำงานของเครื่องยนต์ที่อยู่ภายใน รู้แต่เพียงข้อมูลที่จำเป็นบางส่วน เช่นเปิดประตูอย่างไร start เครื่องยังไง เข้าเกียร์อย่างไร อย่างนี้เป็นต้น การเขียนโปรแกรมในรูปแบบของ OOP ก็เหมือนกัน ถ้าเราพัฒนาโปรแกรมใดสักโปรแกรมหนึ่งให้ผู้อื่นใช้ ผู้ใช้กลุ่มนั้นไม่จำเป็นที่จะต้องรู้ว่าเราเขียนอย่างไร ควรรู้เพียงแต่ว่าจะใช้ส่วนต่าง ๆ ที่เราเขียนขึ้นอย่างไร และนำกลับไปใช้ใหม่ได้อย่างไร (เหมือนเช่นที่เราเรียกใช้ method ต่าง ๆ จาก class String เราไม่รู้ว่าเขาเขียน code อย่างไร รู้แต่ว่าใช้งานอย่างไร)

ถ้ามองกลับไปดู class String ที่เราใช้ในบทที่สี่นั้น จะเห็นว่าเราสามารถที่จะใช้ method ต่าง ๆ ที่มีโดยไม่มีการจำกัดจำนวนครั้งที่ใช้ โปรแกรมตัวไหนเรียกใช้ก็ได้ การทำงานก็ยังคงเหมือนเดิม ผลลัพธ์ของการเรียกใช้ก็เหมือนเดิม object ที่มาจาก class String ไม่ว่าจะเป็นตัวไหน เราก็สามารถที่จะใช้ method ต่าง ๆ กับ object เหล่านี้ได้ ในการออกแบบ class นั้นโดยทั่วไปเราต้องออกแบบให้เหมาะสมกับงานของเรา เช่นถ้าโปรแกรมของเราต้องวุ่นวายกับคนเราก็อาจออกแบบให้ class ของเรามีคุณสมบัติและคุณลักษณะ ที่เกี่ยวข้องกับคน เช่น มีชื่อ นามสกุล ที่อยู่ สถานะภาพ อาชีพ และอะไรอื่น ๆ ท่านองนี้ โดยทั่วไปการออกแบบ class นั้นมีส่วนประกอบที่จำเป็น และ สำคัญอยู่สองส่วน คือ

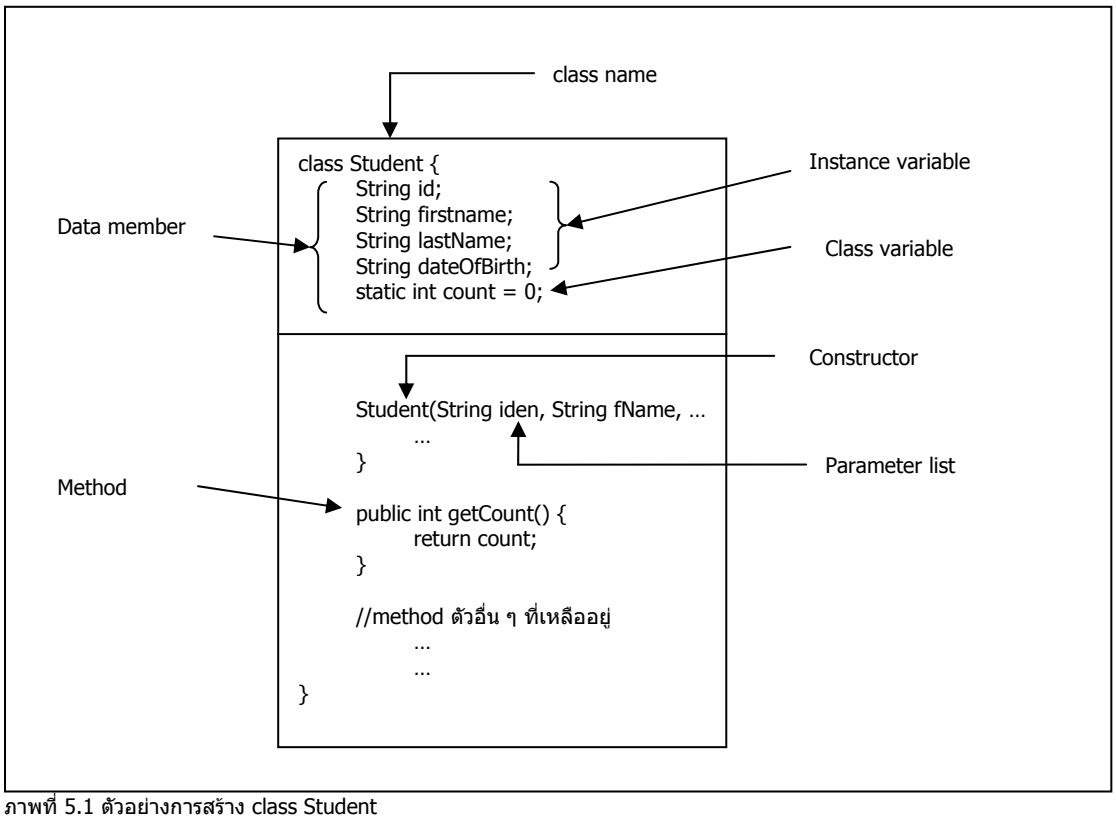
- Field หมายถึงตัวแปรสำหรับการเก็บข้อมูลของ object ที่บ่งบอกถึงความเหมือน หรือ ความแตกต่างกันของ object ที่เกิดมาจาก class เดียวกัน หนังสือหลายเล่มเรียก field ที่ว่านี้ว่า Data member หรือ สมาชิกของ class ที่เป็นที่เก็บข้อมูล หรือที่เราเรียกกันจนคุ้นปากว่า variable (identifier)

- Method หมายถึงกระบวนการ (operation) ต่าง ๆ ที่เราสามารถทำได้กับ object นั้น ๆ ของ class ซึ่งกระบวนการต่าง ๆ ที่ทำได้ส่วนใหญ่จะกระทำกับ data member

Field หรือ Data member สามารถที่จะกำหนดให้เป็นข้อมูลได้ทุกชนิด รวมไปถึงการเป็นตัวอ้างอิง (reference) ถึง object ที่มาจาก class อื่น (เช่น ตัวแปรที่มีชนิดเป็น String หรือ array ที่เราได้พูดถึงก่อนหน้านี้) หรือแม้แต่กระทั่งการเป็นตัวอ้างอิงถึง class ของมันเอง

Method จะประกอบไปด้วยชื่อของ method และชุดคำสั่งต่าง ๆ ที่ตัวมันเองมีหน้าที่ในการทำตามคุณลักษณะที่ได้ถูกออกแบบไว้ ซึ่งโดยส่วนใหญ่จะกระทำกับ data member (แต่ก็ไม่จำเป็นเสมอไป เช่นในกรณีของ method main())

เพื่อให้เห็นภาพชัดเจนยิ่งขึ้น เรามาลองออกแบบ class ขึ้นมาสัก class หนึ่งที่เกี่ยวข้องกับ Student



ภาพที่ 5.1 ตัวอย่างการสร้าง class Student

Class Student ที่เราสร้างขึ้นประกอบไปด้วยส่วนประกอบสองส่วนดังที่ได้กล่าวไปแล้ว คือ ส่วนที่เป็น data member และส่วนที่เป็น method ซึ่งภายในส่วนสองส่วนนี้ ยังมีส่วนประกอบอื่นอีกที่มีความสำคัญต่อการออกแบบ และการใช้ class ก่อนอื่นเรามาดูส่วนประกอบของ data member กัน

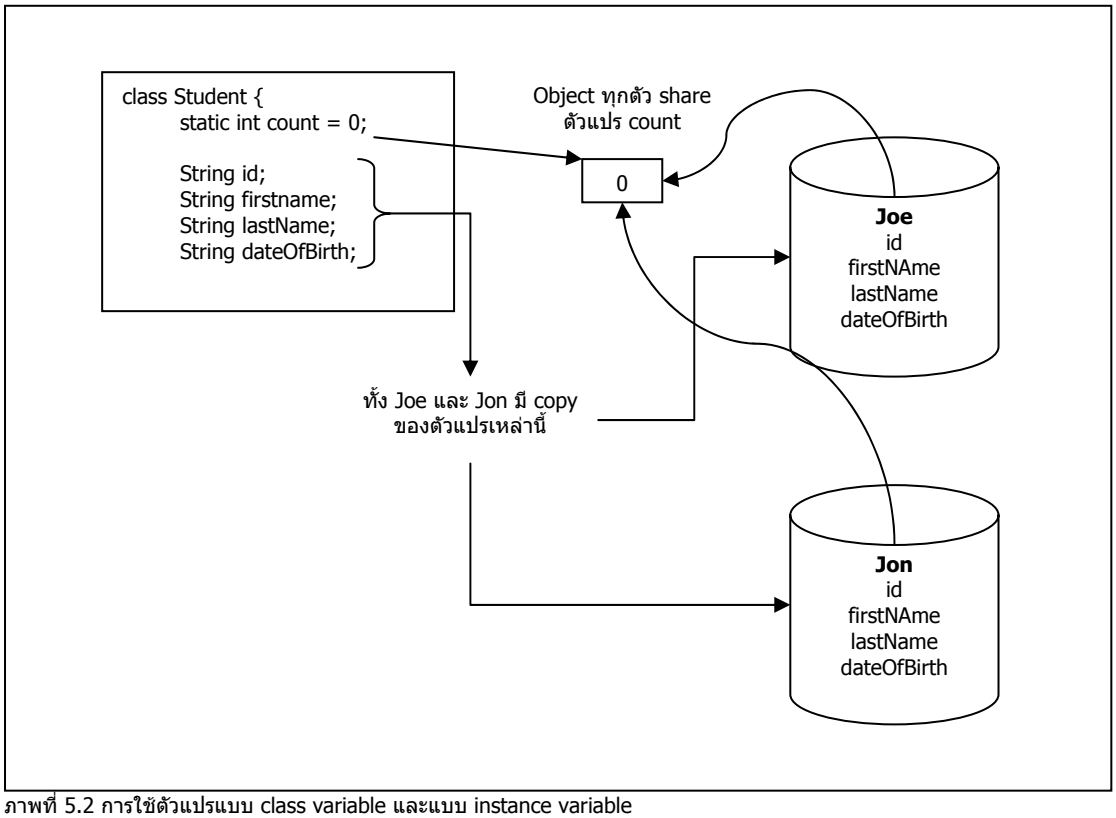
ตัวแปรที่เป็นสมาชิกของ class (Class variable และ Instance variable)

Object ที่เกิดมาจาก class นั้นเราเรียกกันว่า instance ของ class และ object นี้จะมีส่วนประกอบที่ class มีให้ คือ ตัวแปรต่าง ๆ ที่อยู่ใน class ตัวแปรเหล่านี้จะมีข้อแตกต่างกัน คือ อาจเป็นตัวแปรที่เป็น class variable และอาจเป็นตัวแปรที่เป็น instance variable

หลังจากที่เราสร้าง object แล้ว object จะได้รับ copy ของตัวแปรที่ class มีให้ ซึ่งตัวแปรเหล่านี้จะเป็นตัวบ่งบอกว่า object ตัวไหนแตกต่างกันอย่างไร ค่าของตัวแปรของ object แต่ละตัวจะมีค่าเหมือนหรือ

แตกต่างกันนั้นขึ้นอยู่กับการกำหนดของโปรแกรมเอง เช่น ถ้าเราสร้าง object สองตัวจาก class Student ขึ้นมาใช้งาน object สองตัวนี้จะมี copy ของตัวแปรทั้งสี่ตัวของมันเอง เราเรียกตัวแปรเหล่านี้ว่า instance variable

ส่วนตัวแปรอีกชนิดหนึ่งคือ class variable นั้นกำหนดไว้ว่า object ทุกตัวที่เกิดจาก class จะใช้ตัวแปรชนิดนี้ร่วมกัน และตัวแปรนี้สามารถที่จะกำหนดและสร้างขึ้นมาใช้ได้ ถึงแม้ว่าจะไม่มี object เกิดขึ้นจาก class นี้ก็ตาม เพราะว่าตัวแปรนี้เป็นของ class ดังนั้นทั้ง class และ object ก็สามารถที่จะใช้มันได้ ถ้าค่าของตัวแปรนี้เปลี่ยนไป object หรือ class ที่อ้างการใช้ถึงตัวแปรตัวนี้ก็จะได้ค่าใหม่ที่เปลี่ยนไป ซึ่งตรงกันข้ามกับตัวแปรที่เป็น instance variable ถ้าหาก object ตัวไหนเปลี่ยนค่าของตัวแปรใด ๆ (copy) ค่าที่เปลี่ยนไปจะไม่มีผลต่อตัวแปรใด ๆ ใน object ตัวอื่น การประกาศให้ตัวแปรใด ๆ ก็ตามเป็น class variable นั้นเราต้องใช้คำสั่ง static นำหน้าตัวแปรนั้นเสมอ ดังที่ได้แสดงในภาพที่ 5.2



ภาพที่ 5.2 การใช้ตัวแปรแบบ class variable และแบบ instance variable

ตัวแปรทั้งสองชนิดมีการใช้ที่แตกต่างกัน และต่างก็มีความสำคัญทั้งคู่ โดยทั่วไปการใช้ class variable นั้นนิยมใช้ในกรณีที่เราต้องการที่จะเก็บค่าที่เป็นค่าที่ object ทั้งหมดต้องใช้ร่วมกัน เช่นค่าคงที่ต่าง ๆ ที่มีความจำเป็นใน class นั้น ๆ ตัวอย่างของเราใช้ count เป็นตัวแปรที่ใช้ร่วมกันระหว่าง object ต่าง ๆ ส่วนตัวแปรที่เป็น instance variable นั้นมีความจำเป็นอย่างมากในการออกแบบ class ต่าง ๆ ทั้งนี้เพราะตัวแปรชนิดนี้จะเป็นผู้กำหนดความแตกต่างระหว่าง object ต่าง ๆ ที่ได้ถูกสร้างขึ้นจาก class นั้น ๆ เช่น object จาก class Student ที่ถูกสร้างขึ้นอาจมี id ที่ไม่เหมือนกัน ชื่อต้น และ ชื่อสกุลที่แตกต่างกัน วันเดือนปีเกิดที่ไม่เหมือนกัน อย่างนี้เป็นต้น

ข้อแตกต่างระหว่าง class กับ object

class	**object**
▪ เป็นแม่แบบที่มีสมาชิกทั้ง data และ method	▪ ต้องเกิดมาจาก class ใด class หนึ่ง
▪ ไม่มีตัวตนขณะที่โปรแกรมกำลัง execute	▪ มีตัวตนขณะโปรแกรมกำลัง execute
▪ ไม่มีการเปลี่ยนแปลงขณะโปรแกรม execute	▪ สามารถเปลี่ยนแปลงค่าของ data ขณะ execute

Method

จากแผนภาพที่ 5.1 เราจะเห็นว่ามี method อยู่ใน class Student ของเรา (บางส่วน) method เป็นกระบวนการ (ที่รวบรวมชุดคำสั่ง) ที่เราสามารถเรียกใช้ให้ทำงานต่าง ๆ ให้เรา ซึ่งอาจเป็นการทำงานที่เกี่ยวข้องกับ ตัวแปรที่อยู่ใน class หรือกระบวนการอื่น ๆ ที่จำเป็นของ class นั้น ๆ และเช่นเดียวกันกับตัวแปรที่ได้กล่าวไว้ก่อนหน้านี้ method ก็ได้ถูกแบ่งออกเป็นสองชนิด คือ class method และ instance method

Class method ก็คือ method ที่สามารถที่จะได้รับการประมวลผลถึงแม้ว่าจะไม่มี object อยู่เลย ถ้าเรามองย้อนกลับไปดู method main() ของเรา เราจะเห็นว่าไม่มี object ใด ๆ เกิดจาก class ที่มี method main() อยู่เลย และการประกาศใช้ class method ก็เช่นเดียวกันกับ class variable เราต้องใช้คำสั่ง static นำหน้าเสมอ

ส่วน instance method นั้นเป็น method ที่ต้องมีการใช้ร่วมกันกับ instance variable ทั้งนี้เพราะว่า method ประเภทนี้ถูกสร้างขึ้นมาจาก class เดียวกันกับที่ object ถูกสร้างขึ้นมา การกระทำใด ๆ ก็ตามต้องทำผ่าน instance method เท่านั้น

การเข้าหาตัวแปร และ method ของ class

ในการที่จะเข้าหาตัวแปรหรือ method ของ class นั้นไม่ว่าจะเพื่ออะไรก็ตามแต่ เราจะต้องเข้าหาผ่านทาง object ของ class ตามด้วย . (dot) และตามด้วยชื่อของตัวแปร หรือ ชื่อของ method เช่น

```
studentA.firstName;
```

หรือ

```
studentB.getAge();
```

ในตอนนี้เราจะดูเพียงแค่วิธีการเข้าหา แต่ต่อไปเราจะดูเรื่องการกำหนดนโยบายของการเข้าหาตัวแปรหรือ method เหล่านี้ ถ้ามี method หรือ ตัวแปรอยู่เราก็เข้าหาแบบที่ได้กล่าวมาแล้ว Java จะฟ้องด้วย error ถ้าหากอ้างถึงตัวแปรหรือ method ที่ไม่ได้ถูกสร้างขึ้น เพื่อให้เห็นภาพของการสร้าง class รวมไปถึงส่วนประกอบต่าง ๆ ของ class เรามาดู class Student ที่ได้ถูกสร้างขึ้นเพื่อเป็นตัวอย่าง

```
//Student.java - Simple class for student

import java.lang.Integer;
import java.util.Calendar;

class Student {
    String id;            //student id
    String firstName;     //student's first name
    String lastName;      //student's last name
    String dateOfBirth;   //student's date of birth in
                          //the form: dd/mm/yyyy
    static int count = 0; //number of object created

    //class constructor to initialize fields to given values
    Student(String iden, String fName, String lName, String dOfB) {
        id = iden;
        firstName = fName;
        lastName = lName;
        dateOfBirth = dOfB;
        count++;
    }

    //method to return count
    public int getCount() {
```

```
        return count;
    }

    //method to return student's id
    public String getId() {
        return id;
    }

    //method to returnstudent's first name
    public String getFirstName() {
        return firstName;
    }

    //method to return student's last name
    public String getLastName() {
        return lastName;
    }

    //method to return student's date of birth
    public String getDateOfBirth() {
        return dateOfBirth;
    }

    //method to calculate student's age from
    //year of birth
    public int getAge() {
        //retrieve a year
        String year = dateOfBirth.substring(6);
        //convert it to an int
        int birthYear = Integer.parseInt(year);

        //get current year from system's calendar
        Calendar now = Calendar.getInstance();
        int thisYear = now.get(Calendar.YEAR);
        //return the difference
        return thisYear - birthYear;
    }

    //method to display students' info to screen
    public void display() {
        System.out.print(getId() + " ");
        System.out.print(getFirstName() + " ");
        System.out.print(getLastName() + " ");
        System.out.println("Age: " + getAge());
    }
}
```

เราได้กำหนดให้ class Student มี field อยู่ทั้งหมด 5 field คือ id, firstName, lastname, dateOfBirth, และ count โดยที่สี่ตัวแรกเป็น instance variable ที่เอาไว้เก็บข้อมูลโดยเฉพาะของ object แต่ละตัวที่ได้ถูกสร้างขึ้น และตัวสุดท้ายเป็น class variable ที่เอาไว้เก็บจำนวนของ object ที่ได้ถูกสร้างขึ้น

เรายังได้สร้าง method อีก 7 ตัวซึ่งมีหน้าที่ในการทำงานที่ต่างกัน โดยเฉพาะ method ที่มีชื่อเหมือนกันกับ class นั่นก็คือ method Student

```
Student(String iden, String fName, String lName, String dOfB) {
    id = iden;
    firstName = fName;
    lastName = lName;
```

```
    dateOfBirth = dOfB;
    count++;
}
```

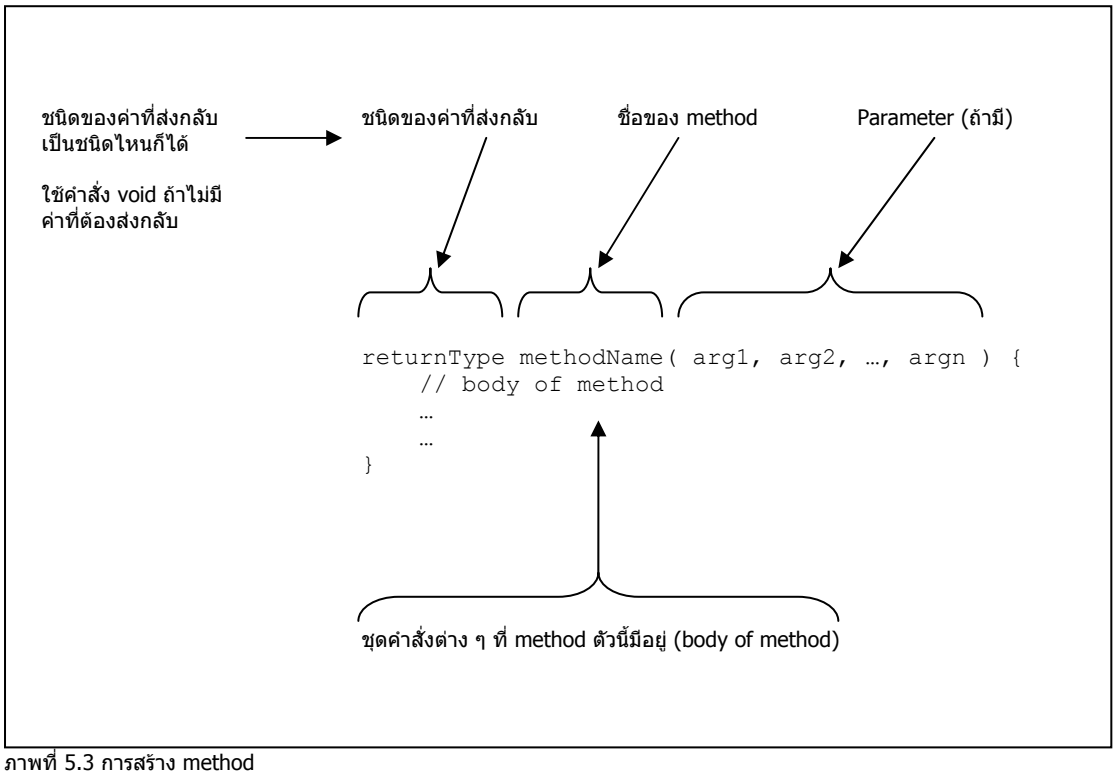
Student() เป็น method พิเศษที่มีชื่อเรียกกันโดยทั่วไปว่าเป็น constructor ของ class ซึ่ง constructor นี้ส่วนใหญ่จะทำหน้าที่ในการกำหนดค่าเบื้องต้นให้กับ object ที่ได้ถูกสร้างขึ้นจาก class เช่น กำหนดค่าให้กับ field ทุก field หรือบาง field (ทั้งนี้ต้องแล้วแต่การออกแบบการทำงานของ constructor นั้น ๆ) ตัวอย่างของเรากำหนดให้ constructor ทำการกำหนดค่าให้กับ field ทุก field (ยกเว้น class variable ที่ชื่อ count) ให้มีค่าเป็นไปตามค่าที่ได้รับเข้ามาจากตัวแปรที่อยู่ใน parameter list แต่ constructor จะมีคุณสมบัติพิเศษคือ การกำหนดค่านั้นจะได้รับการกระทำโดยอัตโนมัติ เราไม่ต้องเรียกใช้ constructor การเรียกใช้ constructor นี้ Java จะเป็นผู้เรียกให้ในตอนที่เราสร้าง object ขึ้นมา ก่อนที่เราจะพูดถึง constructor และการถูกเรียกใช้งานโดยอัตโนมัตินั้นเรามาดูกันถึง method และ วิธีการสร้าง method รวมไปถึงการส่งค่าให้กับ method และการส่งค่ากลับของ method

การสร้าง method

ในการสร้าง method นั้นเราจะต้องกำหนดชื่อให้กับ method กำหนด parameter (ถ้ามี) กำหนดการส่งค่ากลับของ method (ถ้ามี) โดยทั่วไป method ได้ถูกแบ่งออกเป็นสองแบบ คือ

- Method ที่ส่งค่ากลับให้แก่ผู้ที่เรียกใช้ method
- Method ที่ไม่มีการส่งค่าใด ๆ กลับไปให้ผู้เรียก

โครงสร้างของ method โดยทั่วไปจะมีรูปแบบดังนี้



ภาพที่ 5.3 การสร้าง method

เราลองหยิบเอา method getAge() จาก class Student มาดูกันเพื่อให้เกิดความเข้าใจในเรื่องส่วนประกอบต่าง ๆ ที่มีอยู่ใน method ที่ว่านี้

```
public int getAge() {
    String year = dateOfBirth.substring(6);
    int birthYear = Integer.parseInt(year);
```

```
    Calendar now = Calendar.getInstance();
    int thisYear = now.get(Calendar.YEAR);
    return thisYear - birthYear;
}
```

ถ้าเราแยกส่วนประกอบต่าง ๆ ออก เราก็จะได้ข้อมูลดังนี้

ชนิดของค่าที่ส่งกลับ	int
ชื่อของ method คือ	getAge
Parameter	ไม่มี

ชุดคำสั่งคือ	String year = dateOfBirth.substring(6);
	int birthYear = Integer.parseInt(year);
	
	Calendar now = Calendar.getInstance();
	int thisYear = now.get(Calendar.YEAR);
	return thisYear - birthYear;

ถ้าสังเกตให้ดีจะเห็นว่าด้านหน้าสุดของ method getAge() จะมีคำว่า public อยู่ คำว่า public นี้จะเป็นตัวกำหนดนโยบายการเข้าหา method ซึ่งเราจะได้พูดถึงในตอนต่อไป แต่ตอนนี้เราจะดูเฉพาะส่วนประกอบโดยทั่วไปที่ method ต้องมี

ส่วนประกอบที่สำคัญอีกส่วนหนึ่งก็คือ คำว่า return ในบรรทัดสุดท้าย เนื่องจากว่า method getAge() ของเราต้องส่งค่ากลับ ดังนั้นเราจึงจำเป็นที่จะต้องบอกให้ Java รู้ว่าเรามีค่าที่ต้องการส่งกลับ ดังที่ได้ประกาศไว้ ถ้าหากว่าเราไม่มีการใช้ return พร้อมทั้งค่าที่ต้องการส่งกลับ Java ก็จะฟ้องด้วย error ตอนที่เรา compile โปรแกรม

เรามาลองดู method ที่ไม่มีการส่งค่ากลับใน class Student ของเรา ซึ่งมีอยู่เพียงตัวเดียว คือ

```
public void display() {
    System.out.print(getId() + " ");
    System.out.print(getFirstName() + " ");
    System.out.print(getLastName() + " ");
    System.out.println("Age: " + getAge());
}
```

ชนิดของค่าที่ส่งกลับ	ไม่มี
ชื่อของ method คือ	display
Parameter	ไม่มี

ชุดคำสั่งคือ	`System.out.print(getId() + " ");`
	`System.out.print(getFirstName() + " ");`
	`System.out.print(getLastName() + " ");`
	`System.out.println("Age: " + getAge());`

Method display() ทำหน้าที่เป็นเพียงตัวแสดงข้อมูลของ object ที่เกิดจาก class Student ออกทางหน้าจอ โดยการเรียกใช้ method อื่น ๆ ที่มีอยู่ใน class Student การที่จะให้ method ไม่ต้องส่งค่ากลับนั้นเราต้องใช้ คำว่า void นำหน้าชื่อ method เสมอ และ method ที่มีคำว่า void นำหน้าต้องไม่มีคำว่า return ใด ๆ ที่มีค่าในการส่งกลับ ยกเว้นคำว่า return ที่ตามด้วย ; (semicolon) เท่านั้น เช่นถ้าเราใช้ void เราก็สามารถที่จะใช้ return ตามด้วย ; หรือไม่ใช้เลย อย่างใด อย่างหนึ่ง เพราะฉะนั้น method display() ที่เห็นก็สามารถที่จะเขียนได้อีกแบบหนึ่งดังนี้

```
public void display() {
    System.out.print(getId() + " ");
    System.out.print(getFirstName() + " ");
    System.out.print(getLastName() + " ");
    System.out.println("Age: " + getAge());
```

```
    return ;
}
```

ข้อแตกต่างระหว่าง method ที่ส่งค่า และ method ที่ไม่ส่งค่า

method ที่ส่งค่า
- มีคำว่า return และ ค่าที่ต้องส่งกลับ
- ขึ้นต้น method ด้วยชนิดของข้อมูลที่ต้องส่งกลับ

method ที่ไม่ส่งค่า
- ไม่จำเป็นต้องมีคำว่า return แต่ถ้ามีจะต้องตามด้วย ; อย่างเดียวเท่านั้น
- ขึ้นต้น method ด้วยคำว่า void เสมอ

กระบวนการที่เกิดขึ้นเมื่อมีการใช้ parameter list

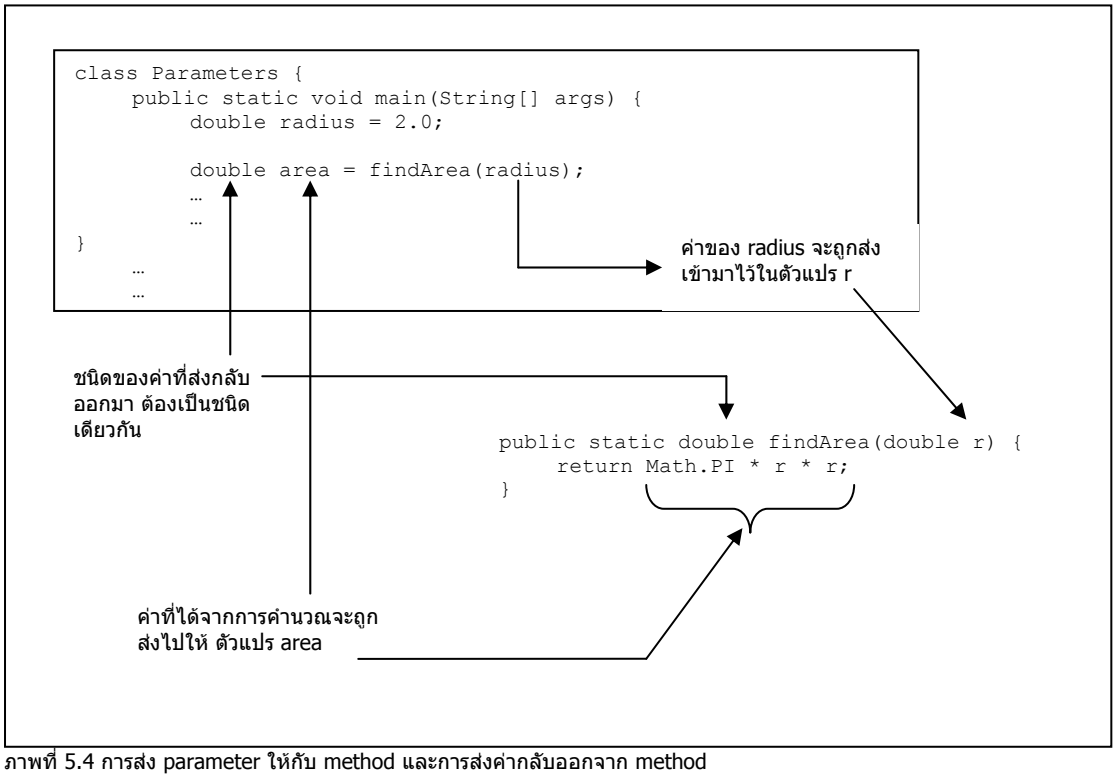
เราลองมาดูโปรแกรมตัวอย่างที่มีการสร้าง method ที่มีการใช้ parameter list ก่อนที่เราจะกลับมาดู class Student อีกครั้งหนึ่ง

```
//Parameters.java

class Parameters {
    public static void main(String[] args) {
        double radius = 2.0;

        double area = findArea(radius);

        System.out.println("Area of a circle is : " + area);
    }

    public static double findArea(double r) {
        return Math.PI * r * r;
    }
}
```

โปรแกรม Parameters.java สร้าง method ที่เราเรียกว่า class method ขึ้นมาใช้งานหนึ่งตัว (การประกาศให้ method เป็น class method ก็เหมือนกับการประกาศให้ตัวแปรเป็น class variable เราต้องใช้คำว่า static นำหน้า) ซึ่ง method นี้มีหน้าที่ในการคำนวณหาพื้นที่ของวงกลมที่มีรัศมีที่กำหนดไว้ในตัวโปรแกรม (ในที่นี้คือ 2) method findArea() มี Parameter หนึ่งตัวที่มีชนิดเป็น double และจะส่งค่าของพื้นที่ที่คำนวณได้กลับไปให้ผู้เรียก ซึ่งในที่นี้คือ class Parameters การเรียก method findArea(() นั้นเราเรียกด้วยประโยค

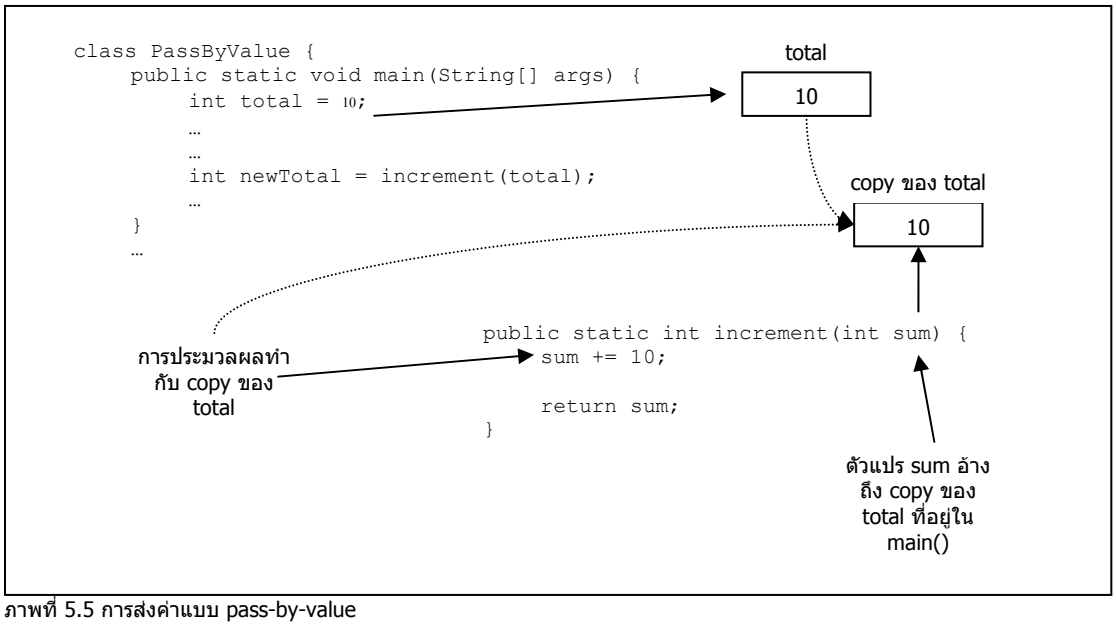
```
double area = findArea(radius);
```

ซึ่งเมื่อ Java เห็นการเรียกเช่นนี้ Java ก็จะไป execute method findArea() ด้วย parameter ที่มีชนิดเป็น double และส่งค่า 2.0 ไปให้ หลังจากที่คำนวณค่าพื้นที่ได้แล้วก็จะส่งกลับออกมา และนำไปเก็บไว้ในตัวแปรชื่อ area ภาพที่ 5.4 แสดงการส่งค่าผ่านทาง parameter list



ภาพที่ 5.4 การส่ง parameter ให้กับ method และการส่งค่ากลับออกจาก method

การส่งค่าผ่านทางตัวแปรที่อยู่ใน parameter list นั้นมีอยู่สองแบบคือ 1) การส่งที่เรียกว่า pass-by-value หรือการส่งเฉพาะค่าเข้าไปใน method นั้น ๆ และ 2) การส่งแบบที่เรียกว่า pass-by-reference ในการส่งค่าที่เป็น primitive type นั้น Java จะกำหนดให้การส่งเป็นแบบ pass-by-value เท่านั้น ซึ่งได้แสดงไว้ในภาพที่ 5.5



ภาพที่ 5.5 การส่งค่าแบบ pass-by-value

ตัวแปร total ที่อยู่ใน method main() นั้นจะไม่ได้รับผลกระทบจากการเปลี่ยนค่า ที่เกิดขึ้นภายใน method increment() ทั้งนี้ก็เพราะว่า กระบวนการที่เกิดขึ้นไม่ได้กระทำกับตัวแปร total แต่เป็นการกระทำกับ copy ของ total

การส่งค่าไปยัง method แบบ pass-by-value นี้ค่าที่ส่งเข้าไปจะถูกนำไปใช้ภายในตัว method เท่านั้น การเปลี่ยนแปลงใด ๆ ที่เกิดขึ้นจะไม่มีผลกระทบกับตัวแปรที่เป็นเจ้าของค่านั้น ภายนอก method เลย ซึ่งตรงกันข้ามกับ การส่งค่าในแบบที่เรียกว่า pass-by-reference ลองมาดูตัวอย่างโปรแกรมการส่งค่าแบบ pass-by-value กัน

```
//PassByValue.java

class PassByValue {
    public static void main(String[] args) {
        int total = 10;

        System.out.println("Before calling: total is " + total);

        System.out.println("Value returns from increment() is " +
                           increment(total));
        System.out.println("After calling: total is " + total);
    }

    public static int increment(int sum) {
        sum += 10;

        return sum;
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
Before calling: total is 10
Value returns from increment() is 20
After calling: total is 10
```

increment() ถูกเรียกภายใน main() จากประโยค System.out.println("Value returns from increment() is " + **increment(total)**); ซึ่งมีการส่งค่าของ total ไปให้ (ซึ่งมีค่าเป็น 10) ค่านี้ถูกนำไปใช้ใน increment() ผ่านทางตัวแปรที่ชื่อว่า sum ภายใน increment() ตัวแปร sum ถูกเปลี่ยนค่าด้วยการนำเอา 10 ไปบวกเพิ่ม ทำให้ sum มีค่าเป็น 20 ซึ่งค่านี้จะถูกส่งกลับไปยัง main() ทำให้การแสดงผลทางหน้าจอมีค่าเป็น 20 แต่หลังจากนั้น เรากำหนดให้มีการแสดงค่าของ total อีกครั้งหนึ่ง ซึ่งผลลัพธ์ที่ได้คือค่าของ total ยังคงเป็น 10 อยู่ จะเห็นว่าค่าของ total ที่ได้รับการเปลี่ยนแปลงภายใน increment() (ผ่านทางตัวแปร sum) ไม่มีผลต่อตัวแปร total อย่างใดเลย การเปลี่ยนแปลงเกิดและตายภายใน increment() เท่านั้น เรามาลองดูโปรแกรมตัวอย่างอีกโปรแกรมหนึ่ง

```
//PassByValue2.java

class PassByValue2 {
    public static void main(String[] args) {
        String myString = "Business Computers";

        System.out.println("Before string is: " + myString);

        changeString(myString); //calling changeString()

        System.out.println("After myString is: " + myString);
    }

    public static void changeString(String string) {
        System.out.print("\tInside changeString(), ");
        System.out.println("myString is: " + string);
        string = "at Far Eastern College";
```

```
    }
}
```

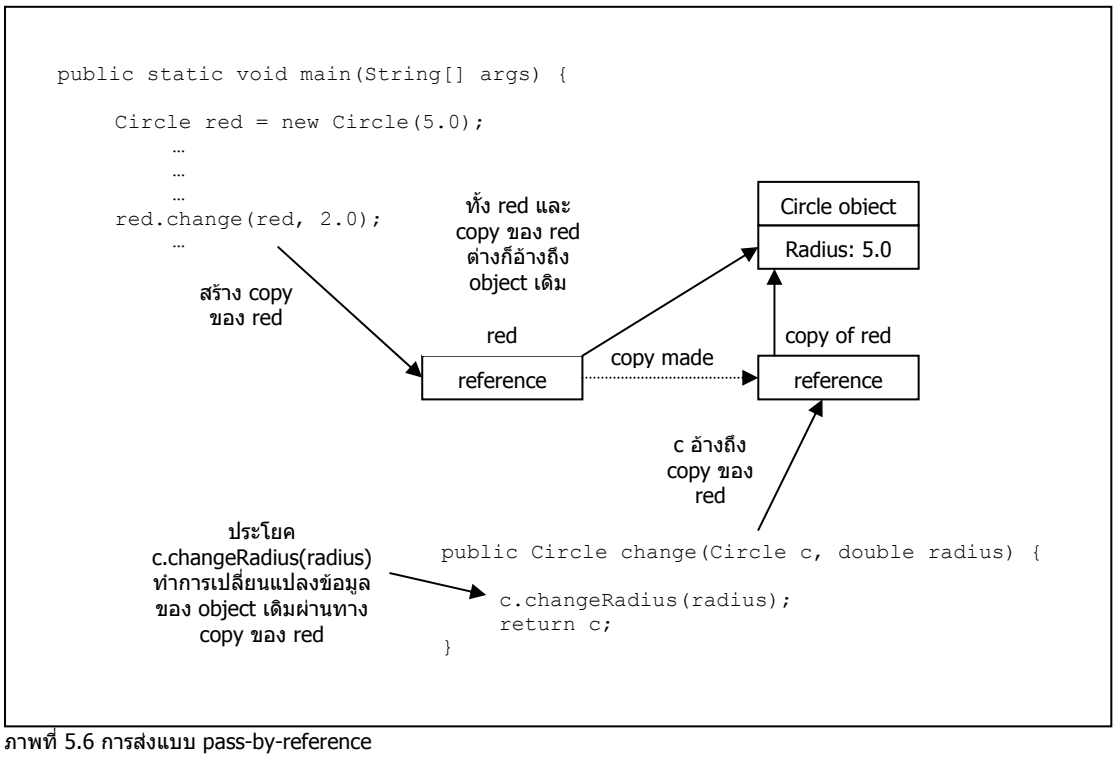
โปรแกรม PassByValue2.java สร้าง string หนึ่งตัวมีข้อความว่า "Business Computers" และส่ง string นี้ไปให้ method changeString() ซึ่งทำหน้าที่ในการเปลี่ยนค่าของ string ที่ส่งเข้ามาให้เป็น "at Far Eastern College" แต่หลังจากที่เรา run โปรแกรมดู เราก็ได้ผลลัพธ์ตามที่เห็นด้านล่างนี้

```
Before string is: Business Computers
    Inside changeString(), myString is: Business Computers
After myString is: Business Computers
```

จะเห็นว่าการเปลี่ยนแปลงค่าของ string ที่ส่งผ่านเข้ามานั้นจะเป็นเพียงการเปลี่ยนแปลงแบบชั่วคราวเท่านั้น การเปลี่ยนแปลงเกิดขึ้นภายใน changeString() และไม่มีผลกระทบใด ๆ กับค่าของตัวแปร myString แต่อย่างใดเลย

การส่งค่าแบบอ้างอิง pass-by-reference

การส่งค่าแบบ pass-by-reference นั้น สิ่งที่เราส่งเข้าไปใน method นั้นไม่ใช่ค่าของตัวแปร แต่เป็นตัวอ้างอิงถึงตัวแปร หรือ object นั้น ๆ โดยตรง เราอาจพูดได้ว่าสิ่งที่เราส่งไปนั้นเป็นที่อยู่ หรือ address ของตัวแปร หรือ object นั้น ๆ ก็ได้ เรามาลองดูตัวอย่างการส่งแบบ pass-by-reference กันในภาพที่ 5.6



ภาพที่ 5.6 การส่งแบบ pass-by-reference

ตัวอย่างการส่งแบบ pass-by-reference ที่เห็นในภาพที่ 5.6 นั้นเราสร้าง object จาก class Circle หนึ่งตัว โดยให้มีชื่อว่า red และมี radius เท่ากับ 5.0 เราเรียก method change() ผ่านทาง object red ด้วยการส่ง object red และค่าใหม่ของ radius ไปให้ เมื่อ method change() ถูกเรียกด้วย parameter ดังกล่าว Java จะทำการสร้าง copy ให้กับ red และจัดเก็บ copy นี้ไว้ที่ c เนื่องจากว่าทั้ง red และ c ต่างก็อ้างถึง object ตัวเดียวกัน ดังนั้นการเปลี่ยนแปลงใด ๆ ที่เกิดขึ้นย่อมมีผลกระทบต่อ object ตัวเดิมที่ทั้ง red และ copy ของ red อ้างถึง (หรือเป็นตัวแทนอยู่)

เราได้สร้าง method changeRadius() ขึ้นมาเพื่อใช้ในการเปลี่ยนค่าของ radius ซึ่ง code ทั้งหมดของโปรแกรมตัวนี้ มีดังนี้

```
//PassByReference.java

//a Circle class to demonstrate pass-by-reference scheme
class Circle {
    private double radius;   //a radius of a circle

    //constructor to initialize radius
    Circle(double r) {
        radius = r;
    }

    //method to calculate area of a circle
    public double area() {
        return Math.PI * radius * radius;
    }

    //a method to change radius of a given circle
    public Circle change(Circle c, double radius) {
        c.changeRadius(radius);  //calling method changeRadius()
        return c;                //with a given radius
    }

    //a method to change a given radius
    private void changeRadius(double r) {
        radius = r;
    }

    //a method to return radius
    public double getRadius() {
        return radius;
    }
}

//a class to test pass-by-reference scheme
class PassByReference {
    public static void main(String[] args) {
        //create a red circle with a radius of 5.0;
        Circle red = new Circle(5.0);

        //calculate area of a red circle
        double area = red.area();
        System.out.println("Radius of red circle is " +
                           red.getRadius());
        System.out.println("Area of red circle is " + area);

        //change radius of a red circle
        red.change(red, 2.0);
        area = red.area();
        System.out.println("Radius of red circle now is " +
                           red.getRadius());
        System.out.println("Area of red circle now is " + area);
    }
}
```

เรามี class อยู่สอง class โดยที่ class ตัวแรก หรือ class Circle เป็น class ที่เอาไว้ใช้สร้างวงกลม ต่าง ๆ รวมไปถึง method หรือกระบวนการต่าง ๆ ที่เราต้องการใช้ในการเปลี่ยนแปลง หรือกำหนดค่าให้กับ

วงกลมนั้น ๆ ส่วน class ตัวที่สองเป็น class ที่ใช้สำหรับการทดสอบ การสร้าง object จาก class Circle และตรวจสอบการส่งค่าแบบ pass-by-reference หลังจากที่ทดลอง run เราก็ได้ผลลัพธ์ดังนี้

```
Radius of red circle is 5.0
Area of red circle is 78.53981633974483
Radius of red circle now is 2.0
Area of red circle now is 12.566370614359172
```

จากผลลัพธ์เราจะเห็นว่าประโยคที่แสดงค่าของ radius นั้นแสดงค่าได้ถูกต้องทั้งก่อน และหลังการเปลี่ยนแปลงค่าของ radius

สิ่งที่เราต้องจำไว้ในเรื่องของการส่งค่าให้กับ method ก็คือ การส่งค่าที่เป็น primitive type เช่น int หรือ double นั้น method จะได้รับแต่เพียงค่าเท่านั้น การเปลี่ยนแปลงใด ๆ ภายใน method จะไม่มีผลกระทบต่อตัวแปรด้านนอก แต่ถ้าเราส่ง object ไปให้ method การเปลี่ยนแปลงที่เกิดขึ้นภายใน method (ต่อตัวแทนของ object นั้น ๆ) จะมีผลกระทบต่อ object ที่อยู่ภายนอกด้วย

สมมติว่าเราต้องการที่จะทำการสลับค่าของตัวแปรสองตัว ผ่านทาง method เราจะทำอย่างไร? เรามาลองดูตัวอย่างแรกกันก่อนว่าจะทำให้เราได้หรือไม่

```
//Swap.java

import java.io.*;

class Values {
    int val1;
    int val2;

    //default constructor
    Values() {
        val1 = val2 = 0;
    }

    //assignment constructor
    Values(int a, int b) {
        val1 = a;
        val2 = b;
    }

    //display val1 and val2
    public void show() {
        System.out.println("(" + val1 + ", " + val2 + ")");
    }
}

class Swap {
    public static void main(String[] args) {
        //create two objects from Values
        Values obj1 = new Values(10, 20);
        Values obj2 = new Values(30, 50);

        obj1.show();   //display values of obj1
        obj2.show();   //display values of obj2

        swap(obj1, obj2); //swap contents of obj1 and obj2

        obj1.show();
        obj2.show();
```

```
    }

    //swapping contents of given objects
    private static void swap(Values pointA, Values pointB) {
        Values temp = new Values();

        temp = pointA;    //copy pointA to temp
        pointA = pointB;  //copy pointB to pointA
        pointB = temp;    //copy temp to pointB
    }
}
```

เรากำหนดให้การสลับค่าของ object ทั้งสองเกิดขึ้นภายใน swap() ซึ่งเราคิดว่าน่าจะทำได้ 100% แต่หลังจากที่ run โปรแกรมดู เรากลับได้ผลลัพธ์ดังนี้

```
(10, 20)
(30, 50)
(10, 20)
(30, 50)
```

ซึ่งบอกให้เรารู้ว่า swap() ของเราใช้ไม่ได้ เราต้องเปลี่ยนแปลง code ของ swap() ใหม่เพื่อให้การสลับเกิดขึ้นจริง และหลังจากที่เราเปลี่ยน code ให้เป็น

```
private static void swap(Values pointA, Values pointB) {
    Values temp = new Values();

    temp.val1 = pointA.val1;
    temp.val2 = pointA.val2;

    pointA.val1 = pointB.val1;
    pointA.val2 = pointB.val2;

    pointB.val1 = temp.val1;
    pointB.val2 = temp.val2;
}
```

ผลลัพธ์ที่เราได้คือ

```
(10, 20)
(30, 50)
(30, 50)
(10, 20)
```

ซึ่งแสดงถึงการสลับค่าที่ได้ผลจริง สาเหตุที่ swap() ตัวแรกของเราทำงานไม่ได้ ก็เนื่องจากว่า การเปลี่ยนแปลงค่าของ object เกิดขึ้นจาก copy ของ object ทั้งสองที่มาจาก parameter ไม่ใช่ตัว object จริง ๆ ดังนั้นการเปลี่ยนแปลงที่เกิดขึ้นจึงเกิดขึ้นภายในตัว swap() ไม่มีผลกระทบใด ๆ กับ object ภายนอกเลย เราต้องทำการเปลี่ยนค่าให้กับสมาชิกที่อยู่ใน object นั้นโดยตรงดังเช่นที่ swap() ตัวที่สองได้แสดงไว้

เราได้พูดถึง class Student ที่เราสร้างค้างไว้โดยเราได้สำรวจ การสร้าง method ต่าง ๆ รวมไปถึงการส่งค่าเข้าสู่ method ตอนนี้เราจะมาดูถึงรายละเอียดของ method ต่าง ๆ ที่มีอยู่ใน class Student รวมไปถึงโปรแกรมที่เรียกใช้ object ที่เกิดจาก class Student

ก่อนที่เราจะพูดถึง method ที่มีอยู่ใน class Student เรามาดูตัวโปรแกรมที่เรียกใช้ method เหล่านี้ดู

```
//TestStudent.java
```

```
class TestStudent {
    public static void main(String[] args) {
        //create two student objects with given info.
        Student stu1 = new Student("19757", "Jim", "Jones",
                                   "02/10/1962");
        Student stu2 = new Student("20135", "Ray", "Smith",
                                   "12/09/1977");

        //display each student's info.
        stu1.display();
        stu2.display();
        //display total number of students
        System.out.println("Total number of objects created: " +
                           stu1.getCount());

        //access field in class via object
        String date = stu1.dateOfBirth;
        System.out.println("Date of birth of stu1 is " + date);
    }
}
```

เราได้สร้าง object stu1 และ stu2 จาก class Student ด้วยข้อมูลดังที่เห็น

```
Student stu1 = new Student("19757", "Jim", "Jones", "02/10/1962")
Student stu2 = new Student("20135", "Ray", "Smith", "12/09/1977")
```

หลังจากนั้นเราก็เรียกใช้ method display() ในการแสดงผลข้อมูลของ object ทั้งสองตัว โดยที่เราได้ออกแบบ method display() ของเราให้มีการเรียกใช้ method อื่น ๆ ที่มีอยู่ใน class Student ดังนี้

```
public void display() {
    System.out.print(getId() + " ");
    System.out.print(getFirstName() + " ");
    System.out.print(getLastName() + " ");
    System.out.println("Age: " + getAge());
    return ;
}
```

ผู้อ่านจะสังเกตได้ว่าเราเรียก method getId() getFirstName() getLastName() และ method getAge() ภายในประโยคของ System.out.println(…) ซึ่ง method เหล่านี้จะส่งค่าของ field ต่าง ๆ ที่ได้ถูกกำหนดไว้ให้กับประโยคในการแสดงข้อมูลไปยังหน้าจอ

หลังจากที่ข้อมูลของ object ทั้งสองตัวได้ถูกแสดงไปยังหน้าจอแล้ว เราก็เรียกประโยค

```
System.out.println("Total number of objects created: " +
                   stu1.getCount());

String date = stu1.dateOfBirth;
System.out.println("Date of birth of stu1 is " + date);
```

สำหรับการแสดงจำนวนของ object ที่ได้ถูกสร้างขึ้นทั้งหมด และวัน เดือน ปี เกิดของ object ที่ชื่อ stu1 ไปยังหน้าจอ และผลลัพธ์ของการ run ที่ได้คือ

```
19757 Jim Jones Age: 41
20135 Ray Smith Age: 26
Total number of objects created: 2
Date of birth of stu1 is 02/10/1962
```

ผู้อ่านต้องกำหนดให้การเรียก method ที่มีอยู่ใน class ผ่านทาง object ที่ได้สร้างขึ้น ด้วยการใช้ . (dot) ตามด้วยชื่อของ method ที่ต้องการเรียกใช้ เช่น

```
stu1.display();
```

เช่นเดียวกันกับการเรียกใช้ data member เราก็ต้องขึ้นต้นด้วยชื่อของ object ตามด้วย . (dot) ตามด้วยตัวแปรที่ต้องการใช้ เช่น

```
stu1.dateOfBirth;
```

เนื่องจากว่าเราไม่ได้กำหนดนโยบายการเข้าหา ตัวแปร หรือ method ที่มีอยู่ใน class ดังนั้นการเข้าหาจึงไม่ข้อกำหนดใด ๆ เราสามารถที่จะเข้าหาข้อมูลทุกตัวโดยไม่มีข้อจำกัด ก่อนที่เราจะพูดถึงนโยบายการเข้าหาข้อมูลที่อยู่ใน class เราอยากที่จะพูดถึง method getAge() ที่มีอยู่ใน class Student สักเล็กน้อย

method getAge() ได้ถูกออกแบบให้ทำการคำนวณหาอายุด้วยการหาจากการลบปีเกิด ออกจากปีปัจจุบัน ดังนั้นเราจึงต้องเรียกใช้ method จาก class Calendar ในการคำนวณหาปีปัจจุบัน ดังนี้

```
public int getAge() {
    //retrieve a year
    String year = dateOfBirth.substring(6);
    //convert it to an int
    int birthYear = Integer.parseInt(year);

    //get current year from system's calendar
    Calendar now = Calendar.getInstance();
    int thisYear = now.get(Calendar.YEAR);
    //return the difference
    return thisYear - birthYear;
}
```

ก่อนที่เราจะคำนวณหาอายุได้ เราต้องดึงเอา field ที่เป็นปีเกิดที่อยู่ในรูปแบบของ dd/mm/yyyy ออกมาก่อนด้วยการใช้ method substring() ที่เราได้พูดไว้ก่อนหน้านี้ เมื่อได้แล้วเราก็เปลี่ยน string ที่มีค่าเป็นปีเกิดนี้ให้เป็น int ด้วยการเรียกใช้ Integer.parseInt()

หลังจากนั้นเราก็ดึงเอาปีปัจจุบันออกจากเครื่องด้วยคำสั่ง

```
Calendar now = Calendar.getInstance();
int thisYear = now.get(Calendar.YEAR);
```

ซึ่งก่อนที่เราจะดึงปีออกได้เราต้องสร้าง instance จาก class Calendar ก่อนและเมื่อได้แล้วเราก็ใช้ method get() ในการดึงเอาปีปัจจุบันออกมา กระบวนการที่เหลืออยู่ก็คือการส่งส่วนต่างระหว่างปีปัจจุบันกับปีเกิดออกไปยังผู้ที่เรียกใช้ method นี้ (การดึงเอาค่าของปีออก ผู้อ่านต้องระวังในเรื่องของปีที่ใช้ในเครื่อง ณ เวลานั้นด้วยว่าอยู่ในรูปแบบของ พุทธศักราช หรือ คริสต์ศักราช Windows ใหม่ ๆ ยอมให้มีการตั้งค่าตามตำแหน่งที่ตั้งของประเทศนั้น ๆ)

การสร้าง class Student ของเรานั้นเราไม่ได้กำหนดนโยบายการเข้าหา field หรือ method ต่าง ๆ ของเราอย่างไรเลย ผู้อ่านคงสังเกตเห็นว่า ตัวแปรทุกตัวของเราไม่มีคำใด ๆ นำหน้าเลย เช่น private public หรือแม้กระทั่งคำว่า protected หัวข้อต่อไปจะพูดถึงเรื่องของการกำหนดนโยบายการเข้าหาตัวแปร และ method ของ class ว่าจะมีประโยชน์อย่างไรในการออกแบบ class (ผู้อ่านลองตั้งคำถามให้ตัวเองก่อนเริ่มหัวข้อถัดไปว่า ถ้าใส่คำว่า private หน้าตัวแปรทุกตัวจะมีผลต่อโปรแกรม TestStudent.java อย่างไร ถ้าใส่หน้า method ต่าง ๆ หละ?)

การกำหนดนโยบายในการเข้าหาสมาชิกของ class (attributes)

จาก class Circle ของเราที่ได้สร้างขึ้น ในการประกาศตัวแปร radius นั้นเราได้กำหนดให้มีคำว่า private อยู่ด้านหน้าของตัวแปร ดังนี้

```
private double radius;
```

ทำไมเราถึงทำเช่นนี้?

สาเหตุที่เราทำเช่นนี้ก็เพื่อที่จะกำหนดการเข้าหาตัวแปรตัวนี้ โดยทั่วไปแล้วถ้าเราไม่ใส่คำว่า private ไว้หน้าตัวแปรใด เราก็สามารถที่จะเข้าหาตัวแปรตัวนี้ ผ่านทางโปรแกรมอื่น ๆ ที่เรียกใช้ class Circle การกำหนดนโยบายของการเข้าหานั้น เราสามารถทำได้สี่วิธี คือ

1. ไม่ใช้คำใด ๆ นำหน้าตัวแปร (friendly)
2. ใช้คำว่า private นำหน้าตัวแปร (not too friendly)
3. ใช้คำว่า public นำหน้าตัวแปร (friendly)
4. ใช้คำว่า protected นำหน้าตัวแปร (somewhat friendly)

ถ้าเราไม่ใช้คำใด ๆ นำหน้าตัวแปรการเข้าหาตัวแปรตัวนี้จะทำได้เฉพาะ class ที่อยู่ภายในพื้นที่เดียวกัน เช่นภายใน class เดียวกัน หรือภายใน package (ดูเรื่องการสร้างและใช้ package ท้ายบท) เดียวกัน ดังโปรแกรมตัวอย่างที่แสดงให้ดูนี้

```
//Access.java

class Dot {
    //no access control
    String name;
    int value;

    Dot(String s, int v) {
        name = s;
        value = v;
    }
}

class Access {
    public static void main(String[] args) {
        Dot myDot = new Dot("I am a dot.", 10);

        //access both name and value fields from class Dot
        System.out.println("Name is " + myDot.name);
        System.out.println("Value is " + myDot.value);
    }
}
```

เมื่อเรา run ดูผลลัพธ์ที่ได้คือ

```
Name is I am a dot.
Value is 10
```

โปรแกรม Access.java สามารถที่จะเข้าหา field ทั้งสองที่เราสร้างขึ้นใน class Dot ของเราผ่านทาง object ชื่อ myDot (หมายความว่าใคร ๆ ก็เข้าหาตัวแปรของเราได้ถ้าเขารู้ว่าเรามีตัวแปรอะไรบ้าง – อันตราย?)

ทีนี้เราลองนำเอาคำว่า private ไปใส่ไว้หน้าตัวแปรทั้งสองของเรา ดังนี้

```
private String name;
```

```
private int value;
```

การ compile โปรแกรมอีกครั้งหนึ่งจะทำให้เกิด error ดังนี้

```
Access.java:20: name has private access in Dot
       System.out.println("Name is " + myDot.name);
                                               ^
Access.java:21: value has private access in Dot
       System.out.println("Value is " + myDot.value);
                                               ^
2 errors
```

เราไม่สามารถที่จะอ้างถึงตัวแปรทั้งสองได้เพราะคำว่า private เราต้องเข้าหาด้วยวิธีอื่น เช่น เขียน method ที่ส่งค่าของตัวแปรทั้งสองกลับ (โดยเขียนไว้ใน class Dot) เหมือนกับที่เราได้ทำไว้ใน class Circle

เรากำหนดให้ตัวแปร radius ของ class Circle เป็น private data member ดังนั้นเราจึงต้องเขียน method ในการเข้าหาตัวแปร radius ของเรา ดังนี้

```
public double getRadius() {
    return radius;
}
```

เพราะฉะนั้นถ้าเราไม่ต้องการให้โปรแกรม หรือ ผู้ใด (object) เข้าหาตัวแปรของเรา เราก็นำคำว่า private ไปแปะไว้ด้านหน้าของตัวแปรนั้น ๆ ถ้าสังเกตให้ดีจะเห็นว่า method changeRadius() ใน class Circle ก็มีคำว่า private อยู่ด้านหน้าเหมือนกัน ทั้งนี้ก็เพราะว่าเราไม่ต้องการให้ใครเข้าหา method ตัวนี้ยกเว้น method ที่อยู่ภายใน class เดียวกัน การเข้าหา method นี้ผ่านโปรแกรม หรือ object ตัวอื่นจะทำไม่ได้ และ Java compiler จะฟ้องด้วย error ทันที

เราจะกลับมาดูคำว่า protected ในโอกาสต่อไป ทั้งนี้เพราะคำ ๆ นี้มีเงื่อนไขซับซ้อนอยู่พอสมควร สำหรับคำว่า public นั้นจะยอมให้ทุกโปรแกรม หรือ object เข้าหาตัวแปร หรือ method ได้โดยไม่มีเงื่อนไขใด ๆ ทั้งสิ้น

การกำหนดนโยบายการเข้าหาตัวแปรหรือ method ทำให้การควบคุมโปรแกรมทำได้ดียิ่งขึ้น อะไรที่ผู้ใช้ไม่ควรรู้ไม่ควรเห็น เราก็ซ่อนสิ่งเหล่านี้ไว้ การกระทำในลักษณะนี้เราเรียกว่า encapsulation ซึ่งเป็นหนึ่งในข้อกำหนดของการออกแบบและเขียนโปรแกรมในรูปแบบของ OOP ในส่วนตัวของผู้เขียนแล้ว ชอบที่จะใช้ private กับตัวแปรทุกตัว และ method บางตัวที่ไม่มีการเข้าหาจากภายนอก เพราะจะทำให้การใช้ตัวแปรมีข้อจำกัด ส่วนที่เข้าหาข้อมูลเหล่านี้ของเราได้เป็นได้เฉพาะ method ที่เราได้กำหนดไว้เท่านั้น

Constructor อีกครั้ง

Constructor มีไว้สำหรับการกำหนดค่าเบื้องต้นให้กับตัวแปรที่อยู่ใน class และการเรียกใช้ constructor นั้นเราไม่ต้องทำโดยตรง constructor จะถูกเรียกโดยอัตโนมัติผ่านทางการสร้าง object จาก class ดังที่ได้แสดงไว้ในโปแกรมที่สร้าง object จาก class Circle จากประโยคของการสร้างวงกลมที่ว่า

```
Circle red = new Circle(5.0);
```

ขั้นตอนที่เกิดขึ้นเมื่อประโยคนี้ถูก execute คือ

Constructor ของ Circle ถูกเรียกด้วยค่า 5.0 ซึ่งหมายความว่าตัวแปร radius ที่อยู่ใน class Circle จะได้รับค่าที่กำหนดให้ คือ 5.0 หลังจากนั้นตำแหน่ง หรือ address ของ object ที่ถูกสร้างและกำหนดค่านี้จะถูกเก็บไว้ใน red เพื่อให้เห็นภาพชัดขึ้น เราจะเพิ่มประโยคด้านล่างนี้เข้ากับ constructor ของ class Circle

```
System.out.println("Constructor activated with parameter value of " +
                   radius);
```

เพราะฉะนั้น constructor ของ class Circle ก็จะเป็น

```
Circle(double r) {
    radius = r;
    System.out.println("Constructor activated with value of " +
                        radius);
}
```

และเราจะลองสร้าง object สองตัวจาก class Circle ดังนี้

```
class PassByReference {
    public static void main(String[] args) {
        Circle red = new Circle(5.0);
        Circle blue = new Circle(3.5);
}
```

หลังจากที่ compile และ run ผลลัพธ์ที่ได้คือ

```
Constructor activated with value of 5.0
Constructor activated with value of 3.5
```

เราจะเห็นว่า object สองตัวถูกสร้างด้วย radius ที่ต่างกัน คือ 5.0 และ 3.5 และทั้ง ๆ ที่ไม่มีประโยคของการแสดงผลใน method main() ใด ๆ เลย เรากลับได้ผลลัพธ์ดังที่เห็น ทั้งนี้ก็เพราะว่า constructor ของเราที่มีอยู่ใน class Circle ได้รับการเรียกใช้โดยอัตโนมัติ

เราสามารถที่จะสร้าง constructor ให้กับ class ได้มากกว่าหนึ่งตัวทั้งนี้การกระทำดังกล่าวจะต้องอยู่ภายใต้เงื่อนไขที่ Java กำหนดไว้ คือ parameter ที่ส่งไปยัง constructor ต้องไม่มีจำนวนที่เท่ากัน และถ้าจำนวนของ parameter เท่ากัน ชนิดของ parameter ต้องไม่เหมือนกัน (หนังสือเกือบทุกเล่มเรียกขั้นตอนนี้ว่าการกำหนด signature) ลองดูโปรแกรมตัวอย่าง

```
//TestMean.java

class Mean {
    private double x, y, z;

    //constructor with one value
    Mean(double value) {
        x = value;
        y = 0.0;
        z = 0.0;
    }

    //constructor with two values
    Mean(double value1, double value2) {
        x = value1;
        y = value2;
        z = 0.0;
    }

    //constructor with three values
    Mean(double value1, double value2, double value3) {
        x = value1;
        y = value2;
        z = value3;
    }
```

```
    public double findMean() {
        return (x + y + z) / 3;
    }
}
```

class Mean ของเรามี constructor อยู่สามตัว โดยที่ตัวแรกมี parameter หนึ่งตัว constructor ตัวที่สองมี parameter สองตัว และ constructor ตัวที่สามมี parameter สามตัว ภายใน class เราได้สร้าง method findMean() สำหรับการคำนวณหาค่าเฉลี่ยของตัวแปรทั้งสามของ class และเราได้เขียนโปรแกรมสำหรับตรวจสอบการใช้ constructor ดังกล่าวดังนี้

```
class TestMean {
    public static void main(String[] args) {
        //create objects with diferent parameters
        Mean m = new Mean(12.0);
        Mean n = new Mean(10.0, 20.0);
        Mean p = new Mean(10.0, 20.0, 30.0);

        System.out.println("Mean of m is " + m.findMean());
        System.out.println("Mean of n is " + n.findMean());
        System.out.println("Mean of p is " + p.findMean());
    }
}
```

โปรแกรม TestMean.java ประกาศใช้ object จาก class Mean สามตัวด้วยจำนวนของ parameter ที่ต่างกัน คือ หนึ่งตัว สองตัว และสามตัวตามลำดับ หลังจากที่เราเรียก method findMean() สำหรับ object ทั้งสามตัวแล้ว ผลลัพธ์ที่เราได้จากการ run คือ

```
Mean of m is 4.0
Mean of n is 10.0
Mean of p is 20.0
```

จะเห็นว่า Java เลือกใช้ constructor ได้ถูกต้องตามการออกแบบที่เรากำหนดไว้ ซึ่งวิธีการแบบนี้เราเรียกว่า constructor overloading เรามาลองดูการ overload ของ constructor ที่มีจำนวนของ parameter เหมือนกันแต่ต่างชนิดกันดู อีกสักตัวอย่างหนึ่ง

```
//TestMean1.java

class Mean {
    private double x, y, z;

    //constructor with three values
    Mean(double value1, double value2, double value3 {
        x = value1;
        y = value2;
        z = value3;
    }

    //constructor with three values (different type)
    Mean(double value1, int value2, String value3) {
        x = value1;
        y = (double)value2;
        z = Double.parseDouble(value3);
    }

    public double findMean() {
        return (x + y + z) / 3;
    }
}
```

class Mean ของเรากำหนดให้มี constructor สองตัว โดยกำหนดให้ตัวแรกรับ parameter ที่เป็น double ทั้งสามตัว ส่วน constructor ตัวที่สองรับ parameter ที่เป็น double int และ String ตามลำดับ ซึ่งเราได้เปลี่ยนค่าที่ไม่ใช่ double ของทั้งสอง parameter ให้เป็น double ภายในตัว constructor เอง

```
class TestMean1 {
    public static void main(String[] args) {
        //create objects with diferent parameters
        Mean p = new Mean(10.0, 20.0, 30.0);
        Mean q = new Mean(10.0, 20, "40.0");

        System.out.println("Mean of p is " + p.findMean());
        System.out.println("Mean of q is " + q.findMean());
    }
}
```

โปรแกรม TestMean1.java ประกาศการใช้ object สองตัวคือ p และ q จาก class Mean ด้วย parameter ต่างกันดังที่เห็น ผลลัพธ์ที่ได้ของการ run คือ

```
Mean of p is 20.0
Mean of q is 23.0333333333333332
```

เช่นเดียวกับโปรแกรมก่อนหน้านี้ ผลลัพธ์ที่ได้แสดงถึงการเรียกใช้ constructor ที่ถูกต้องของ Java

ยังมีวิธีอื่นที่เราสามารถที่จะ overload constructor ได้ วิธีนี้เราจะเรียก constructor ตัวอื่นจาก constructor อีกตัว

การเรียก constructor จาก constructor

วิธีการเรียก constructor จาก constructor ก็ไม่ยาก เราเพียงแต่กำหนดวิธีการเรียกให้เหมาะสม ซึ่ง Java ยอมให้มีการเรียก constructor จาก constructor ผ่านทางคำว่า this ดังตัวอย่างที่ได้แสดง ดังนี้

```
//TestMean2.java

class Mean {
    private double x, y, z;

    //constructor with one value
    Mean(double value) {
        x = value;
        y = 0.0;
        z = 0.0;
    }

    //constructor with two values
    Mean(double value1, double value2) {
        //calling constructor with one argument
        this(value1);
        y = value2;
    }

    //constructor with three values
    Mean(double value1, double value2, double value3) {
        //calling constructor with two arguments
        this(value1, value2);
        z = value3;
    }
```

```
    public double findMean() {
        return (x + y + z) / 3;
    }
}

class TestMean2 {
    public static void main(String[] args) {
        //create objects with diferent parameters
        Mean m = new Mean(12.0);
        Mean n = new Mean(10.0, 20.0);
        Mean p = new Mean(10.0, 20.0, 30.0);

        System.out.println("Mean of m is " + m.findMean());
        System.out.println("Mean of n is " + n.findMean());
        System.out.println("Mean of p is " + p.findMean());
    }
}
```

เราได้ดัดแปลงโปรแกรมของการหาค่าเฉลี่ยของเลขสามตัวก่อนหน้านี้ให้ มี constructor ที่เรียกใช้ constructor ดังนี้

```
Mean(double value) {
    x = value;
    y = 0.0;
    z = 0.0;
}

Mean(double value1, double value2) {
    //calling constructor with one argument
    this(value1);
    y = value2;
}

Mean(double value1, double value2, double value3) {
    //calling constructor with two arguments
    this(value1, value2);
    z = value3;
}
```

constructor ตัวแรกกำหนดค่าให้กับตัวแปรเช่นเดียวกับที่ได้ทำไว้ก่อนหน้านี้ ส่วน constructor ตัวที่สอง และสามเรียกใช้ constructor ที่มีอยู่ภายใน class Mean ด้วยการเรียกผ่าน this() โดยที่ constructor ตัวที่สองเรียก constructor ตัวแรกด้วยคำสั่ง this(value1) เมื่อ Java เห็นการเรียกเช่นนี้ ก็จะไปหา constructor ที่มีอยู่ใน class ว่ามีตัวไหนบ้างที่มีเงื่อนไขตามที่ถูกเรียก ซึ่งก็เป็นไปตามนั้นคือ Java ไปเรียก constructor ที่มี parameter เพียงตัวเดียวมาทำการกำหนดค่าเบื้องต้นให้กับ object ที่ execute constructor ตัวนี้ คือ x จะมีค่าเป็น value1 y และ z จะมีค่าเป็น 0.0 จากการกำหนด หลังจากนั้นการกำหนดค่าให้ y เป็น value2 ก็ได้ถูกกระทำภายใน constructor ตัวที่สอง

constructor ตัวที่สามก็เช่นเดียวกัน การเรียก constructor ที่มี parameter 2 ตัวก็เกิดขึ้น กระบวนการเดิมที่ได้กล่าวไว้ก่อนหน้านี้ก็ได้ถูกกระทำขึ้น เมื่อทำเสร็จแล้ว การกำหนดค่าให้กับตัวแปร z จึงมีขึ้น (ผลลัพธ์ของโปรแกรม TestMean2.java ก็เหมือนกับที่เราได้ก่อนหน้านี้)

การเรียกใช้ constructor ที่มีอยู่แล้วทำให้การเขียน code ที่ซ้ำซ้อนลดลง การเขียนโปรแกรมก็มีประสิทธิภาพมากขึ้น

การ overload method

เนื่องจากว่า constructor ก็เป็น method ชนิดหนึ่งดังนั้นถ้าเราสามารถที่จะ overload constructor ได้ เราก็สามารถที่จะ overload method ได้เช่นกัน เพราะฉะนั้นเราจะมาดูเรื่องการ overload method ซึ่งก็ไม่แตกต่างจากการ overload constructor เท่าใดนัก เรามาเริ่มต้นด้วย method ที่แสดงผลออกทางหน้าจอ จากตัวแปรชนิดต่าง ๆ ที่ส่งเข้าไปยัง method นี้

```
//Overload.java

class Ball {
    private double x, y, z;

    //constructor
    Ball() {
        x = y = z = 0.0;
    }

    //display coordinates
    public void display() {
        System.out.println("(" + x + ", " + y + ", " + z + ")");
    }
}

class Overload {
    public static void main(String[] args) {
        Ball ball = new Ball();          //create object

        display(25);                     //display int
        display(350.0);                  //display double
        display("I love overloading."); //display string
        display('A');                    //display char
        display(ball);                   //display object
    }

    //accept int as a parameter
    public static void display(int data) {
        System.out.println(data);
    }

    //accept double as a parameter
    public static void display(double data) {
        System.out.println(data);
    }

    //accept string as a parameter
    public static void display(String data) {
        System.out.println(data);
    }

    //accept char as a parameter
    public static void display(char data) {
        System.out.println(data);
    }

    //accept object as a parameter
    public static void display(Ball blue) {
        blue.display();   //calling display() from class Ball
    }
```

```
}
```

โปรแกรม Overload.java สร้าง method display() เพื่อรองรับ parameter ต่าง ๆ จำนวน 5 ตัว คือ รับ int รับ double รับ string รับ char และรับ object และก็เช่นเดียวกับการ overload constructor method ที่เราเขียนขึ้นต่างก็มี signature ที่ไม่เหมือนกัน ดังนั้น java ก็สามารถหา method ที่ถูกต้องตามการเรียกใช้ ผลลัพธ์ที่ได้จากการ run โปรแกรมคือ

```
25
350.0
I love overloading.
A
(0.0, 0.0, 0.0)
```

การใช้ overload นั้นมีข้อดีในการเขียนโปรแกรมคือ เราไม่จำเป็นต้องหาชื่ออื่น ๆ ที่เหมาะสมกับหน้าที่ของ method นั้น ๆ นอกเหนือจากที่เราได้เลือกไว้แล้ว และการใช้ชื่อเดียวเพื่อรองรับ parameter หลายตัวทำให้ผู้ใช้ไม่ต้องค้นหา method ตัวอื่น ที่ทำหน้าที่แบบเดียวกัน ตัว Java เองก็ได้สร้าง method ไว้มากมายที่มีการใช้ overload ในการสร้างเพื่อรองรับการใช้งานของ user ที่มีความหลากหลายของข้อมูลที่ส่งเข้าไปยัง method นั้น ๆ ทำให้ผู้ใช้ไม่ต้องเสียเวลาในการค้นหา method ที่ตัวเองต้องการใช้

ผู้อ่านควรฝึกการเขียนโปรแกรมด้วยการใช้ overload method เพื่อให้เกิดความชำนาญ และเข้าใจถึงวิธีการของ method overload มากยิ่งขึ้น

การสร้าง class ภายใน class (nested class)

ตัวอย่างของการสร้างและเรียกใช้ class ที่เราได้ทราบถึงก่อนหน้านี้ เป็นการสร้าง class ที่มีเพียงเฉพาะตัวแปร และ method อยู่ภายใน class เท่านั้น ยังไม่มีข้อมูลภายใน class ที่ตัวมันเองก็เป็น class เช่นเดียวกัน เราจะมาดูถึงวิธีการสร้าง class ภายใน class อย่างง่าย ๆ เพื่อให้ได้รับทราบว่า Java ยอมให้มีการสร้าง class ภายใน class เช่นเดียวกันกับที่เราสามารถสร้าง method อื่น ๆ

เราจะลองเขียนโปรแกรมตัวอย่างแบบง่าย ๆ ที่ class ซ้อนกันอยู่

```
//NestedClass.java

class Parent {
    private String name;
    Child c = new Child("Galvin");   //create a Child object

    Parent(String n) {
        name = n;
    }

    public void showName() {
        System.out.println("Parent: " + name);
        c.showName();  //calling Child's showName()
    }

    class Child {
        private String name;
        //create a GrandChild object
        GrandChild gc = new GrandChild("Jeff");
        Child(String n) {
            name = n;
        }

        public void showName() {
            System.out.println("Child: " + name);
            gc.showName(); //calling GrandChild's showName()
```

```
        }

        class GrandChild {
            private String name;

            GrandChild(String n) {
                name = n;
            }

            public void showName() {
                System.out.println("GrandChild: " + name);
            }
        }//class GrandChild
    }//class Child
}//class Parent

class NestedClass {
    public static void main(String[] args) {
        Parent p = new Parent("Rebecca");

        p.showName();
    }
}
```

โปรแกรม NestedClass.java สร้าง object จาก class Parent หนึ่งตัว แสดงข้อมูลที่อยู่ใน class ผ่านทาง method showName() ซึ่งให้ผลลัพธ์ ดังนี้ คือ

```
Parent: Rebecca
Child: Galvin
GrandChild: Jeff
```

class Parent ของเราได้สร้าง class อีก class หนึ่งที่อยู่ภายในชื่อ Child และ class Child ก็ได้สร้าง class อีก class หนึ่งชื่อ GrandChild โดยทุก class จะมี method ชื่อ showName() ซึ่งเอาไว้ใช้ในการแสดงค่าของตัวแปรของแต่ละ class ไปยังหน้าจอ เราได้สร้าง object จาก class Child ภายใน class Parent ชื่อ c และเราเรียก method showName() ของ class Child จาก method showName() ของ class Parent ในทำนองเดียวกันเราก็ได้สร้าง object จาก class GrandChild ภายใน class Child และก็เรียก method showName() ของ class GrandChild จาก method showName() ของ class Child ด้วย เพราะฉะนั้นขั้นตอนของการ execute คือ

Parent.showName() → Child.showName() → GrandChild.showName()

เราสามารถที่จะสร้าง object จาก class ที่อยู่ด้านในผ่านทาง object ที่อยู่ด้านนอก ดังเช่นตัวอย่างนี้

```
class NestedClass {
    public static void main(String[] args) {
        Parent p = new Parent("Rebecca");

        p.showName();
        Parent.Child newChild = p.new Child("John");
        Parent.Child.GrandChild newGrandChild =
                                  newChild.new GrandChild("Jen");
        newChild.showName();
        newGrandChild.showName();
    }
}
```

เราเพิ่มประโยคใหม่เข้ากับ class NestedClass ของเราเพื่อแสดงการสร้าง object จาก class ที่อยู่ภายในประโยค

```
Parent.Child newChild = p.new Child("John");
```

สร้าง object จาก class Child ผ่านทาง object ที่มาจาก class Parent ด้วยการเรียกใช้ p.new Child("John") เราต้องทำการสร้างผ่านทาง object ของ class Parent เท่านั้นเราไม่สามารถที่จะสร้าง object จาก class Child ผ่านทาง class Parent โดยตรง เช่น ประโยค

```
Parent.Child newChild = Parent.new Child("John");
```

จะไม่ได้รับการ compile และ Java จะฟ้องด้วย error ทันที

การสร้าง object จาก class GrandChild ผ่านทาง class ที่อยู่ภายนอกทั้งสองก็คล้าย ๆ กัน

ผลลัพธ์ของการ run โปรแกรมที่ถูกดัดแปลงแล้ว คือ

```
Parent: Rebecca
Child: Galvin
GrandChild: Jeff
Child: John
GrandChild: Jeff
GrandChild: Jen
```

เนื่องจากว่าเราสร้าง object ของ class Child ผ่านทาง class Parent ดังนั้นเมื่อเรียกใช้ method showName() เราจึงได้ผลลัพธ์

```
Child: John
GrandChild: Jeff
```

อีกครั้งหนึ่ง ส่วน object ที่สร้างขึ้นใหม่จาก class GrandChild เมื่อเรียก method showName() เราจึงได้เพียงแต่ GrandChild: Jen เท่านั้น

ตัวอย่างโปรแกรมการสร้าง nested class ที่เห็นเป็นเพียงแค่ตัวอย่างที่แสดงให้ดูถึงการสร้าง nested class เท่านั้น การออกแบบ nested class ต้องคำนึงถึงความสัมพันธ์ของทั้ง class ที่อยู่ด้านนอก และ class ที่อยู่ด้านในว่ามีความสัมพันธ์กันอย่างไร ถ้าความสัมพันธ์นั้นสามารถที่จะเขียน class แยกกันออกมาได้ การออกแบบก็ควรให้เป็น class ที่แยกกันอยู่ เพราะจะทำให้การออกแบบทำได้ง่ายกว่า การสร้างและเรียกใช้ object ก็จะไม่ซับซ้อนเท่าใดนัก

มาถึงตอนนี้เราได้ทราบถึงการสร้าง class การออกแบบ method และการสร้าง object จาก class พอสมควร ตัวอย่างที่จะแสดงให้ดูต่อไปเป็นความพยายามที่จะออกแบบ class Array ขึ้นมาใช้งาน เพื่อที่จะแสดงให้เห็นถึงศักยภาพของการใช้ Java ในการเขียนโปรแกรม

class MyArray ที่เราสร้างขึ้นจะกำหนดให้เป็น array ที่ใช้เก็บ Object เพื่อให้มีความยืดหยุ่นในการทำงานกับข้อมูลต่าง ๆ ทั้งนี้และทั้งนั้น class MyArray เป็นเพียงแค่ตัวอย่างตัวอย่างหนึ่งเท่านั้น ผู้อ่านสามารถที่จะทำได้มากกว่าที่โปรแกรมตัวอย่างนี้แสดงให้ดู

```
//MyArray.java

class MyArray {
    Object []arr;  //array of objects
    int maxSize;   //maximum size of array
    int count;     //number of items in array

    //default constructor
    MyArray() {
```

```
        count = 0;                  //no item as yet
        maxSize = 10;               //default maximum size
        arr = new Object[maxSize];  //allocate space for arr
    }

    //constructor
    MyArray(int size) {
        count = 0;           //no item as yet
        maxSize = size;          //set maximum size
        arr = new Object[maxSize];  //allocate space for arr
    }

    //method to add item into arr
    public void add(Object item) {
        //expand capacity if arr is full
        //double the size
        if(count == maxSize) {
            //double arr's size
            maxSize *= 2;
            //create temporary array
            Object []temp = new Object[maxSize];
            //copy items into new array: temp
            System.arraycopy(arr, 0, temp, 0, arr.length);
            //refer arr to temp
            arr = temp;
        }
        arr[count++] = item;  //add new object into arr
    }

    //overload method add to accomodate double
    public void add(double item) {
        add(String.valueOf(item));
    }

    //overload method add to accomodate int
    public void add(int item) {
        add(String.valueOf(item));
    }

    //return formatted output when user uses System.out.println()
    //10 items per line
    public String toString() {
        //create new buffer to hold data
        StringBuffer buffer = new StringBuffer();
        for(int i = 0; i < maxSize; i++) {
            //append newline if this line already has 10 items
            if(i % 10 == 0)
                buffer.append("\n");
            //stop appending when there is no item left (null)
            if(arr[i] == null)
                break;
            //append tab + item in arr
            buffer.append("\t" + arr[i]);
        }
        //returning an output String
        return new String(buffer + "\n");
    }
}
```

เรากำหนดให้มี constructor สองตัว โดยที่ตัวแรกเป็น default constructor ที่กำหนดให้ขนาดของ array มีความจุครั้งแรกได้เพียง 10 ตัว ส่วนตัวที่สองเป็น constructor ที่ผู้ใช้สามารถกำหนดขนาดของ array ได้ เราได้ออกแบบให้ array ของเราเป็น array แบบยืดหยุ่น หรือที่เรียกว่า dynamic array โดยเรากำหนดให้มีการขยายขนาดของ array ให้เป็นสองเท่าของขนาดเดิมถ้ามีการใส่ข้อมูลเกินจำนวนที่ array สามารถรองรับได้ ใน method add()

```
public void add(Object item) {
    //expand capacity if arr is full
    //double the size
    if(count == maxSize) {
        //double arr's size
        maxSize *= 2;
        //create temporary array
        Object []temp = new Object[maxSize];
        //copy items into new array: temp
        System.arraycopy(arr, 0, temp, 0, arr.length);
        //refer arr to temp
        arr = temp;
    }
    arr[count++] = item;  //add new object into arr
}
```

method add() ของเราจะตรวจสอบค่าของ count (จำนวนของข้อมูลที่มีอยู่ใน array) กับ maxSize (ความจุสูงสุดที่สามารถเก็บข้อมูลได้) ว่าเป็นค่าเดียวกันหรือไม่ ถ้าเป็นค่าเดียวกันเราจะขยายขนาดของ array ออกเป็นสองเท่า ด้วยการสร้าง array ตัวใหม่ชื่อ temp ที่มีความยาวเป็นสองเท่าของ array ตัวเดิม

```
maxSize *= 2;
Object []temp = new Object[maxSize];
```

หลังจากนั้นเราก็ทำการโยกย้ายข้อมูลจาก array ตัวเก่าสู่ array temp ด้วยการใช้คำสั่ง

```
System.arraycopy(arr, 0, temp, 0, arr.length);
```

คำสั่งนี้จะทำการย้ายข้อมูลจาก arr เริ่มต้นที่ index = 0 ไปสู่ temp เริ่มต้นที่ index 0 จนกว่าจะหมดความยาวที่กำหนดให้ (arr.length) ซึ่งก็คือจำนวนของข้อมูลที่อยู่ใน arr ทั้งหมด

เมื่อโยกย้ายข้อมูลเสร็จแล้วเราก็อ้างถึง array ตัวใหม่ด้วย array ตัวเดิมของเราคือ arr ด้วยคำสั่ง

```
arr = temp;
```

เรายังได้ทำการ overload method add() ของเราเพื่อให้สามารถรองรับการเพิ่มข้อมูลที่เป็น double และ int เข้าสู่ array ของเราได้ วิธีการแบบนี้ไม่ค่อยจะดีสักเท่าไร เพราะเป็นการนำเอาข้อมูลที่ไม่ใช่ชนิดเดียวกัน (โดยตรง) เข้าไปเก็บไว้ใน array ซึ่งถ้าหากมีกระบวนการอื่น ๆ ที่เราต้องการทำกับข้อมูลเหล่านั้น เราจะต้องเสียเวลาในการตรวจสอบชนิดของข้อมูลทุกตัวที่มีอยู่ใน array ของเรา

เราทำการ overload ด้วยการใช้ method ของ class String คือ String.valueOf() ดังที่เห็นนี้

```
public void add(double item) {
    add(String.valueOf(item));
}
```

เนื่องจากว่า String เป็น Object ชนิดหนึ่งดังนั้นการ add String เข้าสู่ array ของเราจึงไม่มีปัญหาอะไร หลังจากนั้นเราได้เขียนโปรแกรมเพื่อตรวจสอบ ดังนี้

```
//TestMyArray.java

import java.lang.Integer;

class TestMyArray {
    public static void main(String[] args) {
        //create array of 10 objects
        MyArray array = new MyArray(10);

        //add 40 objects into array (randomized)
        int count = 0;
        while(count < 40) {
            int data = (int)(Math.random() * 100 + 1);
            array.add(new Integer(data));
            ++count;
        }
        System.out.println(array);   //display contents

        array.add(45.23);            //add double
        array.add(88.23);

        array.add(89);               //add int
        array.add(12);
        System.out.println(array);
    }
}
```

เราก็ได้ผลลัพธ์ ดังนี้

```
    9   76  20  86  98  12  27  35  41  31
    54  65  83  3   36  96  24  63  19  75
    3   91  87  72  3   12  7   46  96  11
    1   70  27  47  14  51  22  74  95  98


    9   76  20  86  98  12  27  35  41  31
    54  65  83  3   36  96  24  63  19  75
    3   91  87  72  3   12  7   46  96  11
    1   70  27  47  14  51  22  74  95  98
    45.23   88.23   89  12
```

ผู้อ่านควรสังเกตถึงผลลัพธ์ที่ได้โดยเฉพาะข้อมูลในบรรทัดสุดท้าย (ที่เราได้เพิ่มข้อมูลที่เป็น double และ int หลังจากการ display ครั้งแรก)

ส่วนประกอบที่สำคัญอีกส่วนหนึ่งที่ต้องกล่าวไว้ในที่นี้ คือ method toString() และก่อนที่จะพูดถึง method toString() เราควรดูประโยคอีกประโยคหนึ่งก่อน นั่นก็คือ

```
System.out.println(array);
```

ประโยค System.our.println() จะรับเฉพาะข้อมูลที่เป็น String หรือ primitive datatype ที่เราได้พูดถึงก่อนหน้านี้ แต่สิ่งที่เราส่งไปให้เป็น object จาก class MyArray ทำไมเราถึงทำได้?

เราส่ง Object ไปให้ System.out.println() ได้ก็เพราะว่าเราได้เขียน method toString() รองรับไว้เรียบร้อยแล้ว เมื่อไรก็ตามที่ Java เห็นข้อมูลที่ไม่ใช่ String หรือ primitive datatype Java จะทำการค้นหา method toString() ถ้ามี Java ก็จะใช้ method toString() นั้น แต่ถ้าไม่มีก็จะฟ้องด้วย error ทันที เรามาดูกันว่าเราเขียน method toString() อย่างไร

```
public String toString() {
    //create new buffer to hold data
    StringBuffer buffer = new StringBuffer();
    for(int i = 0; i < maxSize; i++) {
        //append newline if this line already has 10 items
        if(i % 10 == 0)
            buffer.append("\n");
        //stop appending when there is no item left (null)
        if(arr[i] == null)
            break;
        //append tab + item in arr
        buffer.append("\t" + arr[i]);
    }
    //returning an output String
    return new String(buffer + "\n");
}
```

ภายใน method toString() เราใช้ StringBuffer ในการเก็บส่วนประกอบต่าง ๆ ที่เราต้องการส่งกลับออกไป ซึ่งจะประกอบไปด้วย

- ข้อมูลต่าง ๆ ที่อยู่ใน array (arr[i])
- tab (\t)
- newline (\n)

เรานำข้อมูลเหล่านี้ไปเก็บไว้ใน buffer ด้วยการใช้ method append() ทุก ๆ ครั้งที่เราได้ข้อมูลครบ 10 ตัวเราจะทำการ append newline เข้าสู่ buffer ถ้าหากว่าเราได้ขยายขนาดของ array แต่ไม่ได้ใส่ข้อมูลครบตามขนาดที่ได้ขยายขึ้น เราจะยุติการนำข้อมูลเข้าสู่ buffer เนื่องจากว่าข้อมูลเหล่านี้มีค่าเป็น null และเราก็ไม่ต้องการแสดงค่าเหล่านี้ เมื่อเราได้ข้อมูลทั้งหมดที่ต้องการแล้ว เราก็ส่งกลับออกไป โดยส่งออกไปในรูปแบบของ String (เพราะ System.out.println() ต้องการ String) ด้วยคำสั่ง

```
return new String(buffer + "\n");
```

ผู้อ่านสามารถนำเอาเทคนิคนี้ไปใช้กับการแสดงข้อมูลอื่น ๆ ผ่านทาง System.out.println() ได้โดยไม่มีข้อจำกัดใด ๆ ทั้งสิ้น

เราสามารถที่จะสร้าง class เพื่อการใช้งานของเราโดยไม่มีขีดจำกัดใด ๆ ทั้งนี้ก็ต้องขึ้นอยู่กับลักษณะงานนั้น ๆ ว่าอยู่ในรูปแบบไหน ความยาก ง่ายของงานเป็นอย่างไร ตัวอย่างการสร้าง array ขึ้นมาใช้งานเป็นเพียงตัวอย่างเล็ก ๆ เพียงตัวอย่างเดียวที่เราได้แสดงให้ดูถึงข้อดีของภาษา Java ยังมีสิ่งอื่น ๆ ที่ผู้อ่านสามารถที่จะสร้างขึ้นมาสำหรับงานของตัวเองได้

Package

Package เป็นที่รวมของ class ต่าง ๆ ที่ถูกเรียกใช้ในโปรแกรม หรือ method ต่าง ๆ class ทุก ๆ ตัวของ Java ได้ถูกจัดถูกเก็บอยู่ใน package ต่าง ๆ เช่น java.lang หรือ java.io การเอา class ต่าง ๆ มารวมกันไว้ใน package เดียวกันทำให้การเรียกใช้ class ต่าง ๆ เหล่านี้ทำได้ง่ายขึ้น ถ้าผู้อ่านมองย้อนกลับไปดูโปรแกรมตัวอย่างต่าง ๆ ที่เราเขียนขึ้น ก็จะเห็นการเรียกใช้ package เหล่านี้ในโปรแกรมหลาย ๆ ตัว เราจะมาดูถึงวิธีการสร้าง package เพื่อเก็บ class และการเรียกใช้ package เหล่านี้ในโปรแกรมตัวอย่าง

การสร้าง package

การสร้าง package นั้นไม่ยาก ขั้นตอนก็ไม่ซับซ้อนเท่าไร ขั้นตอนแรกที่เราต้องทำก็คือ ใส่คำว่า package ตามด้วย ชื่อของ package นั้นก่อนประโยคใด ๆ ในไฟล์ที่เก็บ class นั้น ๆ ที่เราสร้างขึ้น (ยกเว้นประโยคที่เป็น comment) เช่น

```
//package Shape
package Shape;
```

```
public class Rectangle {
    …
    …
}
```

ถ้าเราไม่ใส่คำว่า public ไว้ด้านหน้าของ class หรือ method ที่อยู่ใน package ที่เราสร้างขึ้น โปรแกรมหรือ class ตัวอื่น ๆ ที่อยู่นอก package ก็ไม่สามารถที่จะเรียกใช้ class นี้ได้ ยกเว้น class อื่น ๆ ที่อยู่ใน package นี้เท่านั้น package ตัวอย่างของเราใช้ชื่อว่า Sample โดยเรากำหนดให้ Sample ประกอบไปด้วย class One และ class Two และเราได้สร้าง package อีกตัวหนึ่งใน Sample ให้ชื่อว่า oneMore และใน package oneMore นี้เราจะเก็บ class Three ไว้ ดังที่เห็นจาก code ข้างล่างนี้

```
//sample package
package Sample;

public class One {
    public One() {
        System.out.println("This is class One in Sample package.");
    }
}
```

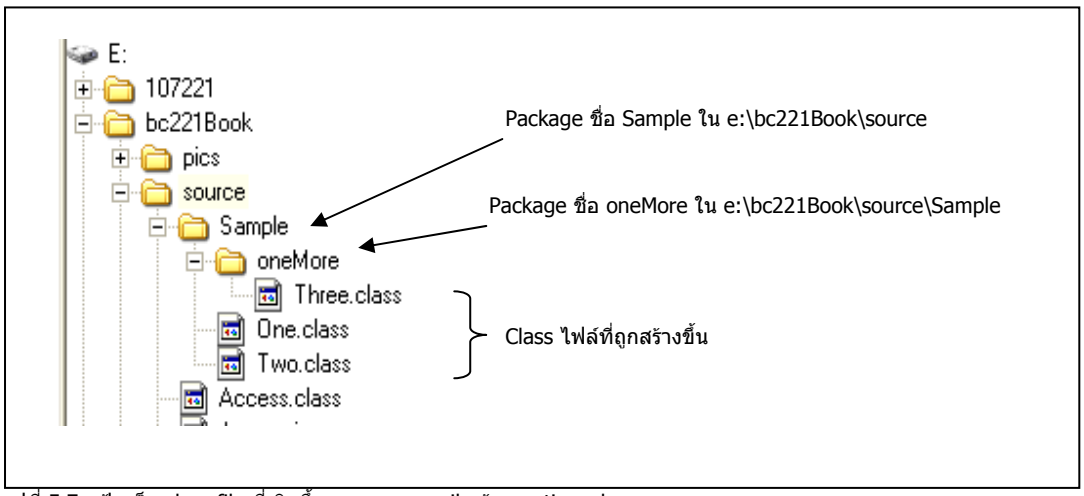
```
//sample package
package Sample;

public class Two {
    public Two() {
        System.out.println("This is class Two in Sample package.");
    }
}
```

```
//sample package
package Sample.oneMore;

public class Three {
    public Three() {
        System.out.println("This is class Three in Sample.oneMore
package.");
    }
}
```

ในการ compile ไฟล์ทั้งสามที่ได้สร้างขึ้นนี้ เราใช้คำสั่ง javac -d e:\bc221Book\source ตามด้วยชื่อไฟล์ เช่น javac -d e:\bc221Book\source One.java

เราต้องใช้ option -d ในการ compile ไฟล์ทั้งสามเพื่อบอกให้ Java รู้ถึงแฟ้มปลายทางที่เราต้องการเก็บ class ไฟล์ที่เกิดขึ้นจากการ compile ในที่นี้เราจะเก็บ class ไฟล์ไว้ใน e:\bc221Book\source

ภายในไฟล์ One.java และ Two.java เราได้ใช้คำสั่ง package Sample เป็นตัวบอกถึงที่เก็บ class ไฟล์ และในตัวไฟล์ Three.java เราใช้คำสั่ง package Sample.oneMore เพราะฉะนั้น class ไฟล์ที่เกิดขึ้นจะถูกเก็บไว้ในแฟ้ม (directory) เหล่านี้ ดังที่ได้แสดงไว้ในรูปที่เห็นด้านล่างนี้



รูปที่ 5.7 แฟ้มเก็บ class file ที่เกิดขึ้นจากการ compile ด้วย option -d

หลังจากนั้นเราก็สร้างโปรแกรมตรวจสอบการเรียกใช้ package ที่เราได้สร้างขึ้น โดยที่เรากำหนดให้มีการเรียกใช้ class ต่าง ๆ ด้วยการใช้คำสั่ง import ดังนี้

```
//test the package

import Sample.One;
import Sample.Two;
import Sample.oneMore.Three;

class TestPackage {
    public static void main(String[] args) {
        One object1 = new One();
        Two object2 = new Two();
        Three object3 = new Three();
    }
}
```

หลังจากนั้นเราก็ compile โปรแกรม TestPackage.java ด้วยคำสั่ง

```
javac -classpath e:\bc221Book\source TestPackage.java
```

โดยเราใช้ option –classpath ตามด้วยที่อยู่ของ class ต่าง ๆ ที่เราได้สร้างขึ้นในการ compile และเมื่อทดลอง run ดูผลลัพธ์ที่ได้คือ

```
This is class One in Sample package.
This is class Two in Sample package.
This is class Three in Sample.oneMore package.
```

ในการสร้าง package นั้น Java จะไปสร้างแฟ้มที่มีชื่อเหมือนกับที่เรากำหนดไว้ ดังที่ได้เห็นในตัวอย่าง และเราสามารถที่จะกำหนด sub-directory ด้วยการใช้ . (dot) เช่นที่เราได้ทำไว้ (Sample.oneMore) และเราต้องไม่ลืมใช้คำว่า public นำหน้า class และ method ต่าง ๆ ที่เราต้องการให้โปรแกรม หรือ class หรือ method อื่น ๆ ที่อยู่นอก package ใช้

เราไม่จำเป็นต้อง compile ด้วยการใช้คำสั่งที่มี option –classpath ในการ compile โปรแกรมที่เรียกใช้ package ต่าง ๆ ของ Java เพราะ Java ได้จัดการเรื่องต่าง ๆ เหล่านี้ให้เราเรียบร้อยแล้ว

ถ้าเราใช้ class ต่าง ๆ ที่อยู่ใน package ที่สร้างขึ้นบ่อยครั้ง หรือทุกครั้ง เราก็สามารถที่จะกำหนดให้ Java ค้นหา class ต่าง ๆ เหล่านี้โดยไม่ต้องกำหนด option –d ในการ compile เราเพียงแต่กำหนด classpath ไว้ก่อนการ compile ครั้งแรกเพียงครั้งเดียว ดังนี้

```
set classpath=e:\bc221Book\source
```

ถ้าเราไม่อยากที่จะกำหนด classpath ทุกครั้งที่เราต้อง compile เราก็กำหนด classpath ไว้ในไฟล์ autoexec.bat สำหรับ Windows รุ่นเก่า ๆ ส่วน Windows รุ่นใหม่เราต้องกำหนด classpath ไว้ใน environment variables (คล้าย ๆ กับการกำหนด path ของ Java ในบทที่หนึ่ง) ถ้าเรามี package อื่น ๆ ที่เราต้องการใช้ เราก็ใช้ ; เป็นตัวแบ่ง classpath เช่น

```
set classpath=e:\bc221Book\source; c:\myOthers\Shape
```

สรุป

ในบทนี้เราได้สำรวจถึงวิธีการสร้าง class การสร้าง object การสร้าง method และการสร้าง method และ constructor ที่มี signature ที่แตกต่างกันในการรองรับการเรียกใช้ด้วย parameter ที่ต่างกัน การกำหนดนโยบายการเข้าหาข้อมูลที่อยู่ใน class รวมไปถึงการสร้าง nested class และการสร้าง package โดยสรุปแล้วเราได้ทราบถึง

- ✓ ส่วนประกอบของ class
- ✓ ชื่อของ class จะต้องเป็นชื่อเดียวกันกับๆฟล์ที่เก็บ class นั้นอยู่
- ✓ การประกาศตัวแปรที่เรียกว่า class variable ต้องใช้คำว่า static นำหน้า
- ✓ ตัวแปรที่เป็น instance variable สามารถที่จะเข้าหาได้ผ่านทาง object ที่เกิดจาก class นั้น
- ✓ การสร้าง method จะต้องบอกถึงสิ่งที่ต้องส่งกลับ ชนิดและจำนวนของ parameter (ถ้ามี)
- ✓ Method ที่ประกาศให้เป็น static สามารถถูกเรียกใช้ได้ถึงแม้ว่าจะไม่มี object ใด ๆ ถูกสร้างขึ้น
- ✓ การเข้าหาข้อมูลใน class มีสามแบบคือ private public และ protected (ไม่นับแบบที่ไม่มีคำใด ๆ นำหน้า)
- ✓ การสร้าง class สามารถที่จะสร้างให้อยู่ใน class อื่นได้
- ✓ การสร้าง และ การใช้ package จะต้องใช้คำว่า public นำหน้า class หรือ method ที่ต้องการให้ผู้อื่นที่อยู่นอก package ใช้

แบบฝึกหัด

1. จงออกแบบ class ที่ใช้เก็บข้อมูลเกี่ยวกับ นักศึกษา เช่น รหัส ชื่อ-นามสกุล ที่อยู่ และข้อมูลอื่น ๆ ที่เห็นว่าเหมาะสม ให้เขียนโปรแกรมทดสอบการสร้าง object ต่าง ๆ จาก class ที่ได้สร้างขึ้นนี้

2. จงออกแบบ class ที่ใช้เก็บข้อมูลที่เกี่ยวข้องกับน้ำหนัก เช่น ตัน กิโลกรัม และ กรัม โดยให้ออกแบบให้มี constructor ที่รองรับการสร้าง object ด้วยจำนวน parameter ที่ต่างกัน ชนิดของ parameter ที่ต่างกัน รวมไปถึงการออกแบบ method ที่ทำการบวก การลบ object ต่าง ๆ ที่ได้ถูกสร้างขึ้น ให้เขียนโปรแกรมทดสอบ

3. จาก class MyArray ที่สามารถเก็บข้อมูลที่เป็น object ต่างชนิดกันได้ จากตัวอย่างที่ให้ไว้ จงเขียน method ที่ส่งจำนวน object ที่มีอยู่ใน array ให้ชื่อว่า getCount() และ method dataAt() ที่ส่งค่าของข้อมูล ณ ตำแหน่งที่กำหนดให้กลับไปยังผู้เรียก เช่น ถ้าเรียกด้วย

   Object obj = arr.dataAt(7);

   จะส่งข้อมูล ณ ตำแหน่งที่ 7 มาให้กับตัวแปร obj

   ให้เขียนโปรแกรมทดสอบ method ที่เขียนขึ้น

4. จงเขียนโปรแกรมที่มี method ในการคำนวณหาพื้นที่ของวงกลมที่มี parameter ต่างชนิดกัน เช่น ผู้เรียกใช้ method อาจส่งข้อมูลที่มีชนิดเป็น int double float หรือ String ที่เป็นตัวเลข เช่น "234" ให้เขียนโปรแกรมตรวจสอบ method เหล่านี้

5. จงเขียนโปรแกรมที่ตรวจสอบการใช้ overloaded constructor ในการกำหนดค่าให้กับ instance variable ที่มีอยู่ใน class นั้น ๆ ให้เขียนโปรแกรมทดสอบ

6. จงออกแบบ class สำหรับเก็บข้อมูลที่เกี่ยวข้องกับนักศึกษา เช่น รหัสประจำตัว ชื่อ นามสกุล ที่อยู่ และข้อมูลอื่น ๆ ที่จำเป็น รวมไปถึง method ต่าง ๆ ที่ใช้ในการกำหนดค่า หรือประมวลผลเกรดเฉลี่ย และ method สำหรับการแสดงผลไปยังหน้าจอ

7. จงออกแบบ class สำหรับเก็บข้อมูลของ วัน เดือน ปี ในรูปแบบของ dd:mm:yyyy โดยกำหนดให้ข้อมูลนี้นำเข้ามาจาก keyboard ให้เขียน Method ที่ทำหน้าที่ในการเปลี่ยนข้อมูลนี้ให้อยู่ในรูปแบบที่ขึ้นต้นด้วย วัน ตามด้วย วันที่ เดือน และ ปี เช่น วันจันทร์ที่ 12 สิงหาคม พ.ศ. 2546 หรือ Monday 12 August 2003 ให้เขียนโปรแกรมทดสอบ

8. จงออกแบบ class Matrix สำหรับการประมวลผลต่าง ๆ เช่น การบวก การลบ การคูณ และกระบวนการอื่น ๆ ที่เกี่ยวข้อง ให้เขียนโปรแกรมทดสอบ

9. จงออกแบบ class ที่ใช้สำหรับการคำนวณหาอายุของ user ที่ใส่เข้ามาจาก keyboard ในรูปแบบของ dd/mm/yyy โดยให้ผลลัพธ์ที่หาได้อยู่ในรูปแบบของ จำนวนของวัน ทั้งหมด ให้เขียนโปรแกรมทดสอบ

10. จงออกแบบ class MyString ที่มี method สำหรับการประมวลที่เกี่ยวข้องกับ string ต่าง ๆ เช่น การบวก string สองตัว การค้นหาคำใน string การลบคำที่อยู่ใน string การนับจำนวนตัวอักษรที่อยู่ใน string ให้เขียนโปรแกรมทดสอบ

6

การสร้าง class ใหม่จาก class เดิม และการถ่ายทอดคุณสมบัติ

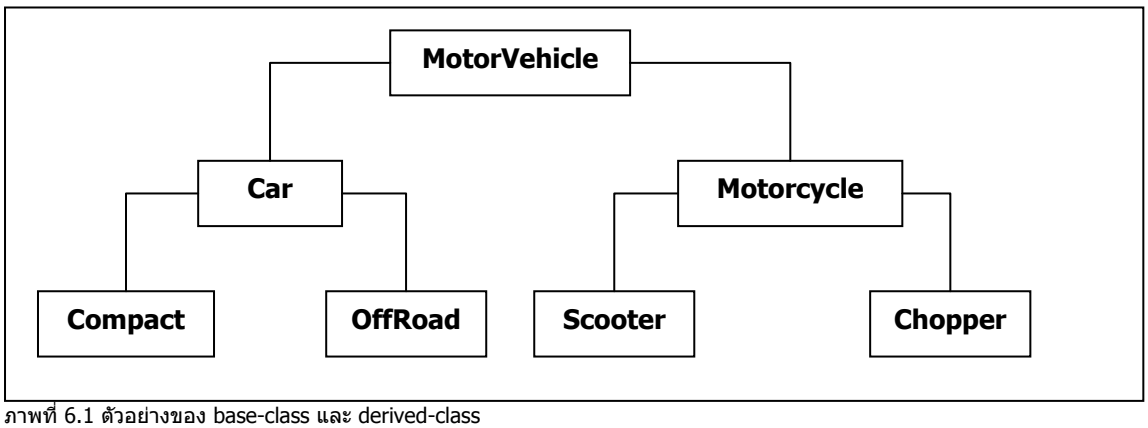
เราได้พูดถึงการสร้าง class และส่วนประกอบต่าง ๆ ที่สำคัญของ class รวมไปถึงการสร้าง method ในบทที่ห้า สำหรับบทที่หกนี้เราจะสำรวจในเรื่องของการสร้าง class ใหม่จาก class เดิมที่มีอยู่ การนำเอาส่วนประกอบของ class เดิมมาใช้ใน class ใหม่ การถ่ายทอดคุณสมบัติและ กระบวนการต่าง ๆ จาก class เดิมสู่ class ใหม่

หลังจากจบบทนี้แล้ว ผู้อ่านจะได้ทราบถึงเรื่องของ

- การใช้ class เดิมสำหรับการสร้าง class ใหม่ (Extended class)
- การถ่ายทอดคุณสมบัติ (Inheritance)
- คุณสมบัติ และการใช้ประโยชน์จาก polymorphism
- การสร้าง abstract class และ abstract method
- การใช้ และ การสร้าง interface

การสร้าง class ใหม่จาก class เดิมที่มีอยู่แล้วเป็นการพัฒนาโปรแกรมที่ทำให้การเขียน code ใช้เวลาน้อย และเป็นการใช้ code อย่างคุ้มค่าที่สุด Java ยอมให้มีการสร้าง class เดิมจาก class ใหม่ได้อย่างง่าย ๆ โดยที่ผู้เขียนโปรแกรมแทบจะไม่ต้องทำอะไรมากมาย (ยกเว้น code ใหม่ที่เขียนขึ้น) ก่อนที่จะพูดถึงศัพท์ต่าง ๆ ที่เกี่ยวข้องกับการสร้าง class ใหม่จาก class เดิมเราจะมาดูกันถึงตัวอย่างของการสร้าง class ใหม่จาก class เดิมก่อน

เราจะเริ่มต้นด้วยการสร้าง class ง่าย ๆ class หนึ่งที่เกี่ยวกับรถที่ใช้เครื่องยนต์ (motor vehicle) โดยกำหนดให้มีข้อมูลที่ไม่ซับซ้อน เช่น ยี่ห้อ (make) รุ่น (model) ปีที่ผลิต (year) และ จำนวนที่นั่ง (seats) พร้อมทั้งกำหนดให้มี method สำหรับการเข้าหาข้อมูลเหล่านั้น หลังจากที่ได้ class MotorVehicle แล้วเราก็ออกแบบ class อื่น ๆ ที่ได้รับคุณสมบัติจาก class MotorVehicle ดังที่แสดงให้เห็นคร่าว ๆ จาก diagram นี้ (เราจะดูโครงสร้างของ class ทั้งหมดต่อไป)



ภาพที่ 6.1 ตัวอย่างของ base-class และ derived-class

เราออกแบบให้ class MotorVehicle เป็น class แม่แบบ หรือที่เรียกว่า parent class (หรือ base class หรือ super class) ซึ่งจะเป็น class ที่อยู่ด้านบนสุดของ class ต่าง ๆ ในกลุ่มนี้ เรากำหนดให้ class MotorVehicle มี class ลูก (children) อยู่สอง class คือ class Car และ class Motorcycle class ลูกทั้งสองต่างก็มี class ที่เป็นลูกของมันเองอีกสอง class โดยที่ class Car มีลูกเป็น class Compact และ class OffRoad ส่วน class Motorcycle มี class Scooter และ class Chopper เป็น class ลูก เราจะกำหนดให้ class ลูกได้รับคุณสมบัติจาก class แม่ทั้งหมด คือ ข้อมูลทุกตัวที่มีอยู่ใน class MotorVehicle เรามาดูกันถึง ข้อมูล และ method ที่เรากำหนดให้มีใน class MotorVehicle

```
//MotorVehicle.java – Inheritance

class MotorVehicle {
    protected String make;    //e.g. Ford, Honda
    protected String model;   //e.g. Taurus, Steed
    protected int year;       //e.g. 2001, 2003.
    protected int seats;      //e.g. 4, 2

    //constructor
    MotorVehicle(String make, String model, int year, int seats) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.seats = seats;
    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }

    public int getSeats() {
        return seats;
    }
}
```

เรากำหนดให้ตัวแปรที่มีอยู่ใน class MotorVehicle ใช้นโยบายของการเข้าหาแบบ protected ซึ่งเป็นการกำหนดให้ class ลูกที่เกิดขึ้นจาก class MotorVehicle สามารถที่จะเรียกใช้ตัวแปรเหล่านี้ได้ class อื่น ๆ ที่ไม่ได้มาจาก class นี้จะมองไม่เห็นข้อมูลเหล่านี้ เรากำหนดให้มี constructor เพียงตัวเดียวสำหรับการสร้าง object โดยเราเลือกใช้คำสั่ง this ในการแยกแยะตัวแปรที่อยู่ใน class กับตัวแปรที่เป็น parameter

```
MotorVehicle(String make, String model, int year, int seats) {
    this.make = make;
    this.model = model;
    this.year = year;
    this.seats = seats;
}
```

คำว่า this เป็นคำที่ Java สงวนไว้สำหรับการบ่งบอกถึง object ที่ได้รับการกระทำด้วยกระบวนการต่าง ๆ ณ เวลานั้น ๆ เช่นในตัวอย่างประโยค

```
this.make = make;
```

เมื่อ Java ประมวลผล constructor ตัวนี้ Java จะทำการกำหนดค่าจากตัวแปร make ที่เป็น parameter ของ constructor ให้กับตัวแปร make ที่อยู่ใน class ซึ่งเป็น copy ที่ object ตัวนี้ใช้อยู่

เราไม่จำเป็นที่จะต้องใช้ this เป็นตัวบ่งบอกตัวแปรเสมอไป ถ้าเราใช้ตัวแปรที่อยู่ใน parameter list ที่มีชื่อไม่เหมือนกับ ตัวแปรของ class (เช่นตัวอย่างอื่น ๆ ที่เราได้แสดงไว้ในบทที่ห้า) ส่วน method อื่น ๆ ที่มีอยู่ก็เป็นเพียง method ที่ส่งค่าของตัวแปรเหล่านี้ให้กับผู้เรียก สำหรับ class อื่น ๆ ที่เราได้สร้างขึ้นมีดังนี้

```
//Car.java - Derived from MotorVehicle

class Car extends MotorVehicle {
    protected int doors;

    Car(String make, String model, int year, int seats, int doors) {
        super(make, model, year, seats);
        this.doors = doors;
    }

    public int getDoors() {
        return doors;
    }

    //display Car's information as string
    //using base class methods
    public String toString() {
        StringBuffer buffer = new StringBuffer();
        buffer.append(super.getMake() + ", " +
                      super.getModel() + ", " +
                      super.getYear() + ", " + " seating: " +
                      super.getSeats() + ", " + doors + "doors.");
        return new String(buffer);
    }
}
```

เราสร้าง class Car จาก class MotorVehicle ด้วยการใช้คำสั่ง class Car extends MotorVehicle ซึ่งเป็นคำสั่งที่ Java กำหนดให้ใช้ถ้าต้องการสร้าง class ใหม่ที่ต้องการใช้คุณสมบัติของ class อื่น

ภายใน class Car ที่เราได้สร้างขึ้นเรากำหนดให้มีตัวแปรหนึ่งตัวที่เป็นตัวกำหนดจำนวนของประตูที่รถคันนี้มีอยู่ และเพื่อให้การแสดงผลไปยังหน้าจอทำได้ง่ายขึ้น เราก็สร้าง method toString() สำหรับการแสดงผลของ object ที่เกิดจาก class Car ของเรา เราเรียกข้อมูลที่เราได้รับการถ่ายทอดจาก class MotorVehicle ด้วยการใช้คำสั่ง ดังนี้

```
super.getMake();
```

เมื่อ Java เห็นการใช้คำสั่งแบบนี้ Java ก็จะไปค้นหา method getMake() ที่อยู่ใน class ที่เป็น class แม่ หรือที่เรียกกันว่า super class พร้อมทั้งประมวลผลคำสั่งต่าง ๆ ที่มีอยู่ใน method getMake()

ผู้อ่านจะเห็นว่าเราไม่ต้องประกาศตัวแปรขึ้นมาใช้ใหม่ใน class Car ของเราเลย เราสามารถเรียกใช้ตัวแปรเหล่านี้จาก class MotorVehicle ได้เลย เรามาทดสอบ class ทั้งสองด้วยการสร้าง object จาก class Car สองตัว เช่นที่เห็นในโปรแกรม TestVehicle.java นี้

```
//TestVehicle.java

class TestVehicle {
    public static void main(String[] args) {
        //create two cars, ford and honda
```

```
        Car ford = new Car("Ford", "Taurus", 2001, 5, 5);
        Car honda = new Car("Honda", "City", 2003, 4, 4);

        //display informatin about cars
        System.out.println(ford);
        System.out.println(honda);
    }
}
```

ผลลัพธ์ที่เราได้จากการ run คือ

```
Ford, Taurus, 2001, seating: 5, 5 doors.
Honda, City, 2003, seating: 4, 4 doors.
```

เราได้ใช้นโยบายการเข้าหาแบบที่ใช้คำว่า protected ซึ่งเราไม่ได้อธิบายไว้เลยก่อนหน้านี้ สาเหตุก็เนื่องจากว่าคำว่า protected โดยส่วนใหญ่จะใช้กันในเรื่องของการถ่ายทอดคุณสมบัติจาก class แม่ไปยัง class ลูก โดยกำหนดไว้ว่า class ลูกสามารถที่จะใช้ตัวแปร (หรือ method ที่ใช้คำว่า protected นำหน้า) จาก class แม่ได้อย่างไม่มีเงื่อนไข แต่ถ้าเราใช้คำว่า private แทน class ลูกไม่สามารถที่จะเข้าหาตัวแปร หรือ method เหล่านี้ได้ ดังตัวอย่างต่อไปนี้

```
Car(String make, String model, int year, int seats, int doors) {
    super(make, model, year, seats);
    this.doors = doors;
    System.out.println("base class: " + super.make);
}
```

ใน class MotorVehicle เราเปลี่ยนนโยบายการเข้าหาตัวแปร make ให้เป็น private

```
private String make;
```

และเพิ่มประโยค

```
System.out.println("base class: " + super.make);
```

ใน constructor ของ class Car ซึ่งเป็นประโยคที่พยายามเข้าหาตัวแปรใน class MotorVehicle ด้วยการใช้ super.make Java จะไม่ยอมให้เรา compile พร้อมกับฟ้องว่าเราไม่สามารถเข้าหาตัวแปรตัวนี้ได้ วิธีเดียวที่เราจะเข้าหาตัวแปรเหล่านี้ได้ ก็คือการเข้าผ่านทาง method getMake() ที่เราได้กำหนดนโยบายการเข้าหาให้เป็น public เท่านั้น

โดยทั่วไปเรามักจะกำหนดให้ตัวแปรที่ใช้ร่วมกันใน class ลูก (ที่มีอยู่ใน class แม่) ทั้งหลายมีนโยบายการเข้าหาที่เป็นแบบ private เพื่อป้องกันการใช้ที่ไม่พึงประสงค์จาก class อื่น และเราก็ควรจะกำหนดให้ method ที่อยู่ใน base class มีนโยบายการเข้าหาที่เป็นแบบ protected ด้วยเหตุผลเช่นเดียวกัน และเหตุผลอีกอันหนึ่งก็คือ คำว่า protected เมื่อมีการนำมาใช้แล้ว class อื่น ๆ ที่อยู่ใน package เดียวกันก็สามารถที่จะเข้าหาตัวแปร หรือ method เหล่านี้ได้ ดังนั้นถ้าต้องการให้ class ต่าง ๆ ของเราที่สร้างขึ้นในตัวอย่างเป็น class ที่การใช้นโยบายการเข้าอย่างถูกต้อง เราก็ต้องเปลี่ยน class ของเราให้เป็น ดังนี้

```
class MotorVehicle {
    private String make;  //e.g. Ford, Honda
    private String model; //e.g. Taurus, Steed
    private int year;     //e.g. 2001, 2003
    private int seats;    //e.g. 4, 2

    //constructor
    MotorVehicle(String make, String model, int year, int seats) {
        this.make = make;
        this.model = model;
        this.year = year;
```

```
        this.seats = seats;
    }

    protected String getMake() {
        return make;
    }

    protected String getModel() {
        return model;
    }

    protected int getYear() {
        return year;
    }

    protected int getSeats() {
        return seats;
    }
}
```

```
class Car extends MotorVehicle {
    private int doors;

    Car(String make, String model, int year, int seats, int doors) {
        super(make, model, year, seats);
        this.doors = doors;
    }

    protected int getDoors() {
        return doors;
    }

    public String toString() {
        StringBuffer buffer = new StringBuffer();
        buffer.append(super.getMake() + ", " +
                      super.getModel() + ", " +
                      super.getYear() + ", " + "seating: " +
                      super.getSeats() + ", " + doors + " doors.");
        return new String(buffer);
    }
}
```

เราไม่สามารถที่จะเปลี่ยนนโยบายการเข้าหาของ method toString() ให้เป็น protected ได้ก็เพราะว่า Java ได้กำหนดให้การเข้าหาของ method toString() เป็นแบบ public ไว้แล้ว ซึ่งเราจะได้อธิบายต่อไปในเรื่องของการ override

เพื่อให้เห็นถึงความพิเศษของ การถ่ายทอดคุณสมบัติ และเพื่อเป็นการแนะนำเรื่องของการ override เราจะสร้าง class Compact ขึ้นมาอีก class หนึ่ง โดยกำหนดให้ class Compact นี้เป็น class ที่ได้รับการถ่ายทอดคุณสมบัติจาก class Car ดังนี้

```
//Compact.java Derived from Car

class Compact extends Car {

    //constructor -using base class (Car) constructor
    Compact(String make, String model, int year,
            int seats, int doors) {
        super(make, model, year, seats, doors);
```

```
    }

    //override Car's toString() method
    //presenting Comapct car's details as string
    public String toString() {
        StringBuffer buffer = new StringBuffer();
        buffer.append("Compact car: ");
        buffer.append(super.toString());

        return new String(buffer);
    }
}
```

class Compact ของเราเรียกใช้ข้อมูลทุกอย่างที่ class แม่มีให้ ตัวแรกที่เราเรียกใช้ก็คือ constructor ของ class Car (ซึ่ง constructor ของ class Car ก็เรียกใช้ constructor ของ class MotorVehicle) ตัวที่สองที่เราเรียกใช้ก็คือ method toString() ของ class Car ซึ่งเราได้เพิ่มคำว่า "Compact car: " ไว้ก่อนหน้าข้อมูลอื่น ๆ ด้วยการใช้ method append() ของ class StringBuffer ดังนี้

```
buffer.append("Compact car: ");
buffer.append(super.toString());
```

เมื่อทดลอง run ด้วยการใช้คำสั่ง

```
Compact honda = new Compact("Honda", "City", 2003, 4, 4);
System.out.println(honda);
```

ผลลัพธ์ที่ได้ คือ

```
Compact car: Honda, City, 2003, seating: 4, 4 doors.
```

ซึ่งต่างจากการ run ก่อนหน้านี้ (ตอนที่เราสร้าง object จาก class Car)

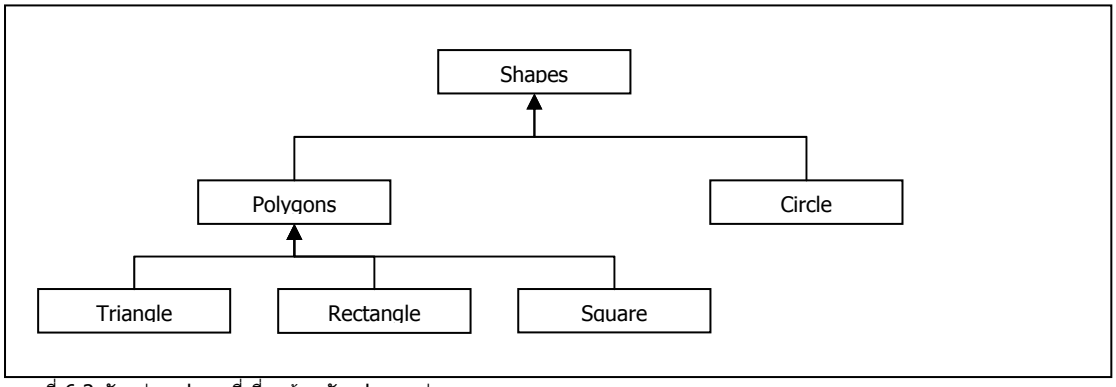
เราใช้เทคนิคที่เรียกว่า method overriding ในการแสดงผลข้อมูลของ object ที่เกิดจาก class Compact ในการ override (ซึ่งต่างกับ overload ที่เราได้ทำก่อนหน้านี้ – overload ทำกับ method ที่อยู่ใน class เดียวกันและต้องเป็น method ที่มี parameter list ไม่เหมือนกัน) method จาก class แม่ หรือที่เรียกว่า base class นั้นเราสามารถที่จะเขียน code ให้ทำงานในลักษณะใดก็ได้ โดยทั่วไปการ override method ทำให้การเรียกใช้งานทำได้ง่ายขึ้น ไม่สับสน เพราะ method ชื่อเดียวกันมีอยู่ใน class ทุกตัว แต่ทำหน้าที่ตามที่ได้กำหนดไว้ใน method ของ class นั้น ๆ

เราสามารถที่จะ override method ที่มีอยู่ใน class แม่ได้ทุกตัว และ object ที่เกิดจาก class ลูก หรือที่เรียกว่า derived class นั้นเมื่อมีการเรียกใช้ method ที่ได้รับการ override Java จะไปเรียก method ที่อยู่ใน class ลูก ไม่ใช่ method ที่อยู่ใน class แม่

จากตัวอย่างของ class Compact ที่ได้กล่าวไว้ เราได้ใช้ทั้ง การถ่ายทอดคุณสมบัติ และการ override method โดยเราได้รับการถ่ายทอดในเรื่องของตัวแปร และได้ใช้เทคนิคการ override ในการใช้ method toString()

Method ที่ได้รับการ override ใน class ลูกนั้นไม่สามารถที่จะเปลี่ยนแปลงนโยบายของการเข้าหาให้เป็นอย่างอื่นที่มีความเข้มงวดกว่าได้ เช่น ถ้า method ใน class แม่มีนโยบายการเข้าหาที่เป็น public ใน class ลูกก็ไม่สามารถที่จะเปลี่ยนให้เป็น private ได้เพราะการเข้าหาแบบ private นั้นเข้มงวดมากกว่า แต่ถ้าใน class แม่นโยบายเป็น private เราสามารถที่จะเปลี่ยนให้เป็น public ได้ พูดง่าย ๆ ก็คือว่า เราเพิ่มความเข้มงวดนโยบายการเข้าหาใน class ลูกไม่ได้ แต่ลดลงได้

เรามาลองดูตัวอย่างของการถ่ายทอดคุณสมบัติกันอีกสักตัวอย่างหนึ่ง สมมติว่าเรามี class ต่าง ๆ ดังที่แสดงใน diagram นี้



ภาพที่ 6.2 ตัวอย่าง class ที่เกี่ยวข้องกับ shape ต่าง ๆ

โดยกำหนดให้การถ่ายทอดคุณสมบัติเป็นไปดังที่เห็นใน diagram คือ Shapes เป็น base class Polygons และ Circle เป็น class ลูกของ Shapes ส่วน Triangle Rectangle และ Square เกิดมาจาก class Polygons

ในทุก ๆ class เรากำหนดให้มี method what() เพียง method เดียว ดังที่แสดงไว้ใน code ที่เห็นนี้

```
//Shapes.java - Base class for all shapes

class Shapes {
    //self info.
    protected String what() {
        return "I am a shape";
}
```

```
//Polygons.java – Polygon

class Polygons extends Shapes {
    //self info.
    protected String what() {
        return "I am a polygon.";
    }
}
```

```
//Circle.java - sub-class of Shapes

class Circle extends Shapes {
    //self info
    protected String what() {
        return "I am a circle.";
    }
}
```

```
//Triangle.java - sub-class of Polygons

class Triangle extends Polygons {
    //self info.
    protected String what() {
        return "I am a triangle.";
    }
}
```

```
//Square.java - sub-class of Polygons
```

```
class Square extends Polygons {
    //self info.
    protected String what() {
        return "I am a square.";
    }
}
```

```
//Rectangle.java - sub-class of Polygons

class Rectangle extends Polygons {
    //self info.
    protected String what() {
        return "I am a rectangle.";
    }
}
```

เราเขียนกำหนดให้โปรแกรมตรวจสอบเป็นดังนี้

```
//TestShapes.java - driver program

class TestShapes {
    public static void main(String[] args) {
        //creating different shape objects from Shapes
        Shapes shape = new Shapes();
        Shapes poly = new Polygons();
        Shapes sq = new Square();
        Shapes cir = new Circle();

        //display info. about particular shapes
        System.out.println(shape.what());
        System.out.println(sq.what());
        System.out.println(cir.what());
        System.out.println(poly.what());
    }
}
```

ในโปรแกรมตรวจสอบเราได้สร้าง object สี่ตัวจาก class Shapes แต่ทำการกำหนดการสร้างผ่านทาง constructor ของ class นั้น ๆ ที่เราต้องการสร้าง เช่น

```
Shapes sq = new Square();
```

ถ้ามองดูอย่างผิวเผินจะเห็นว่าการประกาศแบบนี้ไม่น่าที่จะทำได้ เนื่องจากว่าเรากำหนดให้ sq เป็น object ที่มีสถานะเป็น Shapes แต่กลับไปเรียก constructor ของ Square แต่ Java ยอมให้การสร้าง object แบบนี้เกิดขึ้นได้ก็เพราะว่า Square เป็น class ที่เกิดมาจาก class Shapes (พูดง่าย ๆ ก็คือ Circle เป็น Shapes แบบหนึ่ง) และภายใน class Shapes เรากำหนดให้มี method what() อยู่ ดังนั้น class ลูกที่เกิดมาก็สามารถที่จะเข้าหา method นี้ได้แต่ ภายในตัว class ลูกทั้งหมดก็มี method what() อยู่ดังนั้น การเรียก method what() ขึ้นมาใช้จึงเป็นการเรียก method ที่มีอยู่ภายใน class ลูก (overriding) ดังที่แสดงให้เห็นจากผลลัพธ์นี้

```
I am a shape
I am a square.
I am a circle.
I am a polygon.
```

Polymorphism (ความมีรูปแบบที่หลากหลาย)

การทำงานในลักษณะที่ได้กล่าวมานั้นเราเรียกว่า late binding หรือที่เรียกกันมากมายในหนังสือหลาย ๆ เล่มว่า polymorphism ซึ่งหมายถึงความเปลี่ยนแปลงรูปแบบตามสภาพของ object ที่ได้ถูกสร้างขึ้น หรือที่เป็นอยู่ คำว่า late binding หมายถึงเวลาที่ compiler จัดสรรหน่วยความจำให้กับตัวแปรในขณะที่โปรแกรมกำลังถูก execute (run อยู่) ไม่ใช่ในขณะที่กำลัง compile โปรแกรม

Polymorphism ทำได้เฉพาะ method เท่านั้นเราไม่สามารถจะใช้วิธีการแบบนี้กับตัวแปรที่อยู่ใน class ได้ การเข้าหาตัวแปรจะต้องทำผ่านทาง object ที่เกิดจาก class นั้น ๆ เท่านั้น

สมมติว่าเราต้องการที่จะเขียน method ที่คำนวณหาเส้นรอบวงของ object ต่าง ๆ ที่สร้างขึ้นมาจาก class Shapes เราก็อาจทำได้ด้วยการเขียน method ลงไปใน class นั้น ๆ (ที่ถูกสร้างมาจาก class แม่) แต่การกระทำดังกล่าวคงไม่ค่อยดีเท่าไรนัก ถ้าเราไม่รู้ว่า object ที่เราต้องการสร้างขึ้นมานั้นรูปร่างหน้าตาเป็นอย่างไร และคงจะไม่ค่อยเข้าท่าเท่าไรนักที่จะหาเส้นรอบวงของ shape ที่เรายังไม่รู้ว่าเป็น shape อะไร แต่ Java เอื้อโอกาสให้เราสามารถที่จะประกาศ (declare) method ใน class แม่แต่ไปกำหนดหน้าที่ (define) ใน class ลูก วิธีการดังกล่าวเราเรียกว่า การสร้าง abstract class

Abstract class เป็น class ที่มีการประกาศ method ใน class แม่ แต่ไม่มีการกำหนดหน้าที่ใด ๆ เรื่องของการกำหนดหน้าที่จะเป็นภาระของ class ลูก ดังเช่น ตัวอย่างต่อไปนี้

```
//Shapes.java - Base class for all shapes

public abstract class Shapes {
    private String iAm;

    Shapes(String whatIam) {
        iAm = new String(whatIam);
    }

    //self info.
    public String toString() {
        StringBuffer buffer = new StringBuffer();
        buffer.append("I am a " + iAm);
        return new String(buffer);
    }

    //dummy method - implementation is done in derived classes
    //to calculate a perimeter of a shape
    public abstract double perimeter();
}
```

เราเปลี่ยนแปลง class Shapes ของเราให้เป็น abstract class พร้อมทั้งประกาศ method perimeter() ที่ไม่มี code ใด ๆ อยู่เลย หลังจากนั้นเราก็สร้าง class อื่น ๆ ที่ได้รับการถ่ายทอดคุณสมบัติจาก class Shapes ดังนี้

```
//Polygons.java – Polygon

public abstract class Polygons extends Shapes {

    //constructor
    Polygons(String type) {
        super(type);
    }

    //self info.
    public String toString() {
        return super.toString();
```

```
    }

    //dummy method to be implemented by derived classes
    public abstract double perimeter();
}
```

```
//Triangle.java - sub-class of Polygons

public class Triangle extends Polygons {
    private double side1, side2, side3;

    //constructor
    Triangle(String type, double side1, double side2, double side3) {
        super(type);      //calling base-class constructor
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    //self info.
    public String toString() {
        return super.toString();
    }

    //calculating a perimeter of this triangle
    public double perimeter() {
        return side1 + side2+ side3;
    }
}
```

```
//Circle.java - sub-class of Shapes

public class Circle extends Shapes {
    private double radius;

    //constructor
    Circle(String type, double radius) {
        super(type);
        this.radius = radius;
    }

    //self info
    public String toString() {
        return super.toString();
    }

    //calculating a perimeter of this circle
    public double perimeter() {
        return 2.0 * Math.PI * radius;
    }
}
```

```
//Square.java - sub-class of Polygons

public class Square extends Polygons {
    double side;

    //constructor
```

```
    Square(String type, double side) {
        super(type);
        this.side = side;
    }

    //self info.
    public String toString() {
        return super.toString();
    }

    //calculating a perimeter of this square
    public double perimeter() {
        return side * 4;
    }
}
```

หลังจากที่เราได้สร้าง class Polygons class Circle class Triangle และ class Square พร้อมทั้งกำหนดหน้าที่ของ method perimeter() ใน class Circle class Triangle และ class Square เรียบร้อยแล้วเราก็เขียนโปรแกรมเพื่อตรวจสอบว่า method ของเราใช้งานได้ถูกต้องหรือไม่ ดังนี้

```
//TestShapes.java - driver program

class TestShapes {
    public static void main(String[] args) {
        Shapes cir, tri;
        Polygons sq;

        //instantiate shapes
        sq = new Square("square", 4.0);
        cir = new Circle("circle", 2.0);
        tri = new Triangle("triangle", 3.0, 4.0, 5.0);

        //display info. about particular shapes
        System.out.println(sq + " with a perimeter of: " +
                           sq.perimeter());
        System.out.println(cir + " with a perimeter of: " +
                           cir.perimeter());
        System.out.println(tri + " with a perimeter of: " +
                           tri.perimeter());
    }
}
```

ในโปรแกรม TestShapes.java เราประกาศตัวแปร cir และ tri จาก class Shapes ตัวแปร sq จาก class Polygons สาเหตุที่เราประกาศตัวแปรแทนการสร้าง object ก็เพราะว่า Java ไม่ยอมให้เราทำ เนื่องจากว่า class Shapes และ class Polygons เป็น class ที่ถูกกำหนดให้เป็น abstract class แต่เราสามารถที่จะใช้ตัวแปรทั้งสามในการสร้าง object (instantiation) จาก class Circle class Square และ class Triangle ได้

ประโยคในการสร้าง object ทั้งสาม

```
sq = new Square("square", 4.0);
cir = new Circle("circle", 2.0);
tri = new Triangle("triangle", 3.0, 4.0, 5.0);
```

อาจดูแปลกไปจากการสร้าง object ที่เราได้เคยทำมา แต่เราสามารถทำได้ก็เพราะว่า class Shapes เป็น base class ของ class ทั้งหมด ส่วน class Polygons ก็เป็น base class ของ class Square และ class

Triangle การสร้าง object ทั้งสามจึงสามารถทำได้ พูดง่าย ๆ ก็คือว่า ทั้ง sq cir และ tri ต่างก็เป็น shape ทั้งนั้น (e.g. a square is a shape, a circle is a shape, a triangle is a shape etc.)

ผลลัพธ์ที่เราได้จากการ run คือ

```
I am a square with a perimeter of: 16.0
I am a circle with a perimeter of: 12.566370614359172
I am a triangle with a perimeter of: 12.0
```

Polymorphism มีประโยชน์มากในการกำหนดหน้าที่ของ method ที่ object ต่าง ๆ ใช้ร่วมกัน ซึ่งหน้าที่ต่าง ๆ เหล่านี้จะขึ้นอยู่กับคุณลักษณะของ object ที่เกิดจาก class นั้น ๆ ดังที่เราได้แสดงให้ดูในการหาเส้นรอบวงของรูปทรงต่าง ๆ หากเราต้องการเราก็สามารถที่จะสร้าง array ในการเก็บ shape ต่าง ๆ เหล่านี้ได้ ดังตัวอย่างที่แสดงให้เห็นนี้

```
//TestShapes.java - driver program

class TestShapes {
    public static void main(String[] args) {
        Shapes []shapes = new Shapes[5];

        //instantiate 5 shapes
        for(int i = 0; i < 5; i++)
            shapes[i] = randomShape();

        //display info. about particular shapes
        for(int i = 0; i < 5; i++) {
            System.out.print(shapes[i] + " with a perimeter of: ");
            System.out.println(shapes[i].perimeter());
        }
    }

    public static Shapes randomShape() {
        switch((int)(Math.random() * 3)) {
            default:
            case 0: return new Circle("circle", Math.random() * 12.0);
            case 1: return new Square("square", Math.random() * 10.5);
            case 2: return new Triangle("triangle", 2.5, 4.0, 5.5);
        }
    }
}
```

เราเลือกที่จะให้การสร้างรูปทรงต่าง ๆ เกิดแบบ random ดังนั้นเราจึงเขียน method randomShape() ขึ้นมาใช้สำหรับการสร้างรูปทรงที่ไม่มีข้อกำหนดที่ตายตัว รูปทรงที่เกิดขึ้นจะถูกกำหนดโดย method random() ของ class Math

ผลลัพธ์ที่ได้จากการ run คือ

```
I am a square with a perimeter of: 15.282560130191253
I am a triangle with a perimeter of: 12.0
I am a circle with a perimeter of: 0.209235603993188
I am a triangle with a perimeter of: 12.0
I am a circle with a perimeter of: 7.8292546239806695
```

หากเราอยากรู้ว่า object ที่สร้างขึ้นมานั้นเกิดจาก class อะไรเราก็สามารถหาได้จากการใช้ method getClass() และ method getName() ที่มีอยู่ใน class Class ของ Java เช่นถ้าเราต้องการรู้ว่า object ที่ shapes[0] เก็บไว้เกิดมาจาก class อะไรเราก็ใช้คำสั่งดังนี้

```
Class object = shapes[0].getClass();
System.out.println(object.getName());
```

ถ้าเราอยากรู้ว่า object ตัวนี้มี class แม่หรือไม่เราก็ใช้ method getSuperclass() เช่น

```
Class shape  = object.getSuperclass();
System.out.println(shape.getName());
```

ยังมี method ใน class Class อีกหลายตัวที่เราสามารถนำมาใช้ได้ แต่โดยส่วนใหญ่แล้ว เราแทบที่จะไม่ต้องใช้ method เหล่านี้ในการพัฒนาโปรแกรมทั่วไปเลย เพราะ method เหล่านี้จะให้ข้อมูลที่เกี่ยวกับ class ที่ผู้ใช้ (หรือ ของ Java) สร้างขึ้นมาเท่านั้นเอง

การส่งและรับ object

ตัวอย่างต่อไปนี้เป็นการส่ง object เข้าไปใน method พร้อมกับการส่ง object กลับด้วยการเปลี่ยนแปลงบางอย่าง สมมติว่าเราต้องการสร้าง วงกลมจาก สี่เหลี่ยมด้านเท่าที่มีอยู่แล้ว เราก็อาจเขียน method ที่ทำหน้าที่นี้ให้เรา ดังนี้

```
//SquareToCircle.java - Passing and returning object to/from method

class SquareToCircle {
    public static void main(String []args) {
        Shapes square; //a square object
        Shapes circle; //a circle object

        //creating a square with side = 3
        square = new Square("square", 3.0);
        System.out.println(square + " with an area of " +
                           square.area());

        //creating a circle from a square
        circle = SqToCir(square);
        System.out.println(circle + " with an area of " +
                           circle.area());
    }

    //method to convert a square to a circle
    public static Shapes SqToCir(Shapes square) {
        //finding a radius from an area of a square
        double SqArea = square.area();
        double radius = Math.sqrt(SqArea / Math.PI);

        //instantiating a circle
        Circle circle = new Circle("circle", radius);

        return circle;
    }
}
```

เพื่อให้การทำงานของเราประสพผลสำเร็จ เราได้เพิ่ม abstract method area() ใน class Shapes และ class Polygons และทำการกำหนดหน้าที่ของ method area() ใน class Square และ class Circle ซึ่งเป็นการคำนวณหาพื้นที่ของรูปทรงที่ได้ถูกสร้างขึ้น

ใน class Shapes เรากำหนด ดังนี้ (ใน class Polygons ก็ทำแบบเดียวกัน)

```
public abstract class Shapes {
    …
```

```
    …
    //code อื่น ๆ เหมือนเดิม
    …
    …

    public abstract double perimeter();
    public abstract double area();
}
```

ใน class Circle เรากำหนด code ของ method area() ดังนี้

```
//calculate area of this circle
public double area() {
    return Math.PI * radius * radius;
}
```

และใน class Square เรากำหนด code ของ method area() ดังนี้

```
//calculate area of this square
public double area() {
    return side * side;
}
```

ในตัวโปรแกรมของเรา เราสร้าง object สองตัวคือ square และ circle โดยกำหนดให้ square มีความยาวของด้านแต่ละด้านเท่ากับ 3.0 หลังจากที่แสดงข้อมูลของ square เรียบร้อยแล้วเราก็ส่ง square ไปให้ method SqToCir() ซึ่งเป็น method ที่ทำการเปลี่ยน square ให้เป็น circle โดยกำหนดให้ทั้งสอง object มีพื้นที่เท่ากัน

```
public static Shapes SqToCir(Shapes square) {
    //finding a radius from an area of a square
    double SqArea = square.area();
    double radius = Math.sqrt(SqArea / Math.PI);

    //instantiating a circle
    Circle circle = new Circle("circle", radius);

    return circle;
}
```

เราเริ่มต้นด้วยการหาพื้นที่ของ square ที่ส่งเข้ามา หลังจากนั้นเราก็หารัศมีของวงกลมที่เราต้องการสร้างจากพื้นที่ ที่หาได้ เมื่อได้แล้วเราก็สร้าง circle ด้วยรัศมีนี้ เสร็จแล้วจึงส่งวงกลมที่ได้นี้กลับไป

ผลลัพธ์ที่เราได้จากการ run โปรแกรมของเราคือ

```
I am a square with an area of 9.0
I am a circle with an area of 9.0
```

ข้อดีของการกำหนดให้ base class เป็น abstract class นั้น ก็คือ เราไม่ต้องกำหนดหน้าที่ของ method ใน class แม่ เพราะว่าเราอาจไม่รู้ข้อมูลทั้งหมดที่ method ต้องการใช้ ดังเช่น method perimeter() และ method area() เนื่องจากว่าเรายังไม่รู้ว่า shape ที่มีอยู่เป็น object ชนิดอะไร ข้อมูลเป็นอย่างไร แต่เรารู้ข้อมูลทั้งหมดที่ Circle และ Square ต้องการใช้ในการคำนวณหาเส้นรอบวง และพื้นที่จาก การสร้าง object ของ class ทั้งสอง

การสร้าง Interface

จากเทคนิคการใช้ polymorphism ในตัวอย่างก่อนหน้านี้ เราต้องกำหนดให้มี method ใน class แม่ และไปกำหนดหน้าที่ของ method เหล่านี้ใน class ลูก ซึ่งเป็นการสร้าง class ในลักษณะที่ซ้ำซ้อนพอสมควร ถ้าเราต้องคุณสมบัติของ polymorphism ใน class ที่เกิดจาก class แม่นี้ เราก็ไม่จำเป็นที่จะต้องสร้าง class แม่ขึ้นมา แต่ไปสร้าง interface ขึ้นมาแทน

Interface เป็นที่รวบรวมค่าคงที่ และ abstract method ต่าง ๆ ที่จะต้องมีการเรียกใช้ใน class อื่น ๆ ที่เรียก interface นี้ไปใช้ การสร้าง interface ก็ไม่ยาก แทนที่เราจะใช้คำว่า class เราก็ใช้คำว่า interface แทน เราจะใช้ class ต่าง ๆ ที่เราได้สร้างไว้ก่อนหน้านี้ในเรื่องของรูปทรงต่าง ๆ เป็นตัวอย่าง และเราจะต้องทำการเปลี่ยนแปลง class ต่าง ๆ ดังนี้

ใน class Shapes และ class Polygons เราจะตัดเอา abstract method perimeter() และ area() ออก พร้อมกับเปลี่ยนการประกาศ class Shapes ให้เป็นดังนี้

```
public abstract class Shapes implements Calculate {
    private String iAm;

    Shapes(String whatIam) {
        iAm = new String(whatIam);
    }

    //self info.
    public String toString() {
        StringBuffer buffer = new StringBuffer();
        buffer.append("I am a " + iAm);
        return new String(buffer);
    }
}
```

เราเพิ่มคำว่า implements Calculate เข้าสู่ด้านหลังสุดของ class Shapes โดยมีเงื่อนไขว่าเราต้องมี interface ชื่อ Calculate อยู่แล้ว การสร้าง interface ก็ทำได้ไม่ยาก เราเพียงแต่กำหนดให้ Calculate เป็นชื่อของ interface ที่เราต้องการใช้ พร้อมทั้งกำหนดให้มี method ต่าง ๆ ที่เราต้องการที่จะให้มีใน class อื่น ๆ เช่น

```
//interface for Shapes
public interface Calculate {
    double perimeter();
    double area();
}
```

ในการเขียน Method ที่อยู่ใน interface นั้นเราไม่จำเป็นที่จะต้องกำหนดนโยบายของการเข้าหา การกำหนดนโยบายจะต้องทำใน class ที่เรียกใช้ (implement) interface นี้

class อื่น ๆ ที่เหลืออยู่ก็ไม่มีการเปลี่ยนแปลงอะไรมากมาย ยกเว้น การคำนวณหาพื้นที่ของรูปทรงต่าง ๆ

ใน class Triangle เราเพิ่ม method

```
//calculating an area of this triangle
//using Huron's formula
public double area() {
    double s = perimeter() / 2.0;
    double area = Math.sqrt((s * (s-side1) * (s-side2) * (s-side3)));

    return area;
}
```

ใน class Circle เราเพิ่ม method

```
//calculate area of this circle
public double area() {
    return Math.PI * radius * radius;
}
```

ใน class Square เราเพิ่ม method

```
//calculate area of this square
public double area() {
    return side * side;
}
```

หลังจากที่เปลี่ยนแปลง code บางส่วนในโปรแกรมทดสอบ ดังนี้

```
//TestShapes.java - driver program

import java.text.DecimalFormat;

class TestShapes {
    public static void main(String[] args) {
        Shapes []shapes = new Shapes[5];

        //instantiate 5 shapes
        for(int i = 0; i < 5; i++)
            shapes[i] = randomShape();

        //display info. about particular shapes
        DecimalFormat f = new DecimalFormat("0.00");
        for(int i = 0; i < 5; i++) {
            System.out.print(shapes[i] + " perimeter = ");
            System.out.print(f.format(shapes[i].perimeter()));
            System.out.println(" area = " +
                               f.format(shapes[i].area()));
        }
    }

    public static Shapes randomShape() {
        switch((int)(Math.random() * 3)) {
            default:
            case 0: return new Circle("circle", Math.random() * 12.0);
            case 1: return new Square("square", Math.random() * 10.0);
            case 2: return new Triangle("triangle", 4.0, 5.0, 6.0);
        }
    }
}
```

ผลลัพธ์ที่ได้คือ

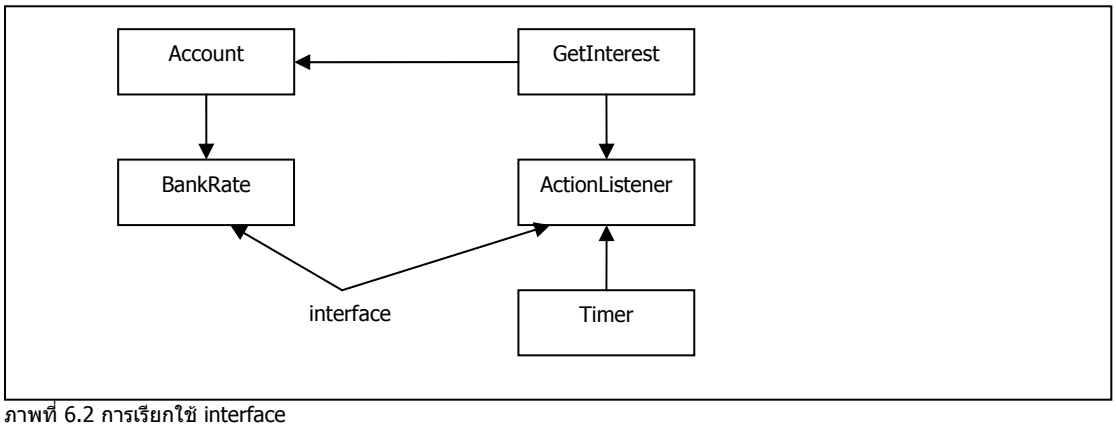
```
I am a triangle perimeter = 15.0 area = 9.90
I am a square perimeter = 10.32 area = 6066
I am a circle perimeter = 45.32 area = 163.46
I am a triangle perimeter = 15.00 area = 9.92
I am a circle perimeter = 6.76 area = 3.64
```

เนื่องจากว่าเราได้ทำการเรียกใช้ interface Calculate ใน class แม่ ดังนั้นเราไม่จำเป็นที่จะต้องเรียกใช้ interface Calculate ใน class ลูกเพราะ class ลูกเหล่านี้ได้รับการถ่ายทอดคุณสมบัติจาก class แม่เรียบร้อยแล้ว เหตุผลอีกอย่างหนึ่งที่เรากำหนดให้ class Shapes มีการ implements Calculate ก็คือ เราไม่ต้องการเปลี่ยนแปลง code ที่เราได้เขียนขึ้นก่อนหน้านี้มากมายนัก

เรามาลองดูตัวอย่างของการใช้ interface ที่ Java มีให้สักตัวอย่างหนึ่ง โดยเราจะเรียกใช้ interface ActionListener ในการตรวจจับเหตุการณ์ที่จะเกิดขึ้นจากตัวโปรแกรม ตัวอย่างนี้อาจมีข้อมูลที่เรายังไม่ได้พูดถึงมากพอสมควร แต่ประเด็นหลักที่ต้องการแสดงก็คือการเรียกใช้ interface เราจะปรับปรุงโปรแกรม Interests.java ที่เราได้เขียนขึ้นในบทที่สาม ซึ่งเป็นโปรแกรมสำหรับการคำนวณหาดอกเบี้ยทบต้น เราจะสร้าง class ขึ้นใหม่ดังนี้

class Account ใช้เป็นการเก็บ balance คำนวณดอกเบี้ย แสดงผล class นี้เรียกใช้ interface BankRate
class Timer เป็น class ที่มีอยู่ใน Java ใช้สำหรับการกระทำที่ต้องใช้เวลา
class Interests1 เป็นโปรแกรมทดสอบส่วนต่าง ๆ ที่เราสร้างขึ้น
interface BankRate ใช้เก็บค่าคงที่ RATE ที่ใช้ในการคำนวณหาดอกเบี้ย
interface ActionListener ใช้ในการตรวจสอบเหตุการณ์ที่เกิดจากผู้ใช้โปรแกรม เช่น ออกจากโปรแกรม

เรากำหนดให้ความสัมพันธ์ของ class เป็นไปตาม diagram ที่เห็นนี้



ภาพที่ 6.2 การเรียกใช้ interface

เราเริ่มต้นด้วย interface BankRate และ class Account

```
//interface for bank rate
public interface BankRate {
    double RATE = 5.0;
}
```

```
//Account.java

public class Account implements BankRate {
    private double balance;

    //constructor - initializing balance
    Account(double balance) {
        this.balance = balance;
    }

    //method to get balance
    protected double getBalance() {
        return balance;
    }

    //method to deposit
```

```
    protected void deposit(double amount) {
        balance += amount;
    }

    //method to calculate interest
    protected double interest() {
        return balance * RATE / 100.0;
    }

    //method to display info. about this account
    public String toString() {
        return "Balance = " + getBalance();
    }
}
```

เราเขียน class Account อย่างง่าย ๆ โดยกำหนดให้มี method สำหรับการกำหนด balance การคำนวณหาดอกเบี้ย และการแสดงผลผ่านทาง method toString() ต่อไปเราจะมาดู class Interests1 ซึ่งเป็นโปรแกรมทดสอบการใช้งานของ class และ interface

```
//Interests1.java

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import javax.swing.Timer;

class Interests1 {
    public static void main(String[] args) {
        //create account with initial balance of 1000
        final Account acc = new Account(1000);

        //inner class performing interest calculation
        //and listening to event from user
        //stop performing when user hit stop button in
        //the dialog window
        class GetInterest implements ActionListener {
            public void actionPerformed(ActionEvent event) {
                double ints = acc.interest();
                acc.deposit(ints);
                System.out.println(acc);
            }
        }

        //create object from class GetInterest
        GetInterest intsPeek = new GetInterest();
        //set performing time interval to one second
        final int DELAY = 1000;
        //calling actionPerformed() every 1000 milliseconds
        Timer t = new Timer(DELAY, intsPeek);
        //start the timer
        t.start();

        //display dialog window - stop performing and exit
        //program when user hit the button
        JOptionPane.showMessageDialog(null, "Stop?");
        System.exit(0);
    }
}
```

ในโปรแกรม Interests1.java เราสร้าง object acc จาก class Account ด้วยค่า balance เริ่มต้นที่ 1000 ภายใน method main() เราสร้าง class GetInterest ที่เรียกใช้ interface ActionListener ของ Java ซึ่งจะคอยตรวจสอบเหตุการณ์ (event) ที่อาจเกิดขึ้นจาก user พร้อมทั้งการคำนวณหาดอกเบี้ย

method actionPerformed() จะถูกเรียกโดย object intsPeek จาก class Timer ด้วยค่า delay 1 วินาที ด้วยประโยคนี้

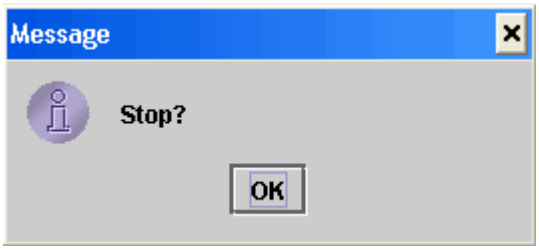
```
Timer t = new Timer(DELAY, intsPeek);
```

แต่กระบวนการทั้งหมดจะเริ่มได้ด้วยการเรียก method start() ของ class Timer ดังนี้ t.start()

โปรแกรมของเราจะคำนวณหาดอกเบี้ยไปจนกว่า user จะกดปุ่ม OK ที่แสดงไว้ใน dialog window ที่ปรากฏอยู่ในหน้าจอ ซึ่ง dialog window นี้ถูกสร้างขึ้นมาด้วยคำสั่ง

```
JOptionPane.showMessageDialog(null, "Stop?");
```

ซึ่งมีหน้าตาดังนี้



เมื่อ user กดปุ่ม OK เราก็จะยุติโปรแกรม และออกจากระบบด้วยคำสั่ง System.exit(0);

ตัวอย่างผลลัพธ์ที่ได้จากการ run โปรแกรมในช่วงเวลาหนึ่ง ก่อนการกดปุ่ม คือ

```
Balance = 1050.0
Balance = 1102.5
Balance = 1157.625
Balance = 1215.50625
Balance = 1276.2815624999998
Balance = 1340.0956406249998
Balance = 1407.1004226562497
Balance = 1477.4554437890622
Balance = 1551.3282159785153
Balance = 1628.894626777441
Balance = 1710.3393581163132
Balance = 1795.8563260221288
Balance = 1885.649142323235
Balance = 1979.931599439397
```

เราต้องการที่จะแสดงให้เห็นถึงการใช้ interface ทั้งที่มีอยู่ใน Java และที่เขียนขึ้นเอง ส่วนการใช้ส่วนประกอบอื่น ๆ ที่ Java มีให้เป็นเพียงสิ่งที่ผู้อ่านจะได้สัมผัสในโอกาสต่อไป

เรามีทางเลือกสองทางในการสร้าง และกำหนดหน้าที่ของ method ดังที่เราได้พูดไว้ คือ

1. กำหนดให้ class แม่เป็น abstract class พร้อมทั้งกำหนดให้มี ชื่อของ method ที่ต้องการ การกำหนดหน้าที่ของ method จะเป็นภาระของ class ลูกเอง

2. กำหนดให้ class เรียกใช้ interface ที่มีการกำหนดชื่อของ method และ class ต้องกำหนดหน้าที่ของ method นั้นเอง

สรุป

เราได้พูดถึงการสร้าง class ใหม่จาก class ที่มีอยู่ก่อนแล้ว การถ่ายทอดคุณสมบัติจาก class แม่สู่ class ลูก การ override method การใช้ polymorphism การสร้างและใช้ interface โดยสรุปแล้วสิ่งสำคัญที่เราได้พูดไว้คือ

- ✓ เราสามารถสร้าง class ใหม่จาก class ที่มีอยู่ก่อนแล้วด้วยการใช้คำว่า extends ซึ่ง class ที่เรา extends มานั้นเราเรียกว่า base class หรือ parent ส่วน class ที่เราสร้างขึ้นใหม่เราเรียกว่า sub class หรือ child
- ✓ เราสร้าง abstract class ด้วยการใช้คำว่า abstract ซึ่งภายใน class ที่เป็น abstract class นั้นจะมี method ประกาศอยู่ แต่หน้าที่ของ method ยังไม่ได้ถูกกำหนด การกำหนดจะทำใน class ลูกเดท่านั้น และเราไม่สามารถที่จะสร้าง object จาก abstract class ได้
- ✓ sub class ไม่ได้รับการถ่ายทอด constructor ของ super class
- ✓ ถ้าเรากำหนดนโยบายการเข้าหาของ method หรือตัวแปรให้เป็น private ใน class แม่ class ลูกจะไม่สามารถเข้าหาตัวแปร หรือ method ได้
- ✓ ตัวแปรที่ประกาศจาก class แม่สามารถที่จะใช้เป็นตัวชี้ไปยัง object ของ class ลูกได้ ซึ่งทำให้การเรียกใช้ method ที่อยู่ใน class ลูกเป็นไปได้
- ✓ abstract class เป็น class ที่ไม่ต้องกำหนดหน้าที่ให้ method ทุก ๆ ตัวใน class
- ✓ interface ประกอบไปด้วย ค่าคงที่ abstract method และ inner class
- ✓ class ที่ไม่กำหนดหน้าที่ให้กับ method ทุกตัวที่ตัวมันเองต้องใช้ interface จะต้องประกาศให้เป็น abstract class

แบบฝึกหัด

1. จากข้อมูลที่ให้เป็นคู่ในแต่ละข้อ จงแสดงถึงความสัมพันธ์ว่าส่วนไหนเป็น super class และส่วนไหนเป็น sub class

ผู้จัดการ	ลูกจ้าง
นักศึกษา	อาจารย์
คน	นักศึกษา
อาจารย์	บุคลากร
สามี	ภรรยา
พ่อ	ลูก

2. สมมติว่า class RumRaisin extends มาจาก class Icecream จงหาว่าประโยคใดต่อไปนี้เป็นประโยคที่ Java ยอมรับ

```
RumRaisin scoop1 = new RumRaisin();
Icecream scoop2 = new Icecream;
scoop2 = scoop1;
scoop1 = scoop2;
scoop2 = new RumRaisin();
scoop1 = new Icecream();
```

3. จงวาด diagram ที่แสดงถึงความสัมพันธ์ของการถ่ายทอดคุณสมบัติของ class ที่ให้นี้

   บุคลากร
   ผู้จัดการ
   นักศึกษา
   อาจารย์
   ห้องเรียน
   อุปกรณ์ประกอยการเรียน

รองอธิการฝ่ายวิชาการ
รองอธิการฝ่ายบริหาร
บรรณารักษ์
เจ้าหน้าที่การเงิน
หัวหน้าฝ่ายการเงิน

4. **เราได้พูดถึงการใช้ method clone() ในบทที่เราพูดถึงเรื่อง array จง override method clone() เพื่อทำการสร้าง object ใหม่จาก class Shapes ที่ได้กล่าวไว้ในบทนี้**

5. **จงเพิ่ม method withdraw() ที่ทำหน้าที่ในการถอนเงินออกจากบัญชี ของ class Account ที่ได้กล่าวไว้ในบทนี้ ให้เขียนโปรแกรมทดสอบ**

6. **จาก class Shapes ในบทนี้ class ที่ไม่ได้พูดถึงและไม่มีการกำหนดหน้าที่ใด ๆ เลย คือ class Rectangle จงเขียน code ที่ทำให้ class Rectangle นี้สมบรูณ์**

7. **จงออกแบบ class Student ที่เก็บข้อมูลของนักศึกษาวิทยาลัยแห่งหนึ่ง โดยให้มี sub class ที่มีทั้งนักศึกษาระดับปริญญาตรี และระดับปริญญาโท รวมไปถึงนักศึกษาภาคพิเศษ พร้อมทั้งเขียน method ในการกำหนด และเข้าหาข้อมูลเหล่านี้ กำหนดให้มี method toString() ในทุก class สำหรับการแสดงข้อมูลของ object ที่เกิดจาก class นั้น ๆ ให้เขียนโปรแกรมทดสอบ**

8. **จงออกแบบ class Employee ที่มี sub class เป็น TempEmployee และ RegularEmployee โดยกำหนดให้ลูกจ้างทุกคนมี ชื่อ และอัตราจ้างงานที่คิดเป็นจำนวน บาท ต่อชั่วโมง จงเขียน method ที่ทำการคำนวณหาค่าจ้างของลูกจ้างทุกคน โดยกำหนดให้การคิดค่าจ้างของ TempEmployee ขึ้นอยู่กับจำนวนชั่วโมงที่ทำ หากทำเกิน 40 ชั่วโมงให้คิดค่าจ้างเพิ่มเป็น หนึ่งเท่าครึ่งของอัตราจ้างประจำของจำนวนชั่วโมงที่ทำเกิน ส่วนลูกจ้างประจำ (RegularEmployee) จะได้ค่าจ้างตามอัตราที่ได้ตกลงไว้ล่วงหน้าโดยไม่คำนึงถึงจำนวนชั่วโมงของงานที่ทำ จงใช้เทคนิคของ polymorphism ในการออกแบบ ให้เขียนโปรแกรมทดสอบ**

   * class Employee ควรเป็น abstract class

9. **จงปรับปรุง class Shapes หรือ sub class ที่มาจาก Shapes ให้มี method ในการสร้าง object จาก class หนึ่งให้เป็น object จากอีก class หนึ่ง ดังเช่นที่ทำเป็นตัวอย่างในการสร้าง object Circle จาก Square จงเขียนโปรแกรมทดสอบ**

7

การตรวจสอบและดักจับ error (Exceptions)

ในบทที่เจ็ดนี้เราจะพูดถึงการใช้ และการออกแบบ exception หรือที่เรียกว่าการควบคุมและดักจับ error ที่อาจเกิดขึ้นในการ compile และ run โปรแกรม

หลังจากจบบทนี้แล้ว ผู้อ่านจะได้ทราบถึง

- ความหมายของ exception
- วิธีการใช้และควบคุม exception
- การใช้ throws และ try
- การออกแบบ และใช้ exception ที่เขียนขึ้นเอง

ในภาษา Java นั้น exception เป็นการบอกถึงสิ่งผิดปรกติที่เกิดขึ้นในโปรแกรม ซึ่งอาจเป็น error หรือเหตุการณ์ที่ไม่ค่อยเข้าท่าที่ต้องการความดูแลเป็นพิเศษ โดยทั่วไปในการเขียนโปรแกรมที่ดีนั้น เราจะต้องตรวจสอบถึงเหตุการณ์ที่อาจทำให้โปรแกรมของเราล้มเหลวในการทำงาน เช่น ข้อมูลถูกหารด้วยศูนย์ เข้าหาข้อมูลใน array ด้วยการใช้ index ที่ไม่มีอยู่จริง หรือ อ้างถึงหน่วยความจำที่เป็น null เป็นต้น

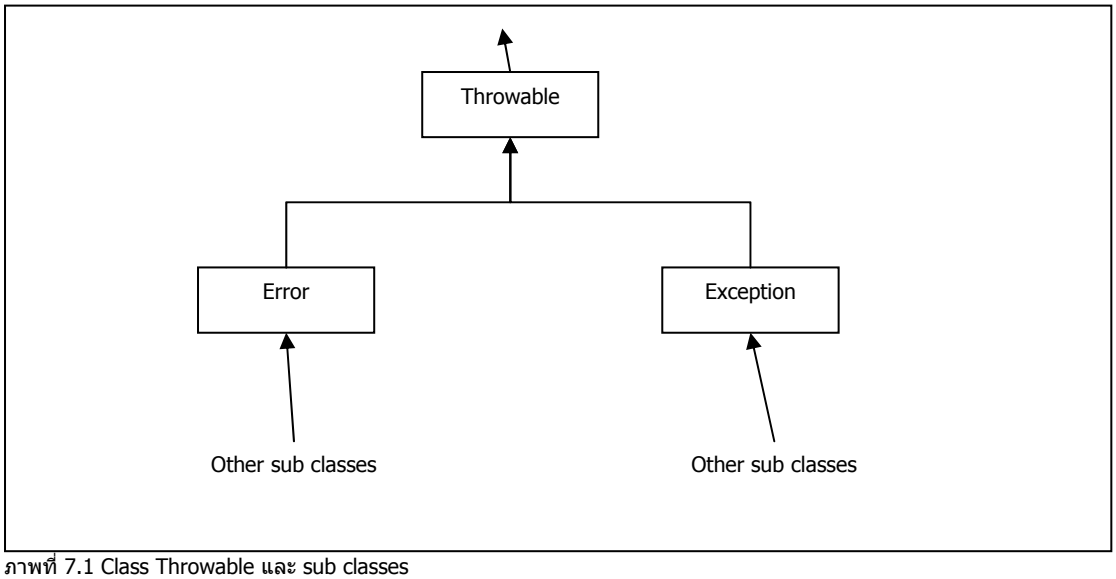
ถึงแม้ว่าเรามีวิธีการตรวจสอบ error ต่าง ๆ เหล่านี้ด้วยการใช้ if-else หรือ การตรวจสอบอื่น ๆ ที่ทำให้โปรแกรมของเราทำงานได้ราบรื่น แต่จะทำให้ code ของเราดูแล้ววุ่นวายเพราะถ้ามีการตรวจสอบ error มาก code ของเราก็จะดูซับซ้อนมากยิ่งขึ้น แต่ไม่ได้หมายความว่าเราจะไม่ตรวจสอบ และดักจับ error เหล่านี้ การนำเอา exception เข้ามาเป็นตัวช่วยในการตรวจสอบและดักจับ ทำให้เกิดการแยกส่วนของ code ที่ทำงานได้ราบรื่น ออกจากส่วนของ code ที่จัดการเกี่ยวกับ error ทำให้เราสามารถที่จะค้นหาส่วนของ code ทั้งสองได้ง่ายขึ้น ถ้ามีการเปลี่ยนแปลง code ในอนาคต ข้อดีอีกอันหนึ่งของ exception ก็คือ ทำให้การตรวจสอบและดักจับเป็นไปอย่างเฉพาะเจาะจง ตรงกับ error ที่เกิดขึ้น ทำให้การแก้ไขเป็นไปอย่างถูกต้อง และเนื่องจากว่า error มีหลากหลายรูปแบบ เราต้องเขียน code ขึ้นมารองรับไม่เช่นนั้นแล้วโปรแกรมของเราก็จะไม่ผ่านการ compile

เราไม่จำเป็นที่จะต้องใช้ exception ในการตรวจสอบและดักจับ error ในโปรแกรมเสมอไป เพราะการใช้ exception จะเสียค่าใช้จ่ายในการประมวลผลมาก ทำให้โปรแกรมทำงานช้าลง ดังนั้นเราจะต้องตัดสินใจให้ดีว่าควรจะใช้ exception ในกรณีไหน อย่างไร ตัวอย่างของโปรแกรมที่ไม่ต้องใช้ exception ก็น่าจะเป็นการใส่ข้อมูลนำเข้าแบบผิด ๆ ของ user ซึ่งถือเป็นเรื่องปกติ ถ้าเรามัวแต่เสียเวลาในการดักจับด้วยการใช้ exception แทนการตรวจสอบและดักจับทั่วไปโปรแกรมของเราก็จะเสียเวลาในการประมวลผลส่วนอื่น ๆ ไป

การใช้ exception ใน Java นั้นจะโดยทั่วไปจะเป็นการโยน (throw) การจัดการ error ที่เกิดขึ้นให้กับส่วนของ code ที่ทำหน้าที่ในด้านการจัดการกับ error นั้น ๆ ที่เกิดขึ้น ซึ่ง code ส่วนนี้จะคอยดักจับ (catch) ถ้ามีการโยนเกิดขึ้น แต่ไม่จำเป็นที่ code ส่วนนี้จะต้องแก้ไข error ที่เกิดขึ้นอาจเป็นเพียงการดักจับเท่านั้น เพื่อให้โปรแกรมทำงานต่อไปได้เท่านั้น

เราคงไม่พูดถึงข้อมูลต่าง ๆ ที่เกี่ยวกับ error และ exception มากนักแต่จะเน้นการใช้ exception ในการตรวจสอบและดักจับ error แบบพอประมาณ ผู้ที่สนใจก็สามารถที่จะหาอ่านได้จากหนังสือ Java ทั่ว ๆ ไป หรือใน web sit ของ Java เอง

Exception โดยปกติจะเป็น object จาก sub class (class ใด class หนึ่ง) ของ class Throwable ที่ Java มีให้ และ class หลัก ๆ สอง class ที่เกิดจาก class Throwable โดยตรงที่เป็น class หลักของ class ที่เป็น class เฉพาะของการจัดการกับ error คือ class Error และ class Exception (ภาพที่ 7.1)



ภาพที่ 7.1 Class Throwable และ sub classes

โดยการออกแบบของ Java เราไม่ควรที่จะ catch error ที่เกิดขึ้นจากการตรวจสอบของ class Error เพราะ error ที่เกิดขึ้นจะเป็น error ที่ โดยทั่วไปจะไม่เกิดขึ้น (หรือไม่คาดว่าจะเกิด) ซึ่งมีอยู่สาม class คือ ThreadDeath LinkageError และ VirtualMachineError เราคงจะไม่เข้าไปวุ่นวายกับ class ทั้งสาม เพราะว่า เราไม่สามารถที่จะแก้ไข error ที่เกิดขึ้นนี้ได้โดยตรง และต้องใช้การตรวจสอบอย่างละเอียดกับ error ที่เกิดขึ้น (แต่คงต้องหวังในใจว่า error ที่ว่าคงไม่เกิดขึ้นกับเรา)

เราจะมาดูตัวอย่างการใช้ exception ในการจัดการกับ error ที่เกิดขึ้นในโปรแกรมของเรา ด้วยโปรแกรมตัวอย่างต่อไปนี้

```
//ThrowException.java

class ThrowException {
    public static void main(String[] args) throws Exception {
        int number = 10;
        int divider = 0;

        System.out.println("Divide " + number + " by " + divider);
        int result = number / divider;
        System.out.println("Result is " + result);
    }
}
```

โปรแกรมตัวอย่างด้านบนนี้ทำการ throw exception ซึ่งเป็น exception ที่เกิดขึ้นในขณะที่โปรแกรมกำลังได้รับการประมวลผล (เช่น การหารตัวเลขที่เป็น int ด้วย 0 – เราตั้งใจให้ error นี้เกิดขึ้น) โปรแกรมตัวอย่างของเราไม่ได้ทำการดักจับใด ๆ ทำแต่เฉพาะการตรวจสอบ error ที่เกิดขึ้นเท่านั้น ดังที่เห็นจากผลลัพธ์ของการ run นี้

```
Divide 10 by 0
java.lang.ArithmeticException: / by zero
    at ThrowException.main(ThrowException.java:10)
```

จะเห็นว่า Java จะฟ้องให้เรารู้ว่า error ที่เกิดขึ้นเป็น error ชนิดไหน (ArithmeticException) ที่พยายามหาร int ด้วย 0 พร้อมกับบอกว่าเกิดขึ้นในบรรทัดที่เท่าไร หลังจากนั้นก็ยุติการทำงาน

ถ้าหากว่าเราต้องการที่จะแก้ไข หรือทำการใด ๆ สักอย่างหนึ่งกับ error ที่เกิดขึ้นเราต้องใช้ try และ catch ดังตัวอย่างต่อไปนี้

```
//TryAndCatchExample1.java
//adopted from Ivor Horton's

class TryAndCatchExample1{
    public static void main(String[] args) {
        int number1 = 15;
        int number2 = 0;

        try {
            System.out.println("Try to divide by zero in
                                a try block.");
            int result = number1 / number2;
            System.out.println("Leaving a try block.");
        }
        catch(ArithmeticException ex) {
            System.out.println("Exception caught in a catch block.");
        }

        System.out.println("After a try block.");
    }
}
```

โปรแกรม TryAndCatchExample1.java เป็นโปรแกรมตัวอย่างการใช้ try และ catch ในการจัดการกับการหาร int ด้วย 0 (เราจะไม่ใช้การ throws exception เหมือนกับตัวอย่างแรก) เรากำหนดให้ number1 มีค่าเป็น 15 และ number2 มีค่าเป็น 0 พร้อมทั้งกำหนดให้

```
result = number1 / number2;
```

ใน block ของ try ซึ่งประโยคดังกล่าวจะทำให้เกิดการโยน exception ไปยัง block ของ catch และใน block ของ catch เราจะดักด้วย exception ที่ชื่อ ArithmaticException เราไม่ได้ทำอะไรมากมายไปกว่าการแสดงข้อความว่ามีการดักจับ error ผลลัพธ์ที่ได้จากการ run คือ

```
Try to divide by zero in a try block.
Exception caught in a catch block.
After a try block.
```

จากผลลัพธ์ที่ได้ code ที่ได้รับการประมวลผลคือ ประโยคที่อยู่ใน block ของ try 2 ประโยคแรก โดยเฉพาะประโยคที่สอง ที่ทำให้การประมวลกระโดดไปยัง block ของ catch ทำให้ประโยคที่สามไม่ได้รับการประมวลผล และหลังจากที่ประมวลผลประโยคที่อยู่ใน block ของ catch แล้ว การประมวลผลประโยคสุดท้ายที่ตามหลัง block ของ catch จึงเกิดขึ้น

เราสามารถใช้ ตัวแปร ex ที่เราได้ประกาศใน parameter list ของ method catch() แสดงถึงข้อความที่บ่งบอกเกี่ยวกับ error ที่เกิดขึ้น ด้วยการเพิ่มประโยคต่อไปนี้ใน block ของ catch

```
System.out.println(ex.getMessage());
ex.printStackTrace();
```

ประโยคแรกเรียกใช้ method getMessage() ที่เกิดขึ้นทำการแสดงผลไปยังหน้าจอ ส่วนประโยคที่สองเป็นการตรวจสอบถึงที่มาของ error ที่เกิดขึ้น ดังที่แสดงให้เห็นจากการ run โปรแกรมอีกครั้งหลังจากการเปลี่ยนแปลง code ใน block ของ catch

```
Try to divide by zero in a try block.
Exception caught in a catch block.

/ by zero
java.lang.ArithmeticException: / by zero
```

```
    at TryAndCatchExample1.main(TryAndCatchExample1.java:11)

After a try block.
```

เราไม่จำเป็นที่จะต้องแสดงผลที่เกิดขึ้นไปยังหน้าจอ ส่วนใหญ่แล้วข้อมูลเหล่านี้จะเป็นประโยชน์ต่อการตรวจสอบถึง error ที่เกิดขึ้นเท่านั้นเอง สิ่งที่สำคัญที่เราต้องกังวลก็คือ การตรวจสอบและดักจับ error ที่อาจเกิดขึ้น

Java กำหนดให้ try และ catch เป็นของคู่กัน เราไม่สามารถที่จะแยก block สองออกจากกันได้ ถ้าเรากำหนดให้มีประโยคอะไรก็ได้สักประโยคหนึ่งระหว่าง block ของ try และ catch โปรแกรมของเราจะ compile ไม่ผ่าน ดังตัวอย่างที่มีการเพิ่มประโยคระหว่าง block ทั้งสองนี้

```
    …
    …
    System.out.println("Leaving a try block.");
}
System.out.pritnln("In between try and catch blocks.");
catch(ArithmeticException ex) {
    …
    …
```

ถ้าเรา compile เราจะได้ error ดังที่เห็นนี้

```
TryAndCatchExample1.java:9: 'try' without 'catch' or 'finally'
       try {
                ^
TryAndCatchExample1.java:15: 'catch' without 'try'
       catch(ArithmeticException ex) {
                ^
2 errors
```

ตัวอย่างการใช้ try และ catch ที่ classic อีกอันหนึ่งก็คือ การใช้ try และ catch ใน loop ดังตัวอย่างต่อไปนี้

```
//TryAndCatchExample2.java
//adopted from Ivor Horton's

class TryAndCatchExample2 {
    public static void main(String[] args) {
        int number = 10;

        for(int index = 5; index >= -1; index--)
            try {
                System.out.println("try block: index = " + index);
                int result = number / index;
                System.out.println("Leaving a try block.");
            }
            catch(ArithmeticException e) {
                System.out.println("Exception caught in
                                    a catch block.");
                System.out.println(e.getMessage());
                e.printStackTrace();
            }

            System.out.println("After a try and catch blocks.");
    }
}
```

ก่อนที่จะอธิบายถึงการทำงานของตัวโปรแกรม เรามาดูผลลัพธ์ที่ได้จากการ run

```
try block: index = 5
Leaving a try block.
try block: index = 4
Leaving a try block.
try block: index = 3
Leaving a try block.
try block: index = 2
Leaving a try block.
try block: index = 1
Leaving a try block.
try block: index = 0
Exception caught in a catch block.
/ by zero
java.lang.ArithmeticException: / by zero
    at TryAndCatchExample2.main(TryAndCatchExample2.java:11)
try block: index = -1
Leaving a try block.
After a try and catch blocks.
```

ประโยคใน block ของ try จะถูกประมวลผลจนกว่า ค่าของ index จะเป็น 0 จึงจะกระโดดไปประมวลผลประโยคที่อยู่ใน block ของ catch หลังจากนั้นก็กลับไปประมวลผลประโยคที่เหลืออยู่ใน block ของ try และประโยคที่อยู่ท้ายสุดในโปรแกรม อีกประโยคหนึ่ง

จะเห็นได้ว่าเรามีทางเลือกอยู่สองทางในการจัดการกับ error ทางแรกคือ ยุติการทำงาน (terminate) ของโปรแกรม หรือ code ส่วนนั้น ๆ ที่ทำให้เกิด error หรือ ทางที่สอง ซึ่งเป็นทางเลือกที่ทำให้โปรแกรมหรือ code ส่วนนั้น ๆ กลับไปรับการประมวลผลใหม่ (resumption) ทั้งสองวิธีเป็นการตรวจสอบและดักจับ error ที่ดีพอกัน ทั้งนี้ก็ขึ้นอยู่กับลักษณะของงานที่ทำอยู่ ลองมาดูตัวอย่างอีกสักตัวหนึ่งในการใช้ class NumberFormatException ในการดักจับ error

```
//NumFormatException.java

import java.lang.Integer;

class NumFormatException {
    public static void main(String[] args) {
        int inputNum, sum = 0;

        //reading input from argument list
        //only those in [0..9]
        for(int i = 0; i < args.length; i++) {
            try {
                inputNum = Integer.parseInt(args[i]);
                sum += inputNum;
            }
            //ignore input that's not a number
            catch(NumberFormatException e) {
                e.printStackTrace(System.out);
            }
        }
        System.out.println("Sum is " + sum);
    }
}
```

เรากำหนดให้โปรแกรมของเรารับข้อมูลจาก command line argument ซึ่งเป็นวิธีการที่ยอมให้ผู้ใช้ใส่ข้อมูลตามหลังชื่อของโปรแกรม (ดูจากผลลัพธ์การ run) เราจะทำการหาผลรวมของตัวเลขเหล่านั้น

ทั้งหมดจนกว่าผู้ใช้จะพอใจ คือ กดปุ่ม <enter> หลังจากใส่ข้อความจนพอใจ ใน catch block เราดักจับด้วย NumberFormatException ซึ่งจะไม่ยอมรับข้อมูลนำเข้าที่ไม่ใช่ตัวเลขที่เป็น int และทุกครั้งที่เจอข้อมูลแบบนี้โปรแกรมจะส่งรายละเอียดไปให้ผู้ใช้ และจะหาผลรวมของข้อมูลที่ถูกต้องนั้น ดังที่แสดงไว้ด้านล่างนี้

```
>java NumFormatException 1 2 3 4 p 5 6
java.lang.NumberFormatException: For input string: "p"
        at
java.lang.NumberFormatException.forInputString(NumberFormatException.
java:48)
        at java.lang.Integer.parseInt(Integer.java:426)
        at java.lang.Integer.parseInt(Integer.java:476)
        at NumFormatException.main(NumFormatException.java:12)
Sum is 21
```

ในการสร้าง block สำหรับการ catch error ที่เกิดขึ้นเราไม่จำเป็นที่จะต้องมี catch block เพียง block เดียว เราอาจมีมากกว่าหนึ่ง catch block ได้แต่มีข้อแม้ว่า การ catch exception ต้องทำจาก sub class ออกไปหา super class มิฉะนั้นแล้วการดักจับในหลาย ๆ block อาจไม่เกิดขึ้น เช่น

```
try {
    …
    …
}
catch(Exception e) {
    …
    …
}
catch(ArithmeticException ex) {
    …
    ..
}
```

เนื่องจากว่า class ArithmeticException เป็น class ที่เกิดมาจาก class Exception ดังนั้น error ที่เกิดขึ้นก็จะถูกดักจับใน catch block ทั้งหมด การดักจับใน catch block ที่สองจึงไม่เกิดขึ้น

เรามาลองดูตัวอย่างการดักจับ error ในรูปแบบนี้กันดู

```
//MultipleCatches.java

class MultipleCatches {
    public static void main(String[] args) {
        //force divide by zero at index = 2
        int []number = {1, 2, 0, 4};
        int num = 12;

        //force array index out of bound at index = 4
        for(int i = 0; i < number.length + 1; i++) {
            try {
                num /= number[i];
                System.out.println("result = " + num);
            }
            catch(ArithmeticException e) {
                System.out.println("error message: " +
                                   e.getMessage());
                e.printStackTrace();
            }
            catch(ArrayIndexOutOfBoundsException e) {
```

```
                System.out.println("error message: " +
                                    e.getMessage());
                e.printStackTrace();
            }
        }
        System.out.println("After a for loop.");
    }
}
```

เราต้องการที่จะบังคับให้เกิด error สองอย่างคือ หารด้วยศูนย์ และเข้าหา index ของ array ที่ไม่มีอยู่จริง โดยเรากำหนดให้ข้อมูลที่ index = 2 มีค่าเป็นศูนย์ และให้ for loop ทำงานเกินไปหนึ่งครั้ง การ run โปรแกรมตัวอย่างของเราทำให้เกิด ผลลัพธ์ดังนี้คือ

```
result = 12
result = 6

error message: / by zero
java.lang.ArithmeticException: / by zero
    at MultipleCatches.main(MultipleCatches.java:11)

result = 1

error message: 4
java.lang.ArrayIndexOutOfBoundsException: 4
    at MultipleCatches.main(MultipleCatches.java:11)

After a for loop.
```

จะเห็นว่าโปรแกรมของเราดักจับ error ได้ตามที่เราตั้งใจไว้ การหารด้วยศูนย์ และการอ้างถึง index ของ array ที่ไม่มีอยู่จริง ถูกดักไว้ทั้งสองตัว ถ้าเป็นการจัดการที่เป็นการพัฒนาโปรแกรมจริง ๆ เราคงทำแค่การดักจับไม่ได้ คงต้องมีกระบวนการอื่น ๆ ในการทำให้โปรแกรมทำงานได้อย่างราบรื่น และกระบวนการจัดการกับ error ที่เกิดขึ้นก็ต้องมีความซับซ้อนเพิ่มขึ้น

Java ยังมี block อีก block หนึ่งที่เอาไว้ catch error ที่เกิดขึ้น ที่ซึ่งเป็น block สุดท้ายของการ catch ทั้งหมด โดยกำหนดให้ block นี้ทีชื่อว่า finally

Finally เป็นการดักจับ error สุดท้ายของการดักทั้งหมด ดังนั้น finally จะอยู่ที่อื่นไม่ได้นอกจาก block สุดท้ายของ catch block ทั้งหมด เราจะแสดงให้ดู ด้วยการนำเอา finally ไปใส่ไว้เป็น block สุดท้ายในตัวอย่างก่อนหน้านี้ ดังนี้

```
    …
    …
}
catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("error message: " + e.getMessage());
    e.printStackTrace();
}
finally {
    System.out.println("In finally block.");
}
    …
    …
```

finally block ที่เราใช้จะถูกประมวลผลเสมอ ไม่ว่าอะไรจะเกิดขึ้นก็ตามใน block อื่น ๆ ก่อนหน้า ดังตัวอย่างของผลลัพธ์ที่แสดงให้ดูบางส่วนนี้

```
    …
```

```
    …
    at MultipleCatches.main(MultipleCatches.java:13)
In finally block.
After a for loop.
```

โดยทั่วไปหน้าที่หลักของ finally จะเป็นการเก็บกวาด code ที่โปรแกรมต้องการประมวลผล (เช่น การปิดเปิดไฟล์)ก่อนออกจาก ตัว method main() และกลับออกไปสู่ระบบในที่สุด

```
//ThrowsWithTry.java

import java.io.*;
import java.lang.Integer;

class ThrowsWithTry {
    public static void main(String[] args) throws IOException {
        BufferedReader buffer;
        InputStreamReader isr;
        String input;
        int first, divider;

        //keep looping until error occur
        while(true) {
            System.out.print("Enter a number: ");
            isr = new InputStreamReader(System.in);
            buffer = new BufferedReader(isr);

            //get first number
            input = buffer.readLine();
            first = Integer.parseInt(input);

            //get second number
            System.out.print("Enter a divider: ");
            input = buffer.readLine();
            divider = Integer.parseInt(input);

            try {
                double result = first / divider;
                System.out.println("Result = " + result);
            }
            catch(ArithmeticException e) {
                System.out.println("Exception: " + e.getMessage());
                e.printStackTrace();
            }
        }
    }
}
```

โปรแกรม ThrowsWithTry.java เป็นโปรแกรมที่ต้องการแสดงการใช้ throws และ try – catch ควบคู่กัน โดยตัวโปรแกรมเองจะทำการอ่านข้อมูลสองตัวจาก keyboard ซึ่งกำหนดให้ข้อมูลทั้งสองตัวเป็น int และจะทำการหาผลลัพธ์ของการหารข้อมูลตัวแรก ด้วยข้อมูลตัวที่สอง

โปรแกรมจะหยุดการทำงานก็ต่อเมื่อผู้ใช้ใส่ข้อมูลที่ไม่ใช่ตัวเลข และจะทำการฟ้องถ้าข้อมูลตัวที่สองของการหารมีค่าเป็นศูนย์ ผลลัพธ์ที่ได้จากการ run คือ

```
Enter a number: 23
Enter a divider: 5
Result = 4.0
Enter a number: 56
```

```
Enter a divider: 0
Exception: / by zero
java.lang.ArithmeticException: / by zero
        at ThrowsWithTry.main(ThrowsWithTry.java:27)
Enter a number: 12
Enter a divider: 3
Result = 4.0
Enter a number: pop
Exception in thread "main" java.lang.NumberFormatException: For input
string: "pop"
        at
java.lang.NumberFormatException.forInputString(NumberFormatException.
java:48)
        at java.lang.Integer.parseInt(Integer.java:426)
        at java.lang.Integer.parseInt(Integer.java:476)
        at ThrowsWithTry.main(ThrowsWithTry.java:20)
```

จากผลลัพธ์ที่ได้จะเห็นว่าโปรแกรมของเราจะฟ้องถ้ามีการหารด้วยศูนย์ และหยุดถ้าข้อมูลไม่ใช่ int (ในตัวอย่างนี้ผู้ใช้ใส่คำว่า "pop") โปรแกรมของเราจะทำงานไปเรื่อย ๆ ถ้าผู้ใช้ใส่ข้อมูลที่ถูกต้องตามที่โปรแกรมได้กำหนดไว้ แต่นี่ไม่ใช่การตรวจสอบที่ดีเท่าไรนัก ถ้าเราต้องการที่จะบอกผู้ใช้ว่าข้อมูลไม่ถูกต้อง และให้ใส่ข้อมูลใหม่เราก็ต้องแก้ไข code ของเราให้รองรับการทำใหม่ ซึ่งอาจทำได้ดังนี้

```
//ThrowsWithTry1.java

import java.io.*;
import java.lang.Integer;

class ThrowsWithTry1 {
    public static void main(String[] args) throws IOException {
        BufferedReader buffer;
        InputStreamReader isr;
        String input;
        int first = 0, divider = 0;

        while(true) {
            System.out.print("Enter a number: ");
            isr = new InputStreamReader(System.in);
            buffer = new BufferedReader(isr);

            //get first number
            input = buffer.readLine();
            first = readInt(input);

            //get second number only if first number is valid
            if(first != -1) {
                System.out.print("Enter a divider: ");
                input = buffer.readLine();
                divider = readInt(input);
            }

            //perform division if both numbers are valid
            if(first != -1 && divider != -1) {
                try {
                    double result = first / divider;
                    System.out.println("Result = " + result);
                }
                catch(ArithmeticException e) {
                    System.out.println("Exception: " + e.getMessage());
```

```
                    e.printStackTrace();
                }
            }
        }
    }

    //method to convert an int from a string
    public static int readInt(String in) throws IOException {
        try {
            int num = Integer.parseInt(in);
            return num;
        }
        catch(NumberFormatException e) {
            System.out.println("Exception caught: " + e.getMessage());
            return -1;
        }
    }
}
```

เราได้เขียน method readInt() สำหรับการเปลี่ยน string ที่อ่านจาก buffer ให้เป็น int โดยเราได้ใช้ try และ catch ในการตรวจสอบและดักจับ error ที่อาจเกิดขึ้นจากการใส่ข้อมูลผิดพลาดที่อาจเกิดขึ้นของผู้ใช้ และเรากำหนดให้ส่งค่า -1 กลับออกไปเพื่อใช้เป็นตัวบอกให้ code ใน while loop รู้ว่าข้อมูลไม่ถูกต้อง เพื่อที่เราจะได้ไม่ต้องอ่านข้อมูลตัวที่สอง ในส่วนของ code ใน while loop เราได้ทำการเปลี่ยนแปลงการอ่าน และ การประมวลผลให้เป็น

```
//get second number only if first number is valid
if(first != -1) {
    System.out.print("Enter a divider: ");
    input = buffer.readLine();
    divider = readInt(input);
}

//perform division if both numbers are valid
if(first != -1 && divider != -1) {
    try {
        double result = first / divider;
        System.out.println("Result = " + result);
    }
    catch(ArithmeticException e) {
        System.out.println("Exception: " + e.getMessage());
        e.printStackTrace();
    }
}
```

โปรแกรมของเราจะทำการประมวลผลถ้าข้อมูลทั้งสองเป็นข้อมูลที่ถูกต้อง ดังนั้นเราจึงต้องทำการตรวจสอบว่า first และ divider มีค่าที่ถูกต้องหรือไม่ด้วยการตรวจสอบกับ -1 ซึ่งถ้าตรวจสอบแล้วได้ข้อมูลที่ถูกต้อง เราก็จะทำการประมวลผลข้อมูลทั้งสอง ลองมาดูผลลัพธ์กัน

```
Enter a number: 23
Enter a divider: 3
Result = 7.0
Enter a number: p
Exception caught: For input string: "p"
Enter a number: 3
Enter a divider: p
Exception caught: For input string: "p"
Enter a number: 32
Enter a divider: 0
```

```
Exception: / by zero
java.lang.ArithmeticException: / by zero
        at ThrowsWithTry1.main(ThrowsWithTry.java:34)
Enter a number: -1
Enter a number: Exception caught: null
Enter a number:
```

จะเห็นได้ว่าโปรแกรมของเราทำงานได้อย่างที่ได้ตั้งใจไว้ คือ ตรวจสอบข้อมูลทั้งสอง พร้อมทั้งทำการประมวลผลถ้าข้อมูลทั้งสองถูกต้อง ผู้อ่านควรสังเกตด้วยว่า โปรแกรมตัวอย่างนี้ยังไม่สมบรูณ์เลยทีเดียว ดังจะเห็นได้จากการใส่ข้อมูลที่เป็น -1 ทำไม? พร้อมกันนี้เรายังไม่สามารถออกจากโปรแกรมได้ ถ้าไม่ใช้คำสั่ง <crtl> + c

ในการเขียนโปรแกรมที่ต้องรองรับการทำงานที่มีการตรวจสอบเพื่อให้เกิดความถูกต้องนั้น เราต้องคำนึงถึงปัจจัยหลายด้าน ทั้งนี้ก็ขึ้นอยู่กับลักษณะของงานที่เราต้องเขียนโปรแกรมรองรับ โปรแกรมตัวอย่างของเราต้องการแสดงให้เห็นถึงการตรวจสอบและดักจับ error และแก้ไขในระดับหนึ่งเท่านั้น ดังนั้นผู้เขียนจึงไม่ได้แสดงถึงการแก้ไขทั้งหมด เพราะข้อกำหนดของงานยังไม่ชัดเจน

การสร้าง exception ขึ้นมาใช้เอง

Java ยอมให้เราสร้าง exception ขึ้นมาใช้กับโปรแกรมต่าง ๆ ที่เราคิดว่าน่าจะเพิ่มข้อมูลบางอย่างให้กับผู้ใช้ ถ้าหากมี error เกิดขึ้น เช่น การหารด้วยศูนย์ที่เราได้เขียนขึ้นมาก่อนหน้านี้ เราอาจต้องการบอกรายละเอียดให้มากกว่าเดิมแก่ผู้ใช้ เราจะใช้โปรแกรมการหารด้วยศูนย์ที่มีการเปลี่ยนแปลงบางอย่าง เป็นตัวอย่างต่อไป

```
//TestInputValidation.java

import java.io.*;
import java.lang.Integer;

class TestInputValidation {
    public static void main(String[] args) throws Exception {
        BufferedReader buffer;
        InputStreamReader isr;
        String input;
        int first, divider;

        while(true) {
            //get first number
            first = readInt();

            //get second number only if first number is valid
            divider = readInt();

            //perform division if both numbers are valid
            try {
            //calling method divide()
                int result = divide(first,divider);
                System.out.println("Result = " + result);
            }
            //catching user defined exception
            catch(ZeroDivideException e) {
                e.printStackTrace(System.err);
            }
        }
    }

    //method to get an int from the keyboard
```

```
    public static int readInt() throws Exception {
        BufferedReader buffer;
        InputStreamReader isr;
        String input = null;
        int result = 0;
        boolean ok = false;

        isr = new InputStreamReader(System.in);
        buffer = new BufferedReader(isr);
        //keep looping until user enter an int
        while(!ok) {
            System.out.print("Enter a number: ");
            try {
                input = buffer.readLine();
                result = Integer.parseInt(input);
                ok = true;
            }
            catch(NumberFormatException e) {
                System.out.println(e.getMessage());
                e.printStackTrace(System.err);
            }
        }
        return result;
    }

    //method to divide first number by the second number
    public static int divide(int first, int second)
                    throws ZeroDivideException {
        int result;
        try {
            result = first / second;
        }
        catch(ArithmeticException e) {
            throw new ZeroDivideException(); //throw new exception
        }
        return result;
    }

}
```

เราได้เปลี่ยนโปรแกรมก่อนหน้านี้มากพอสมควรในการที่จะอ่านข้อมูลให้ถูกต้อง พร้อมทั้งแสดงถึงการเขียน exception ขึ้นมาใช้เอง เราเริ่มด้วยการสร้าง method readInt() ขึ้นมาใหม่ โดยกำหนดให้ผู้ใช้ต้องใส่ข้อมูลให้ถูกต้อง ถ้าไม่ถูกเราก็จะวนกลับไปถามใหม่จนกว่าจะได้

Method ตัวที่สองที่เราสร้างขึ้นมาคือ method divide() ที่รับ parameter 2 ตัว ทำหน้าที่ในการหารตัวเลขตัวแรกด้วยตัวเลขตัวที่สอง ภายใน method divide() นี้เราจะดักจับ error ที่เกิดขึ้นผ่านทาง ArithmeticException ด้วย การ throw exception ที่เราได้สร้างขึ้นเอง ด้วยคำสั่ง

```
catch(ArithmeticException e) {
    throw new ZeroDivideException(); //throw new exception
}
```

แต่ก่อนที่เราจะเรียกใช้ exception ที่ว่านี้เราต้องสร้าง code สำหรับ exception เสียก่อน ซึ่งเราได้ออกแบบให้เป็นการดักจับแบบง่าย ๆ ด้วยการสร้าง class ZeroDivideException จาก class Exception เพื่อที่จะให้เราสามารถที่จะได้รับการถ่ายทอดคุณสมบัติจาก class Exception เรากำหนดให้ code ของ class ZeroDivideException มีดังนี้

```
//ZeroDivideException.java
```

```
class ZeroDivideException extends Exception {

    //default constructor
    ZeroDivideException() {
        super("divide by zero"); //call super class' constructor
    }

    ZeroDivideException(String message) {
        super(message);
    }
}
```

เราไม่ได้ทำอะไรมากไปกว่าเรียกใช้ constructor ของ class Exception ด้วยข้อความที่เราต้องการให้ปรากฏหาก error ที่ว่านี้เกิดขึ้น (divide by zero)

ภายในตัว method main() เอง code ส่วนที่สำคัญคือ

```
try {
    //calling method divide()
    int result = divide(first,divider);
    System.out.println("Result = " + result);
}
//catching user defined exception
catch(ZeroDivideException e) {
    e.printStackTrace(System.err);
}
```

เราเรียก method divide() ภายใน try และถ้าหากการหารที่เกิดขึ้นมี error มันก็จะถูกจับใน catch block ที่เราได้เรียกด้วย object ที่มาจาก class ZeroDivideException ที่เราได้สร้างขึ้น หลังจากที่ลอง run ดู เราได้ผลลัพธ์ดังนี้

```
Enter a number: 45
Enter a number: 36
Result = 1
Enter a number: 0
Enter a number: 3
Result = 0
Enter a number: pop
For input string: "pop"
java.lang.NumberFormatException: For input string: "pop"
        at
java.lang.NumberFormatException.forInputString(NumberFormatException.
java:48)
        at java.lang.Integer.parseInt(Integer.java:426)
        at java.lang.Integer.parseInt(Integer.java:476)
        at TestInputValidation.readInt(TestInputValidation.java:47)
        at TestInputValidation.main(TestInputValidation.java:16)
Enter a number: 3
Enter a number: 0
ZeroDivideException: divide by zero
        at TestInputValidation.divide(TestInputValidation.java:64)
        at TestInputValidation.main(TestInputValidation.java:23)

Enter a number: null (เราออกจากโปรแกรมด้วยการกด <crtl> + c ณ ตำแหน่งนี้)
java.lang.NumberFormatException: null
        at java.lang.Integer.parseInt(Integer.java:394)
        at java.lang.Integer.parseInt(Integer.java:476)
```

```
        at TestInputValidation.readInt(TestInputValidation.java:47)
        at TestInputValidation.main(TestInputValidation.java:16)
Enter a number:
```

ผลลัพธ์ที่ได้ก็เป็นไปตามที่เราได้คาดไว้ คือ error ถูกดักจับไว้ได้หมด ผู้อ่านควรสังเกตด้วยว่าเรายังใช้ loop ใน method main() ดังนั้นการออกจาก loop ก็ต้องใช้ <crtl> + c เหมือนเดิม ทำให้มีการดักจับ error ที่เกิดขึ้นครั้งสุดท้ายก่อนออกจากโปรแกรม

อีกส่วนหนึ่งที่น่าสนใจในโปรแกรมของเราก็คือ การส่งข้อความบอกถึงรายละเอียดของ error ผ่านทาง System.err ซึ่งเป็นอีกช่องทางหนึ่งที่ Java มีให้ใช้สำหรับการส่งข้อมูลที่ไม่ใช่ข้อมูลปกติออกทางหน้าจอ ถึงแม้ว่าข้อมูลจะออกไปที่เดียวกันแต่ช่องทางการส่งออก ไม่เหมือนกัน สาเหตุที่เป็นเช่นนี้ก็เพื่อให้มีการแยกแยะ error ออกจากข้อมูลหรือผลลัพธ์ปกติของโปรแกรม

เรามาดูอีกสักตัวอย่างหนึ่ง

```
//CheckRange.java

import java.lang.Integer;

class CheckRange {
    public static void main(String[] args) {
        try {
            //getting input from argument list
            int first = Integer.parseInt(args[0]);
            int second = Integer.parseInt(args[1]);
            int third = Integer.parseInt(args[2]);

            //check if the third input is between first and second
            checked(first, second, third);
            System.out.println("Number " + third +
                               " is in the given range");
        }
        catch(OutOfRangeException e) {
            e.printStackTrace(System.err);
        }
    }

    //method to validate range
    public static void checked(int start, int stop, int num) throws
                        OutOfRangeException {
        if(num < start) {
            String errMsg = new String(num + " < " + start);
            throw new OutOfRangeException(errMsg, start, stop);
        }
        if(num > stop) {
            String errMsg = new String(num + " > " + stop);
            throw new OutOfRangeException(errMsg, start, stop);
        }
    }
}
```

โปรแกรมตัวอย่างนี้รับข้อมูลผ่านทาง argument list ที่เป็นตัวเลขสามตัว โดยจะทำการตรวจสอบว่าตัวเลขตัวที่สามอยู่ระหว่างตัวแรกและตัวที่สองหรือไม่ ถ้าไม่ก็จะ throw exception OutOfRangeException ที่ได้เขียนขึ้น ดังนี้

```
//OutOfRangeException.java
```

```
public class OutOfRangeException extends Exception {
    //default constructor
    OutOfRangeException() {}

    //constructor
    OutOfRangeException(String message, int p, int q) {
        super(message);
    }
}
```

จะเห็นว่าโดยส่วนใหญ่แล้วเราจะเรียกใช้ super class constructor ให้ทำงานให้เรา มากกว่าที่จะเขียน code ขึ้นมา ซึ่งเป็นสิ่งที่ทำให้การออกแบบของเราไวขึ้น เราเพียงแต่เขียนถึงสิ่งที่บ่งบอกถึง error ที่เกิดขึ้นว่าเป็นอะไรเท่านั้นเอง จากการ run ผลลัพธ์ที่เราได้คือ

```
>java CheckRange 12 45 569
OutOfRangeException: 569 > 45
        at CheckRange.checked(CheckRange.java:31)
        at CheckRange.main(CheckRange.java:15);

E:\bc221Book\source>java CheckRange 12 45 2
OutOfRangeException: 2 < 12
        at CheckRange.checked(CheckRange.java:27)
        at CheckRange.main(CheckRange.java:15)

E:\bc221Book\source>java CheckRange 12 45 15
Number 15 is in the given range
```

สรุป

ในการเขียนโปรแกรมที่ดีนั้น ผู้เขียนจะต้องคำนึงถึง error ที่อาจเกิดขึ้นในโปรแกรม ซึ่งได้แบ่งออกเป็นสองลักษณะใหญ่ คือ error ที่โปรแกรมเมอร์ไม่มีทางแก้ไขได้ คือ อยู่นอกเหนือการควบคุมของผู้พัฒนาโปรแกรม เช่น error ที่เกิดจากหน่วยความจำไม่พอเพียง ส่วน error อีกอันหนึ่งที่ผู้เขียนโปรแกรมต้องตรวจสอบก็คือ error ที่เกิดตอน run โปรแกรม เช่น error จากการหารด้วยศูนย์ การเข้าหา Index ของ array ที่ไม่มีอยู่จริง การป้อนข้อมูลที่ไม่ถูกกับชนิดที่ต้องการ อย่างนี้เป็นต้น

Exception ที่ Java มีให้มีอยู่มากพอสมควร และมากพอที่จะเรียกใช้ได้เกือบทุกกรณีของการเขียนโปรแกรม แต่ถ้าผู้เขียนโปรแกรมได้พัฒนาโปรแกรมที่ไม่สามารถใช้ exception ที่ Java มีให้ก็สามารถที่จะออกแบบ exception ขึ้นมาใช้เองได้

ในการดัก error นั้นผู้เขียนอาจสร้าง exception ขึ้นมาใช้ในการดักจับ error เองได้ ทั้งนี้ก็ขึ้นอยู่กับลักษณะของงานที่ผู้เขียนโปรแกรมกำลังพัฒนาอยู่ โดยสรุปแล้วสิ่งที่ต้องควรคำนึงในการตรวจจับ error คือ

- ✓ Exception เป็นตัวบ่งชี้ถึง error ที่เกิดในโปรแกรม
- ✓ Exception เป็น object ที่เกิดมาจาก class Throwable
- ✓ เราสามารถที่จะเรียกใช้ exception ที่มาจาก Java โดยตรง หรือ สร้างขึ้นมาใช้เอง
- ✓ ถ้าเราต้องการที่จะควบคุม exception ใน method เราต้องเขียน code ใน try block และไม่จำเป็นที่จะต้องมี try block เพียงหนึ่ง block เท่านั้น อาจมีมากกว่าหนึ่ง block ได้
- ✓ Code ที่ใช้ในการควบคุม หรือ แก้ไข exception จะต้องทำใน catch block ที่ตามหลัง try block ทันทีและในหนึ่ง try block เราสามารถที่จะมี catch block มากกว่าหนึ่งตัวได้
- ✓ Finally เป็น block ท้ายสุดที่อยู่ใน try โดยทั่วไปจะใช้ finally block ในการเก็บกวาดการทำงานของโปรแกรม เช่น การปิดไฟล์

แบบฝึกหัด

1. จงเขียนโปรแกรมที่รับข้อมูลจาก keyboard เฉพาะที่เป็น int เท่านั้นให้ใช้ exception ในการดักจับ error ที่ผู้ใช้อาจใส่ผิด เช่น ใส่ตัวอักษร ให้แสดงข้อความฟ้องทันทีที่ผู้ใช้ใส่ผิด

2. จงปรับปรุงโปรแกรมที่เขียนขึ้นในข้อหนึ่ง โดยเพิ่มการตรวจสอบที่ไม่รับตัวเลขที่ต่ำกว่าศูนย์

3. จงเขียนโปรแกรมที่รับข้อมูลจาก keyboard เป็นตัวเลขสามตัว โดยกำหนดให้ ตัวแรกเป็นจุดเริ่มต้น ตัวที่สองเป็น จุดจบ ของค่าขององศา Celsius และตัวที่สามเป็นตัวกำหนดการเพิ่มค่า (step) ให้โปรแกรมทำการคำนวณหา องศา Fahrenheit ตามข้อมูลที่กำหนดไว้ในสองตัวแรก และให้ทำการเพิ่มขึ้นตามค่าของข้อมูลตัวที่สาม ดังตัวอย่างผลลัพธ์ที่แสดงให้ดูนี้

```
>java Temperature 0 100 10

CELSIUS FAHRENHEIT
0       32.0
10      50.0
20      68.0
30      86.0
40      104.0
50      122.0
60      140.0
70      158.0
80      176.0
90      194.0
100     212.0
```

ให้ทำการตรวจสอบ และดักจับ error ที่อาจเกิดขึ้นจากการใส่ข้อมูลที่ไม่ถูกต้อง เช่น ค่าของการเพิ่มเป็นตัวเลขที่น้อยกว่าศูนย์ (negative number) ค่าเริ่มต้น มากกว่าค่าสุดท้าย และจำนวนข้อมูลที่นำเข้ามีจำนวนไม่ครบตามที่กำหนด ตัวอย่างของ error ที่เกิดขึ้น

```
 >java Temperature 0 100 -5
3rd arg < 0 NUMBER = -5

 >java Temperature 40 20 5
 Ordering Err  FIRST = 40 SECOND = 20

 >java Temperature 10 40
 java.lang.ArrayIndexOutOfBoundsException:
 at Temperature.main(Temperature.java:41)
```

4. จงเขียนโปรแกรมที่หาค่าของ m^n โดยกำหนดให้ทั้ง m และ n มาจาก command line argument list ให้ทำการตรวจสอบและดักจับ error ทุก ๆ กรณีที่อาจทำให้โปรแกรมไม่ทำงาน

5. จงปรับปรุง class Account ในบทที่หกให้มีการตรวจสอบและดักจับ error ที่อาจเกิดขึ้นในการ ถอนเงิน ฝากเงิน และ คิดดอกเบี้ย ถ้า method ใด ๆ ที่ทำหน้าที่ดังกล่าวไม่มี ให้เขียนขึ้นใหม่ พร้อมทั้งเขียนโปรแกรมทดสอบ

6. จงปรับปรุงโปรแกรมในข้อ เจ็ด แปด และ เก้า ในบทที่ห้า ให้มีการตรวจสอบและดักจับ error พร้อมทั้งเขียนโปรแกรมทดสอบ

8
Streams I/O

ในบทที่แปดนี้เราจะพูดถึงการอ่าน และเขียนข้อมูลผ่านทาง stream หรือช่องทางการส่งข้อมูล เราจะดูถึงวิธีการ กระบวนการต่าง ๆ ที่เกี่ยวข้องกับข้อมูลนำเข้า และการนำข้อมูลออก

หลังจากจบบทนี้แล้ว ผู้อ่านจะได้ทราบถึง

- ความหมายของ stream
- Class ต่าง ๆ ที่ Java มีให้ในการประมวลผลด้วย stream
- การสร้าง directory
- การสร้าง file การเปิดและปิด file การอ่านและเขียน file
- ข้อแตกต่างระหว่าง text file และ binary file

ในการทำงานที่เกี่ยวข้องกับไฟล์ใน Java นั้นจะต้องทำผ่านทาง stream ซึ่งเป็นตัวแทนของอุปกรณ์ที่นำเข้า (input) หรือส่งออก (output) ข้อมูล เราสามารถเขียน และอ่านข้อมูลผ่านทาง stream ด้วยวิธีการต่าง ๆ ในรูปแบบต่าง ๆ กัน เช่น อ่านข้อมูลจาก keyboard ทีละหนึ่งตัวอักษร หรือทีละ 10 byte

โดยทั่วไป ข้อมูลนำเข้าจะมาจาก keyboard หน่วยความจำสำรอง (disk) หรือ อุปกรณ์อื่น ๆ เช่น scanner light-pen memory-card แต่โดยส่วนใหญ่แล้วมักจะมาจาก keyboard และ disk ในบทนี้เราจะใช้อุปกรณ์ทั้งสองที่ได้กล่าวมา เป็นช่องทางหลักในการนำเข้าข้อมูล และจะใช้ช่องทางส่งออกเพียงสองทางคือ จอ (monitor) และ เครื่องพิมพ์ (printer)

รูปแบบพื้นฐานของข้อมูลที่ Java รู้จักมีอยู่สองลักษณะคือ binary streams และ character streams ซึ่งมีความแตกต่างกันในรูปแบบของการจัดเก็บ ถ้าเราเขียนข้อมูลเข้าสู่ stream โดยเขียนทีละ byte หรือ ทีละหลาย ๆ byte เราเรียกข้อมูลเหล่านี้ว่า binary data การเขียนจะไม่มีกระบวนการอื่นใดมายุ่งเกี่ยว (เช่น การเปลี่ยนข้อมูลให้อยู่ในรูปแบบอื่นก่อนการจัดเก็บ) ข้อมูลจะถูกส่งเข้าสู่ stream ตามที่ปรากฏอยู่ใน memory เช่น ถ้า memory ส่วนนี้เก็บข้อมูลที่เป็นตัวเลขอยู่ ตัวเลขเหล่านี้จะถูกส่งเข้าสู่ streams ตามที่ปรากฏใน memory

ในการเก็บข้อมูลที่เป็น character streams นั้นโดยทั่วไปจะทำกับข้อมูลที่เป็นตัวอักษร (text) เราจะใช้ character streams ในการอ่านไฟล์ที่เป็น text ตัวเลขที่อยู่ในไฟล์จะถูกเปลี่ยนให้อยู่ในรูปแบบที่ Java ได้กำหนดขึ้น การอ่านตัวเลขด้วยการใช้ character streams เป็นกระบวนการที่ยุ่งยากมาก เพราะเราต้องรู้ว่าตัวอักษรที่เป็นตัวแทนของตัวเลขนั้นใช้กี่ตัวอักษรในการเก็บ เช่น ถ้าเราส่งตัวเลข 17 ผ่านทาง character stream ตัวเลขนี้จะถูกเปลี่ยนให้เป็นค่า ASCII ของ '1' และ '7' หรือ 0x310x37 (เลขฐาน 16) และสมมติว่าเราเขียนเลข 727 อีกตัวใน stream ของเราก็จะมีข้อมูลเป็น 0x310x370x370x320x37 การอ่านตัวเลขทั้งสองตัวออกมาจะทำได้ยากมากเพราะเราไม่รู้ว่าจะต้องอ่านตัวอักษรทีละกี่ตัวถึงจะได้ข้อมูลที่ถูกต้องเหมือกับที่ได้เขียนไว้

เราได้ทราบแล้วว่า Java เก็บข้อมูลในรูปแบบของ Unicode ดังนั้นในการเขียนข้อมูลเข้าสู่ stream ในรูปแบบของ character ข้อมูลจะถูกเปลี่ยน (โดยอัตโนมัติ) ให้อยู่ในรูปแบบของการเก็บ (local code) ตามที่เครื่องนั้นใช้อยู่ เช่นถ้าเครื่องของเราใช้ภาษาไทยเป็นภาษาหลัก รูปแบบของการเก็บก็จะเป็น code ของภาษาไทย (Java เรียกระบบนี้ว่า localization) ในการอ่านข้อมูลนั้น Java จะเปลี่ยนข้อมูลที่ถูกจัดเก็บในรูปแบบของ local code ให้เป็น Unicode ก่อน โดยสรุปแล้วในการเขียนและอ่านนั้น Java จะเปลี่ยนข้อมูลให้เป็น Unicode ส่วนการจัดเก็บนั้นจะจัดเก็บในรูปแบบของภาษาที่ใช้อยู่ในเครื่อง

เราจะมาดูตัวอย่างที่เกี่ยวข้องกับไฟล์ ในหลาย ๆ รูปแบบเพื่อทำให้เกิดความเข้าใจถึงวิธีการ และกระบวนการต่าง ๆ ที่เกี่ยวข้องกับ input และ output เราจะเริ่มต้นด้วยการตรวจสอบข้อมูลต่าง ๆ ที่เกี่ยวกับไฟล์ จากโปรแกรม FileInfo.java

```
//FileInfo.java

import java.io.*;

class FileInfo {
    public static void main(String[] args) {
        //get path from command-line argument
        File path = new File(args[0]);

        //display some info. about this file
        if(path.exists()) {
            System.out.println(path + " does exist.");
            System.out.println("Readable   : " +
                               statusOk(path.canRead()));
            System.out.println("Writable   : " +
                               statusOk(path.canWrite()));
            System.out.println("Directory? : " +
                               statusOk(path.isDirectory()));
            System.out.println("File?      : " +
                               statusOk(path.isFile()));
            System.out.println("Hidden?    : " +
                               statusOk(path.isHidden()));
        }
        else {
            System.out.println(path + " does not exist.");
        }
    }

    //return "Yes" or "No"
    public static String statusOk(boolean yes) {
        return yes ? "Yes" : "No";
    }
}
```

โปรแกรมเริ่มต้นด้วยการสร้าง object path จาก class File ด้วยข้อมูลที่มาจาก command-line argument ซึ่งอยู่ในรูปแบบ

```
e:\bc221Book\source
e:\bc221Book\source
\source
\bc221Book\source\FileInfo.java
```

ตัวอย่างสองตัวแรกอยู่ในรูปแบบที่เรียกว่า absolute path ส่วนสองตัวสุดท้ายอยู่ในรูปแบที่เรียกว่า relative path

absolute path เป็นการกำหนดที่มาที่ไปด้วยตัวอักษรของ drive ส่วน relative path จะไม่มีการกำหนดตัวอักษรของ drive แต่การทำงานทั้งหมดจะอยู่ภายใน directory นั้น ๆ ดังตัวอย่างผลลัพธ์ของการ run นี้

```
>java FileInfo \bc221Book
\bc221Book does exist.
Readable      : Yes
Writable      : Yes
```

```
Directory?     : Yes
File?          : No
Hidden?        : No

>java FileInfo \bc221Book\source\FileInfo.java
\bc221Book\source\FileInfo.java does exist.
Readable       : Yes
Writable       : Yes
Directory?     : No
File?          : Yes
Hidden?        : No

>java FileInfo \bc221Book\FileInfo.java
\bc221Book\FileInfo.java does not exist.
```

ผลลัพธ์ของการ run ทั้งสามครั้งใช้ relative path เป็นตัวกำหนดการค้นหา directory และ file ส่วนผลลัพธ์ด้านล่างนี้เป็นการาค้นหาจาก absolute path

```
E:\bc221Book\source>java FileInfo e:\bc221Book\source\FileInfo.java
e:\bc221Book\source\FileInfo.java does exist.
Readable       : Yes
Writable       : Yes
Directory?     : No
File?          : Yes
Hidden?        : No
```

เราเรียกใช้ method ที่มีอยู่ใน class File ในการตรวจสอบข้อมูลของ file ว่าเป็น file ลักษณะไหน ข้อมูลเฉพาะคือ อะไร คำว่า file ใน Java นั้นเป็นได้สองอย่าง คือ file ที่เก็บข้อมูลที่เราใช้ (หรือ ระบบใช้) และ file ที่เก็บ file หรือที่รียกว่า directory (หรือ folder)

Java ยังมี method ที่ใช้ในการตรวจสอบข้อมูลของ file อีกหลายตัว เราจะลองใช้ method เหล่านี้ในโปรแกรมตัวอย่างต่อไปนี้

```
//FileInfo1.java

import java.io.*;
import java.util.Date;

class FileInfo1 {
    public static void main(String[] args) {
        //explain how to use if no argument is given
        if(args.length < 1) {
            System.out.println("Usage: FileInfo1 file-name");
            System.exit(1);
        }
        //display information about this file
        echoFileInfo(new File(args[0]));
    }

    private static void echoFileInfo(File file) {
        //display if file is a file or a directory
        if(file.isFile())
            System.out.println("\n" + file + " is a file.");
        if(file.isDirectory()) {
            String []list = file.list();
            System.out.println("\n" + file + " is a directory.");
            System.out.println("There are " + list.length +
                               " items in this directory.");
```

```
        }
        //lastModifeid() returns amount of milliseconds the file
        //was last modified, e.g. 1045375241775 since
        //January 1, 1970, 00:00:00 GMT. We have to create a date
        //object from this data.
        long lastModified = file.lastModified();
        Date dateModified = new Date(lastModified);
        //display information about this file
        System.out.println(
            "Absolute path: " + file.getAbsolutePath() +
            "\n Name: " + file.getName() +
            "\n Parent: " + file.getParent() +
            "\n Path: " + file.getPath() +
            "\n Length: " + file.length() +
            "\n Last modified: " + dateModified);
    }
}
```

โปรแกรม FileInfo1.java เรียกใช้ method 5 ตัวในการแสดงข้อมูลของไฟล์ที่กำหนดให้ (ผ่านทาง command-line argument) ซึ่งเป็นการเรียกใช้โดยตรง และไม่มีความยุ่งยากในการเรียกใช้ มี method เพียงตัวเดียวเท่านั้นที่เราต้องปรับปรุงผลลัพธ์ที่ method ตัวนี้ส่งมาให้ method ที่ว่านี้คือ lastModified()

lastModified() จะส่งค่าที่มีชนิดเป็น long ที่เป็นค่าของ วันและเวลาที่ไฟล์ได้รับการเปลี่ยนแปลงครั้งสุดท้ายในรูปแบบของ millisecond เช่น 1045375241775 ซึ่งเราไม่สามารถบอกได้ว่าเป็นวันที่เท่าไร เวลาอะไร เราต้องใช้ค่านี้ในการสร้าง วันขึ้นใหม่ผ่านทาง class Date ดังนี้

```
long lastModified = file.lastModified();
Date dateModified = new Date(lastModified);
```

หลังจากนั้นเราก็แสดงผลออกทางหน้าจอ ดังที่แสดงให้ดูเป็นตัวอย่างด้านล่างนี้

```
>java FileInfo1 e:\bc221Book\source

e:\bc221Book\source is a directory.
There are 328 items in this directory.
Absolute path: e:\bc221Book\source
 Name: source
 Parent: e:\bc221Book
 Path: e:\bc221Book\source
 Length: 0
 Last modified: Wed Feb 26 07:47:48 GMT+07:00 2003

>java FileInfo1 FileInfo1.java

FileInfo1.java is a file.
Absolute path: E:\b221Book\source\FileInfo1.java
 Name: FileInfo1.java
 Parent: null
 Path: FileInfo1.java
 Length: 1236
 Last modified: Tue Feb 25 14:06:13 GMT+07:00 2003

>java FileInfo1
Usage: FileInfo1 file-name
```

ตัวอย่างต่อไปเป็นโปรแกรมที่แสดงรายละเอียดของ directory ว่าประกอบไปด้วยไฟล์ หรือ sub-directory อะไรบ้าง

```
//DirListing.java
//adopted from Bruce Eckel's

import java.io.*;
import java.util.Arrays;

//extract file name from path
//must implements FilenameFilter and override method accept()
//to get files with given information into a list
class DirFilter implements FilenameFilter {
    String fileName;

    //constructor
    DirFilter(String fileName) {
        this.fileName = fileName;
    }

    //this method is called by method list() from class File
    //to include a file if it contains a given info
    public boolean accept(File path, String name) {
        //get only file name
        String file = new File(name).getName();
        //if in fact the file exists we will get the first index
        //of this file, which causes accept() to return true,
        //otherwise we will get -1 and returns false
        return file.indexOf(fileName) != -1;
    }
}

class DirListing {
    public static void main(String[] args) {
        //create file object with a default directory
        File path = new File(".");
        String[] fileList;    //list of files with given criteria

        //no argument provided, get all files on this directory
        if(args.length == 0)
            fileList = path.list();
        //information provided, get rid of path and extract files
        //with given info.
        else
            fileList = path.list(new DirFilter(args[0]));

        Arrays.sort(fileList);    //sort the list
        showDir(fileList);        //display the list
    }

    //display files to screen
    public static void showDir(String[] list) {
        for(int i = 0; i < list.length; i++)
            System.out.println(list[i]);
    }
}
```

ก่อนที่จะอธิบายถึงการทำงานของโปรแกรม เรามาดูผลลัพธ์ (ตัดทอนบางส่วนออก) กันก่อน

```
>java DirListing java
```

```
Access.java
Account.java
Add1To100.java
    …
    …
AddNumbers2.java
ThrowsWithTry.java
ThrowsWithTry1.java
Triangle.java
TryAndCatchExample1.java
TryAndCatchExample2.java
Two.java
UpperLower.java
UpperLower2.java
Variables.java
Vowels.java
ZeroDivideException.java
p.java

>java DirListing te
BankRate.class
BankRate.java
ByteShort.class
ByteShort.java
Calculate.class
Calculate.java
```

โปรแกรม DirListing.java จะแสดงรายการของข้อมูลที่อยู่ใน directory นั้น ๆ ตามข้อกำหนดที่เราตั้งให้ เช่นถ้าเราต้องการแสดงไฟล์ทั้งหมดที่มีอยู่เราก็ใช้คำสั่ง DirListing โดยไม่มีการกำหนดใด ๆ แต่ถ้าต้องการแสดงผลด้วยข้อกำหนด เช่น ต้องการแสดงไฟล์ทุกตัวที่มีคำว่า "te" อยู่ในไฟล์เราก็ใช้คำสั่ง DirListing te เป็นต้น

โปรแกรมของเราต้องเรียกใช้ interface FileNameFilter เพื่อทำการ override method accept() ที่ทำหน้าที่ในการกรองข้อมูลสำหรับให้ method list() ที่อยู่ใน class File ใช้ โดยกำหนดให้ method accept() ตรวจสอบไฟล์จาก path เฉพาะไฟล์ที่ถูกต้องตามเงื่อนไขที่กำหนดไว้แล้วจึงบอก method list() ว่าไฟล์ตัวนี้ควรเก็บไว้ใน list ประโยคที่แสดงผลโดยไม่ต้องกำหนดเงื่อนไข ต้องตรวจสอบกับ args.length ถ้าค่านี้เป็น 0 เราจะเก็บไฟล์ทุกตัวไว้ใน array list ด้วยการเรียกใช้

```
fileList = path.list();
```

แต่ถ้ามีการกำหนดเงื่อนไข เราจะส่งเงื่อนไขนี้ไปให้ method accept() เพื่อทำการคัดเลือกไฟล์ที่ถูกต้องตามเงื่อนไข ด้วยประโยค

```
fileList = path.list(new DirFilter(args[0]));
```

สมมติว่า args[0] ของเรามีค่าเป็น "te" (ดังตัวอย่าง) "te" ก็จะถูกเก็บไว้เป็นตัวตรวจสอบว่าไฟล์ที่อยู่ใน path มีคำนี้อยู่หรือไม่ เช่นไฟล์ชื่อ Calculate.java

Method list() กำหนดให้ parameter ที่ส่งเข้าไปเป็น object จาก class FileNameFilter ดังนั้นเราสามารถที่จะเขียน method accept() ในลักษณะใดก็ได้ ในที่นี้เราดึงเอาเฉพาะชื่อไฟล์ที่ไม่มี path รวมอยู่ด้วย ด้วยการกำหนดดังนี้

```
String file = new File(name).getName();
```

หลังจากนั้นเราใช้ method indexOf() จาก class String ในการตรวจสอบว่า "te" มีอยู่ในไฟล์นี้หรือไม่ ซึ่งถ้ามีค่าที่ได้จาก indexOf() จะเป็นตำแหน่งที่ "te" อยู่ แต่ถ้าไม่มีได้ค่า -1 เรานำค่านี้มาเปรียบเทียบกับ -1 เพื่อบอกให้ list() รู้ว่าควรเก็บไฟล์ตัวนี้หรือไม่ (true = เก็บ false = ไม่เก็บ)

```
return file.indexOf(fileName) != -1;
```

ไฟล์ทุกตัวที่เราได้จาก method list() จะถูกเก็บไว้ในตัวแปร fileList เพื่อเอาไว้ใช้ในการแสดงผลต่อไป แต่ก่อนที่เราจะแสดงผล เราทำการเรียงลำดับของไฟล์ใน fileList ด้วยการเรียกใช้ method sort() ของ class Arrays

โปรแกรมตัวนี้อาจดูเข้าใจยากสักหน่อย แต่ก็ไม่ยากเกินไปนัก ผู้อ่านต้องทำความเข้าใจในเรื่องของการเรียกใช้ method list() ที่มาจาก class File รวมไปถึงการเขียน code ตามข้อกำหนดที่เราต้องการใน method accept() สิ่งที่เราต้องคำนึงถึง คือ

Method list() ใน class File มีอยู่สองตัว คือ ตัวที่ไม่มี parameter และตัวที่มี parameter สำหรับ method ตัวที่มี parameter นี้ parameter ต้องเป็น object จาก class FileNameFilter ดังนั้นเราต้องสร้าง class ที่มาจาก class นี้พร้อมทั้งทำการ override method accept() ให้ทำงานตามที่เราต้องการ ผู้อ่านควรทดลอง run โปรแกรมตัวนี้ด้วยเงื่อนไขต่าง ๆ เพื่อให้เกิดความเข้าใจมากยิ่งขึ้น และควรทดลองเขียน code ให้ method accept() ใหม่โดยกำหนดให้ทำการต่าง ๆ ที่ต่างออกไปจากที่เขียนในโปรแกรมตัวอย่างนี้

เราได้พูดถึงการแสดงรายการของไฟล์ที่มีอยู่ใน directory การสร้างเงื่อนไขให้กับ list() และ accept() พอสมควรแล้ว ต่อไปเราจะพูดถึงการอ่าน การเขียน ไฟล์ โดยเราจะเริ่มต้นที่ Input streams

Input and Output streams (ช่องทางการนำข้อมูลเข้า และ การนำข้อมูลออก)

Java กำหนดให้มีการนำข้อมูลเข้าสู่โปรแกรมได้หลายทาง ซึ่งเราก็ได้สัมผัสถึงวิธีการมาบ้างพอสมควรในเรื่องของการนำข้อมูลเข้าผ่านทางช่องทางการนำเข้ามาตรฐาน (standard I/O) ต่อไปนี้เราจะมาดูถึงการนำข้อมูลที่มาจากไฟล์เข้าสู่โปรแกรม

รูปแบบของข้อมูลที่ถูกจัดเก็บในไฟล์นั้นถูกแบ่งออกเป็นสองลักษณะคือ

1. ข้อมูลที่อยู่ในรูปแบบของ character เช่น code ของโปรแกรมที่สร้างขึ้นจาก Text editor พูดง่าย ๆ ก็คือ เราสามารถที่จะอ่านข้อมูลเหล่านี้ได้ เราเรียกไฟล์แบบนี้ว่า text file

2. ข้อมูลที่ถูกจัดเก็บในรูปแบบของ binary data เช่นข้อมูลที่ถูกเก็บในรูปแบบของ MS Word เราไม่สามรถที่จะอ่านข้อมูลเหล่านี้ได้ต้องอาศัยโปรแกรมในการช่วยอ่าน เราเรียกไฟล์ในรูปแบบนี้ว่า binary file

เราจะเริ่มด้วยการอ่านข้อมูลจาก text file ที่เก็บ code ของโปรแกรมที่เราได้เขียนขึ้น

```
//ReadingTextFile.java

import java.io.*;

class ReadingTextFile {
    //don't want to deal with error, let Java takes care of it
    public static void main(String[] args) throws IOException {
        BufferedReader in;//input buffer
        FileReader file;  //input file
        String str;       //string to store characters read from file

        file = new FileReader(args[0]); //open an input file
        in = new BufferedReader(file);  //store contents in buffer

        int lineNo = 1;
```

```
        //keep reading from buffer until null is reached
        while((str = in.readLine()) != null) {
            System.out.println(lineNo + "  " + str);
            lineNo++;
        }
        in.close();    //manually close the file
    }
}
```

เราเปิดไฟล์ที่ต้องการอ่านด้วย FileReader() พร้อมกับการเรียกใช้ BufferedReader() เพื่อให้การอ่านข้อมูลทำได้รวดเร็วขึ้น เราอ่านข้อมูลเข้ามาเก็บไว้ใน str ทีละหนึ่งแถวจนกว่าข้อมูลหมดจาก buffer หลังจากนั้นเราก็ส่งข้อมูลที่อ่านได้ออกไปยังหน้าจอพร้อมทั้งเลขที่บรรทัด เมื่อส่งข้อมูลหมดแล้วเราก็ปิดไฟล์ด้วยการใช้ close() ผลลัพธ์ที่เราได้จากการส่งไฟล์ ReadingTextFile.java ไปให้โปรแกรมนี้ คือ

```
1  //ReadingTextFile.java
2
3  import java.io.*;
4
5  class ReadingTextFile {
6       //don't want to deal with error, let Java takes care of it
7       public static void main(String[] args) throws IOException {
8               BufferedReader in; //input buffer
9               FileReader file;   //input file
10              String str;        //string to store characters read
from file
11
12              file = new FileReader(args[0]); //open an input file
13              in = new BufferedReader(file);  //store contents in
buffer
14
15              int lineNo = 1;
16              //keep reading from buffer until null is reached
17              while((str = in.readLine()) != null) {
18                      System.out.println(lineNo + "  " + str);
19                      lineNo++;
20              }
21              in.close();     //manually close the file
22      }
23  }
```

เรามาดูตัวอย่างที่อ่านข้อมูลด้วยการใช้ FileInputStream() และ read() ในการอ่านกันบ้าง

```
//ReadTextFile.java

import java.io.*;

class ReadTextFile {
    public static void main(String[] args) throws IOException {
        FileInputStream in;   //file input stream
        int ch;               //store each character read

        try {
            in = new FileInputStream(args[0]);
            while((ch = in.read()) != -1) {
                System.out.print((char)ch);
            }
            in.close();
        }
```

```
        //file not found
        catch(FileNotFoundException e) {
            System.err.println("Cannot find: " + args[0]);
            System.exit(1);
        }
        //no argument specified
        catch(ArrayIndexOutOfBoundsException e) {
            System.err.println("Usage: ReadTextFile file-name");
            System.exit(1);
        }
    }
}
```

โปรแกรมตัวนี้ใช้ FileInputStream() เป็นตัวเปิดไฟล์ และใช้ read() ในการอ่านข้อมูลทีละ byte จาก FileInputStream ทำการจัดเก็บข้อมูลที่อ่านได้ไว้ใน ch พร้อมทั้งส่งข้อมูลที่อ่านได้นี้ออกไปทางหน้าจอ จนกว่าการอ่านจะสิ้นสุดลง โดยอ่านเจอ -1 หรือ EOF (End Of File)

ในการส่งข้อมูลที่อ่านได้ออกไปยังหน้าจอนั้น เราต้องทำการ cast ให้กับ ch ก่อนมิเช่นนั้นแล้วข้อมูลที่ส่งออกไปจะไม่ตรงกับที่อ่านเข้ามา โปรแกรมของเรายังได้ทำการตรวจจับ error ที่อาจเกิดขึ้นสองตัว คือ โปรแกรมหาไฟล์ไม่เจอ และ ผู้ใช้ไม่กำหนดชื่อไฟล์ที่ใช้เป็นข้อมูลสำหรับการเปิดอ่าน

หลังจากทดลอง run ด้วยไฟล์ ReadTextFile.java ผลลัพธ์ที่ได้คือ

```
//ReadTextFile.java

import java.io.*;

class ReadTextFile {
      public static void main(String[] args) throws IOException {
          FileInputStream in;     //file input stream
          int ch;                 //store each character read

          try {
              in = new FileInputStream(args[0]);
              while((ch = in.read()) != -1) {
                    System.out.print((char)ch);
              }
              in.close();
          }
          //file not found
          catch(FileNotFoundException e) {
              System.err.println("Cannot find: " + args[0]);
              System.exit(1);
          }
          //no argument specified
          catch(ArrayIndexOutOfBoundsException e) {
              System.err.println("Usage: ReadTextFile file-name");
              System.exit(1);
          }
      }
}
```

character streams โดยทั่วไปเหมาะสมกับการอ่าน และเขียนข้อมูลที่เป็น text ดังนั้นหากจะเอามาใช้กับข้อมูลอื่น ๆ ก็จะสร้างความยุ่งยากในการอ่านและเขียนมากพอสมควร ดังนั้นในการอ่านและเขียนข้อมูลที่ไม่ใช่ text เราจึงต้องใช้การอ่านและเขียนข้อมูลที่อยู่ในรูปแบบของ binary data แต่ก่อนที่จะพูดถึงเรื่องของ Binary file เราจะมาดูกันถึงความยุ่งยากที่ว่า ในการทำงานกับ text file

โปรแกรม TextWithStreamTok.java อ่านข้อมูลที่ถูกเก็บไว้ในรูปแบบของ text นำมาประมวลผล เสร็จแล้วจึงส่งข้อมูลกลับไปยังหน้าจอ เพื่อให้ผู้อ่านเข้าใจง่ายขึ้น เราจึงจัดเก็บข้อมูลอยู่ใน format

```
String     double int
```

เช่น

```
Notebook       12.00  30
CoffeeMug      5.00   40
PaperCutter    12.50  25
```

และเราจะกำหนดให้มีข้อมูลเพียง 3 ตัวเท่านั้น ลองมาดูการใช้ StreamTokenizer ที่เราได้พูดถึงตอนที่เราอ่านข้อมูลนำเข้าจาก keyboard ก่อนหน้านี้ ว่าจะนำมาใช้กับข้อมูลที่อยู่ในไฟล์ได้อย่างไร ในโปรแกรมตัวอย่างด้านล่างนี้

```
//TextWithStreamTok.java - Reading text data from a file

import java.io.*;

class TextWithStreamTok {
    public static void main(String[] args) throws IOException {
        BufferedReader in = null;    //input buffer
        FileReader file = null;      //input file
        StreamTokenizer stream = null;   //tokens

        try {
            file = new FileReader(args[0]); //get a file from
argument-list
            in = new BufferedReader(file);   //storage buffer
            //create tokens
            stream = new StreamTokenizer(in);
            stream.eolIsSignificant(true);   //EOL counted but ignored
        }
        catch(FileNotFoundException e) {
            System.err.println("Cannot find file: " + args[0]);
            System.exit(1);
        }

        //arrays to store data only 3 items as example e.g.
        //description prices unit
        //Coffee-Mug  12.5 30
        String[] descs = new String[3];
        double[] prices = new double[3];
        int[] units = new int[3];

        int i = 0, j = 0, k = 0; //arrays' indices
        boolean first = true;    //make sure correct data is read
        try {
            //reading tokens into corresponding arrays
            //until EOF is reached
            while(stream.nextToken() != StreamTokenizer.TT_EOF) {
                //data is a string
                if(stream.ttype == StreamTokenizer.TT_WORD) {
                    descs[i++] = stream.sval;
                }
                //data is a number (price or unit)
                if(stream.ttype == StreamTokenizer.TT_NUMBER) {
                    //first data in line is price
```

```
                    if(first == true) {
                        prices[j++] = stream.nval;
                        first = false;
                    }
                    //next one is unit price
                    else {
                        units[k++] = (int)stream.nval;
                        first = true;
                    }
                }
                //ignore EOL
                if(stream.ttype == StreamTokenizer.TT_EOL) {
                    /* do nothing */
                }
            }
            in.close();    //close the stream
        }
        //trouble reading file
        catch(IOException e) {
            System.err.println("Error reading file.");
            e.printStackTrace();
            System.exit(1);
        }
        //too many items for arrays
        catch(ArrayIndexOutOfBoundsException e) {
            System.err.println("Array index out of bound.");
            e.printStackTrace();
            System.exit(1);
        }

        display(descs, prices, units);   //display data
    }

    //display data to screen
    private static void display(String[] d, double[] p, int[] u) {
        double total = 0.00;
        for(int i = 0; i < d.length; i++) {
            total = p[i] * u[i];
            System.out.print(d[i] + "\t" + p[i]);
            System.out.println("\t" + u[i] + "\t" + total);
        }
    }
}
```

การทำงานหลัก ๆ ของโปรแกรมก็คือ การอ่านข้อมูลที่ถูกจัดเก็บใน format ที่ได้กล่าวไว้ สิ่งสำคัญที่เราต้องคำนึงถึงคือ ตำแหน่งและชนิดของข้อมูลที่เราต้องอ่าน ก่อนอื่นเราต้องกำหนดช่องทางนำเข้าข้อมูลดังนี้

```
file = new FileReader(args[0]); //get a file from argument-list
in = new BufferedReader(file);  //storage buffer
//create tokens
stream = new StreamTokenizer(in);
stream.eolIsSignificant(true);  //EOL counted but ignored
```

หลังจากนั้นเราก็ตรวจสอบ token ที่เราอ่านจาก stream ว่าเป็นตัวเลขหรือว่าเป็น string

```
while(stream.nextToken() != StreamTokenizer.TT_EOF) {
    //data is a string
```

```
    if(stream.ttype == StreamTokenizer.TT_WORD) {
        descs[i++] = stream.sval;
    }
    //data is a number (price or unit)
    if(stream.ttype == StreamTokenizer.TT_NUMBER) {
        //first data in line is price
        if(first == true) {
            prices[j++] = stream.nval;
            first = false;
        }
        //next one is unit price
        else {
            units[k++] = (int)stream.nval;
            first = true;
        }
    }
    //ignore EOL
    if(stream.ttype == StreamTokenizer.TT_EOL) {
        /* do nothing */
    }
}
```

เราจะเก็บข้อมูลที่เป็น string ไว้ใน descs[] ข้อมูลที่เป็น double ไว้ใน prices[] และข้อมูลที่เป็น int ไว้ใน units[] สิ่งที่เราต้องคำนึงก็คือ ลำดับของข้อมูลที่เป็นตัวเลข เรารู้ว่าตัวเลขกลุ่มแรกที่อ่านได้คือ prices และข้อมูลถัดไปคือ units ดังนั้นเราต้องใช้ตัวแปร first เป็นตัวกำหนดที่เก็บข้อมูลว่า ที่อ่านได้ครั้งแรกควรไปอยู่ที่ prices[] และที่อ่านได้ถัดมาควรไปอยู่ที่ units[] และเมื่อเราอ่านจนหมดแล้วเราก็ส่ง arrays ทั้งสามตัวไปให้ display() ทำการประมวลผล และแสดงผลต่อไป

ผลลัพธ์ที่เราได้ จากการ run คือ

```
>java TextWithStreamTok data.dat
Notebook       12.0    30  360.0
CoffeeMug      5.0     40  200.0
PaperCutter    12.5    25  312.5
```

ข้อมูลที่เก็บไว้ในไฟล์ data.dat จะต้องมีช่องว่างระหว่างข้อมูลแต่ละตัว เช่น

```
Notebook 12.0 30
CoffeeMug 5.0 40
PaperCutter 12.5 25
```

หรือจะให้อยู่ในบรรทัดเดียวกันก็ได้ เช่น

```
Notebook 12.0 30 CoffeeMug 5.0 40 PaperCutter 12.5 25
```

ทั้งนี้เพราะ nextToken() จะอ่านข้อมูลที่อยู่ถัดจากช่องว่างไปจนกว่าจะเจอช่องว่างอีกครั้งหนึ่ง

จะเห็นได้ว่าเราต้องคอยระวังการอ่านข้อมูล เพราะถ้าลำดับของการอ่านข้อมูลผิด โปรแกรมของเราก็จะเกิดความผิดพลาด และที่สำคัญอีกอย่างหนึ่งคือ ขั้นตอนที่ยุ่งยากของการกำหนดลำดับของข้อมูล การอ่านข้อมูล (ลองนึกถึงการอ่าน ถ้าเรามีข้อมูลที่เป็นตัวเลขอยู่มากกว่าสองตัว และมีข้อมูลที่เป็น string อยู่ระหว่างข้อมูลเหล่านี้) ดังนั้นเราจึงไม่นิยมใช้ text file ในการเก็บข้อมูลที่เป็น primitive type เราควรใช้ text file ในการทำงานกับข้อมูลที่เป็น text เท่านั้น

ตัวอย่างต่อไปเป็นการใช้ StreamTokenizer นับจำนวนของคำที่มีอยู่ในไฟล์ เพื่อที่จะแสดงให้เห็นว่าถ้าข้อมูลเป็น text ทั้งหมดการทำงานจะไม่มีปัญหาเท่าไร

```
//WordCount.java - Using StreamTokenizer to count word
```

```
import java.io.*;

class WordCount {
    public static void main(String[] args) throws IOException {
        //exit if there's no file specified
        if(args.length < 1) {
            System.out.println("Usage: WordCount file-name.");
            System.exit(1);
        }
        FileReader inFile = null;    //file stream
        StreamTokenizer stream = null;  //tokens stream
        int count = 0;           //number of words in the file

        try {
            inFile = new FileReader(args[0]);
            stream = new StreamTokenizer(inFile);
            count = wordCount(stream);
            System.out.println(count + " words counted in " +
                               args[0]);
        }
        catch(FileNotFoundException e) {
            System.err.println("Cannot find: " + args[0]);
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            //close the file
            if(inFile != null) {
                try {
                    inFile.close();
                }
                catch(IOException e) { /* do nothing */ }
            }
        }
    }

    //method to count word in the stream
    private static int wordCount(StreamTokenizer s)
                    throws IOException {
        int count = 0;
        try {
            while(s.nextToken() != StreamTokenizer.TT_EOF) {
                //count only word
                if(s.ttype == StreamTokenizer.TT_WORD)
                    count++;
            }
        }
        catch(IOException e) {
            System.err.println("Error reading stream: " + s);
            e.printStackTrace();
            System.exit(1);
        }
        return count;
    }
}
```

โปรแกรม WordCount.java ทำการนับจำนวนของคำที่ปรากฏอยู่ในไฟล์ที่มาจาก command-line argument โดยเรียกใช้ StreamTokenizer เป็นตัวตรวจสอบว่า ข้อมูลที่มีอยู่ในไฟล์เป็น string หรือไม่ ถ้าเป็นก็จะนับ โดยไม่สนใจถึงรูปแบบของ string นั้น ๆ เช่น "s13" ก็นับเป็น string โปรแกรม WordCount.java ของเราตัวนี้ใช้ object จาก FileReader เป็นตัวกำหนดช่องทางผ่านไปยัง StreamTokenizer โดยตรง ซึ่งต่างจากตัวอย่างก่อนหน้านี้ ที่เราใช้ BufferedReader เป็นที่เก็บข้อมูล การใช้ FileReader โดยตรงจะทำให้เสียเวลาในการประมวลผลมากถ้าไฟล์มีขนาดใหญ่ เพราะฉะนั้นถ้าผู้อ่านต้องการความเร็วในการประมวลผล ก็ต้องใช้ BufferedReader เป็นตัวเก็บข้อมูลที่ต้องการประมวลผล

เราได้ทดลอง run โปรแกรมด้วยไฟล์ของ Windows Xp → vbe6.dll ซึ่งมีขนาดเท่ากับ 2,498,560 bytes ผลลัพธ์ที่ได้ก่อนและหลังการใช้ BufferedReader คือ

```
>java WordCount vbe6.dll
Start: Fri Feb 28 10:08:52 GMT+07:00 2003
279829 words counted in vbe6.dll
Stop: Fri Feb 28 10:08:54 GMT+07:00 2003

>java WordCount vbe6.dll
Start: Fri Feb 28 10:10:35 GMT+07:00 2003
279829 words counted in vbe6.dll
Stop: Fri Feb 28 10:10:36 GMT+07:00 2003
```

จะเห็นว่าการประมวลผลด้วยการใช้ BufferedReader ใช้เวลาน้อยกว่าการใช้ FileReader โดยตรง (ดีขึ้นประมาณ 1 วินาที) เราใช้การวัดหาเวลาอย่างง่าย ๆ ด้วยการใช้ Date เป็นตัวบอกถึงความแตกต่างก่อนและหลังการอ่านข้อมูลในไฟล์ โดยเปลี่ยนแปลง code บางส่วนดังนี้

```
    …
BufferedReader buf = null;       //buffer holding data
    …

try {
    System.out.println("Start: " + new Date());
    inFile = new FileReader(args[0]);
    buf = new BufferedReader(inFile);
    stream = new StreamTokenizer(buf);
    …
    …
    …

    finally {
        //close the file
        if(inFile != null) {
            try {
                inFile.close();
            }
            catch(IOException e) { /* do nothing */ }
        }
        //close stream
        if(buf != null) {
            try {
                buf.close();
                System.out.println("Stop: " + new Date());
            }
            catch(IOException e) { /* do nothing */ }
        }
    …
    …
```

อีกสิ่งหนึ่งที่เราได้พูดถึงก่อนหน้านี้ แต่ไม่เคยได้ใช้เลยก็คือ finally เพื่อแสดงให้ดูถึงประโยชน์ของ finally เราจึงใช้ finally เป็นตัวปิดไฟล์ในโปรแกรมให้เรา ผู้อ่านควรใช้ finally เป็นตัวปิดไฟล์เสมอ (โปรแกรมตัวอย่างในบทนี้หลาย ๆ ตัว อาจไม่ใช้ finally เป็นตัวปิดไฟล์ แต่นั่นไม่ได้หมายความว่า ผู้อ่านจะใช้ไม่ได้)

ถ้าผู้อ่านต้องการที่จะจับเวลาการทำงานของโปรแกรม หรือส่วนของโปรแกรม Java มี method อีกตัวหนึ่งที่ผู้อ่านสามารถเรียกใช้ได้นั่นก็คือ method currentTimeMillis() ดังตัวอย่างการใช้ที่แสดงไว้ด้านล่างนี้

```
//ElapsedTime.java

class ElapsedTime {
    public static void main(String[] args) {
        long startTime, stopTime;

        //get starting time
        startTime = System.currentTimeMillis();

        //do some looping
        for(int i = 0; i < 100000; i++)
            for(int j = 0; j < 100000; j++) ;

        //get ending time
        stopTime = System.currentTimeMillis();

        //calculate elapsed time
        double elapsedTime = (double)(stopTime - startTime) / 1000.0;

        System.out.print("Elapsed time: ");
        System.out.println(elapsedTime + " seconds.");
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

Elapsed time: 85.093 seconds.

โปรแกรม ElapsedTime.java เรียกใช้ currentTimeMillis() ก่อนและหลังจากการใช้ loop หลังจากนั้นก็หาเวลาที่ใช้ไปด้วยการนำเอาเวลาเริ่มต้นไปลบออกจากเวลาที่ได้หลังจากการใช้ loop

เราสามารถที่จะทำให้การทำงานกับข้อมูลที่เป็น text ได้ง่ายขึ้นถ้าเรากำหนดให้มี ตัวแบ่งข้อมูล (delimiter) ระหว่างข้อมูลแต่ละตัวที่อยู่ในไฟล์ ซึ่งตัวแบ่งที่ว่านี้จะเป็นตัวอักษรใด ๆ ก็ได้ที่ไม่มีส่วนเกี่ยวข้องกับข้อมูลที่อยู่ในไฟล์ เช่น เราอาจใช้เครื่องหมาย "|" เป็นตัวแบ่งข้อมูลในไฟล์ตัวอย่างที่ได้พูดถึงก่อนหน้านี้ ลองมาดูตัวอย่างการใช้ตัวแบ่งที่ว่านี้กัน

```
//TokenWithDelimiter.java - Using '|' as delimiter in a text file

import java.io.*;
import java.util.StringTokenizer;
import java.lang.Integer;

class TokenWithDelimiter {
    public static void main(String[] args) throws IOException {
        //data to write to file
        String[] items = {"Notebook", "Coffee Mug", "Paper cutter"};
        double[] prices = {12.0D, 5.0D, 12.5D};
```

```
        int[] units = {30, 40, 25};

        //check to see if a file is provided
        if(args.length < 1)  {
            System.err.println("Usage: TokenWithDelimeter file-
name.");
            System.exit(1);
        }
        writeToFile(args[0], items, prices, units);
        processData(args[0]);
    }

    //writing data in arrays to a given file
    private static void writeToFile(String fileName, String[] items,
                                    double[] prices, int[] units)
                      throws IOException {
        File dataFile = null;
        FileWriter fWriter = null;
        BufferedWriter buffer = null;
        PrintWriter out = null;

        try {
            dataFile = new File(fileName);
            fWriter = new FileWriter(dataFile);
            buffer = new BufferedWriter(fWriter);
            out = new PrintWriter(buffer);

            //printing data to file with '|' in between
            for(int i = 0; i < items.length; i++) {
                out.print(items[i]);
                out.print('|');
                out.print(prices[i]);
                out.print('|');
                out.println(units[i]);
            }
        }
        catch(IOException e) {
            System.err.println("Error writing to file");
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            if(out != null)
                out.close();
        }
    }

    //reading data from a given file & display to screen
    private static void processData(String fileName)
                    throws IOException {
        String item;   //item read
        double price;  //price read
        int unit;      //unit read
        File dataFile = null;
        FileReader fReader = null;
        BufferedReader buffer = null;
        String input = null;

        try {
```

```
            dataFile = new File(fileName);
            fReader = new FileReader(dataFile);
            buffer = new BufferedReader(fReader);
            input = buffer.readLine();   //read first line of data
            //keep reading until there's no more lines to read
            while(input != null) {
                //get token from input-line - skip '|'
                StringTokenizer token = new
                                        StringTokenizer(input, "|");
                item = token.nextToken();
                price = Double.parseDouble(token.nextToken());
                unit = Integer.parseInt(token.nextToken());
                //display to screen
                System.out.print(item + "\t" + price + "\t" + unit);
                System.out.println("\t" + price * unit);
                input = buffer.readLine();
            }
        }
        catch(IOException e) {
            System.err.println("Error reading file");
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            if(buffer != null)
                buffer.close();
        }
    }
}
```

เราใช้เครื่องหมาย '|' ในการแบ่งข้อมูลที่อยู่ในไฟล์ พร้อมทั้งใช้ class StringTokenizer เป็นตัวดึงข้อมูลที่อ่านมาจากไฟล์ หลังจากที่เราเขียนข้อมูลที่อยู่ใน arrays ทั้งสามตัวไปเก็บไว้ในไฟล์ด้วยการใช้ print() และ println() ผู้อ่านควรสังเกตถึงการเขียนข้อมูลในแต่ละครั้งว่า เราเขียน '|' ตามหลังเสมอ ยกเว้นการเขียนข้อมูลตัวสุดท้าย

```
dataFile = new File(fileName);
fWriter = new FileWriter(dataFile);
buffer = new BufferedWriter(fWriter);
out = new PrintWriter(buffer);

//printing data to file with '|' in between
for(int i = 0; i < items.length; i++) {
    out.print(items[i]);
    out.print('|');
    out.print(prices[i]);
    out.print('|');
    out.println(units[i]);
}
```

หลังจากนั้นเราก็อ่านข้อมูลกลับออกมาด้วย readLine() เราทำการดึงข้อมูลแต่ละตัวที่ถูกแบ่งด้วย '|' ด้วยการใช้ nextToken() แต่ก่อนที่เราจะทำการดึงข้อมูลทุกครั้งในแต่ละบรรทัด เราจะต้องกำหนดเงื่อนไขให้ token ของเราเสียก่อน ด้วยการใช้

StringTokenizer token = new StringTokenizer(input, "|");

เพื่อที่จะบังคับให้การดึงข้อมูลนั้นเกิดขึ้นระหว่างเครื่องหมาย '|' ข้อมูลตัวแรกที่เราดึงออกมาเป็น String เราจึงไม่ต้องแปลงข้อมูลตัวนี้ แต่ข้อมูลอีกสองตัวที่เราดึงออกเป็น double และ int ที่ถูกเก็บให้เป็น

String ในการเขียน ดังนั้นเราจึงต้องแปลงข้อมูลทั้งสองตัวนี้ หลังจากที่เราได้ข้อมูลทั้งสามตัวแล้วเราก็แสดงผลออกไปยังหน้าจอ

```
dataFile = new File(fileName);
fReader = new FileReader(dataFile);
buffer = new BufferedReader(fReader);
input = buffer.readLine();   //read first line of data
//keep reading until there's no more lines to read
while(input != null) {
    //get token from input-line - skip '|'
    StringTokenizer token = new StringTokenizer(input, "|");
    item = token.nextToken();
    price = Double.parseDouble(token.nextToken());
    unit = Integer.parseInt(token.nextToken());
    //display to screen
    System.out.print(item + "\t" + price + "\t" + unit);
    System.out.println("\t" + price * unit);
    input = buffer.readLine();
}
```

ในการเขียนไฟล์ของเราครั้งนี้เราใช้ PrintWriter เป็นตัวช่วย ซึ่งภายใน class PrintWriter นี้เราสามารถที่จะเรียกใช้ print() และ println() ได้ ทำให้การเขียนข้อมูลส่งไปยังไฟล์ของเราเหมือนกับการเขียนที่เราได้ทำมาก่อนหน้านี้ตอนที่เราส่งข้อมูลไปยังหน้าจอ แต่เราจะต้องใช้ PrintWriter ร่วมกับ FileWriter เพื่อให้การเขียนไปยังไฟล์ทำได้ และ BufferedWriter เพื่อให้ทำได้อย่างมีประสิทธิภาพ (การใช้ buffer จะช่วยให้การทำงานกับข้อมูลไวขึ้น)

ผลลัพธ์ที่ได้จากการ run

```
>java TokenWithDelimiter tokens.dat
Notebook        12.0    30      360.0
Coffee Mug      5.0     40      200.0
Paper cutter    12.5    25      312.5
```

Binary file

ในการเขียน binary data เข้าสู่ไฟล์นั้นเราจะต้องคำนึงถึงวิธีการในการที่จะเขียนข้อมูล เพราะในการเขียนข้อมูลเข้าสู่ไฟล์นั้น มีผลต่อการอ่านข้อมูลกลับออกมา ลำดับของข้อมูลที่เขียนเข้าสู่ไฟล์ จะต้องอ่านกลับออกมาด้วยขั้นตอนเดียวกัน

ในการเขียนข้อมูลที่เป็น binary data นั้น เราสามารถที่จะเลือกใช้ method หลาย ๆ ตัวที่ Java มีให้เป็นตัวช่วยในการเขียน เช่น

Method	Throws	ความหมาย
writeBoolean(boolean)	IOException	เขียนค่าของ boolean จำนวน 1 byte
writeShort(int)	IOException	เขียนค่าของ short จำนวน 2 byte
writeInt(int)	IOException	เขียนค่าของ int จำนวน 4 byte
writeLong(long)	IOException	เขียนค่าของ long จำนวน 8 byte
writeFloat(float)	IOException	เขียนค่าของ float จำนวน 4 byte
writeDouble(double)	IOException	เขียนค่าของ double จำนวน 8 byte
writeChar(int)	IOException	เขียนค่าของ char จำนวน 2 byte
writeChars(String)	IOException	เขียน String จำนวน 2 byte ต่อ 1 ตัวอักษร
writeUTF(String)	IOException	เขียน String จำนวน 1 byte ต่อตัวอักษร บวกอีก 2 byte สำหรับค่าของความยาวของ String

สำหรับการอ่านนั้น เราจะใช้ method นี้

Method	Throws	ความหมาย
readBoolean()	EOFException	อ่าน 1 byte ส่งค่า boolean กลับ
readShort()	EOFException	อ่าน 2 byte ส่งค่า Short กลับ
readInt()	EOFException	อ่าน 4 byte ส่งค่า int กลับ
readLong()	EOFException	อ่าน 8 byte ส่งค่า long กลับ
readFloat(0	EOFException	อ่าน 4 byte ส่งค่า float กลับ
readDouble()	EOFException	อ่าน 8 byte ส่งค่า double กลับ
readChar()	EOFException	อ่าน 2 byte ส่งค่า char กลับ
readUTF()	EOFException	อ่าน String UTF
skipBytes(int)	EOFException	ไม่อ่านข้อมูลจำนวนเท่ากับ byte ที่กำหนด

ตัวอย่างต่อไปจะเป็นตัวอย่างการเขียนข้อมูลเข้าสู่ไฟล์ในรูปแบบของ binary data หลังจากนั้นจะอ่านกลับออกมา เราจะเริ่มต้นด้วยการสร้าง record จำนวน 3 ตัว สร้างไฟล์สำหรับเก็บ record เหล่านี้ อ่าน record ออกมาทีละตัว ประมวล พร้อมทั้งแสดงผลไปยังหน้าจอ

Record คือกลุ่มข้อมูลที่ประกอบไปด้วยข้อมูลต่าง ๆ ซึ่งอาจเป็นข้อมูลชนิดเดียวกันหรือข้อมูลต่างชนิดกันก็ได้ ใน Java เราสร้าง record ด้วยการใช้ class เป็นตัวสร้าง เราจะกำหนดให้ record ของเรามี ข้อมูลอยู่ 3 field เหมือนกับตัวอย่างก่อนหน้านี้ คือ

1. description
2. price
3. unit

เรามาดูโปรแกรมตัวอย่างกันดีกว่า

```
//DataFile.java - Sequential Access file

import java.io.*;

class DataFile {
    public static void main(String[] args) throws IOException {
        FileOutputStream fout;    //file output stream
        DataOutputStream dout;    //data output stream
        FileInputStream fin;      //file input stream
        DataInputStream din;      //data input stream
        //setting up data to be written to a file
        String[] desc = {"Coffee Mug", "Bookmark", "Note book"};
        double[] prices = {5.00, 2.00, 10.00};
        int[] unit = {12, 50, 30};

        //variables storing data read from file
        String description;
        double price, total = 0.00;
        int unitPrice;

        //open file for writing
        try {
            fout = new FileOutputStream(args[0]);
            dout = new DataOutputStream(fout);
            for(int i = 0; i < desc.length; i++) {
                dout.writeUTF(desc[i]);
                dout.writeDouble(prices[i]);
                dout.writeInt(unit[i]);
            }
            dout.close();
```

```
        }
        catch(FileNotFoundException e) {
            System.err.println("Cannot open " + args[0]);
            return;
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.err.println("Usage: DataFile file-name.");
            return;
        }

        //open file for input
        try {
            fin = new FileInputStream(args[0]);
            din = new DataInputStream(fin);
            System.out.println("Description\tPrice\tUnit\tTotal");
            while(true) {
                description = din.readUTF();
                price = din.readDouble();
                unitPrice = din.readInt();
                total = price * unitPrice;
                display(description, price, unitPrice, total);
            }
        }
        catch(EOFException e) {
            //use this exception to stop the while loop
            //and exit the program
            System.exit(1);
        }
        catch(IOException e) {
            System.err.println("Error reading file: " + e.toString());
            System.exit(1);
        }
    }

    //display a record to screen
    public static void display(String d, double p, int u, double t) {
        System.out.println(d+"\t"+p+"\t"+u+"\t"+t);
    }
}
```

โปรแกรมของเราใช้ DataOutputStream และ DataInputStream เป็นตัวส่งข้อมูลไปเก็บไว้ที่ไฟล์ และเราใช้ for loop เป็นตัวกำหนดจำนวนครั้งของการเขียน

```
fout = new FileOutputStream(args[0]);
dout = new DataOutputStream(fout);
for(int i = 0; i < desc.length; i++) {
    dout.writeUTF(desc[i]);
    dout.writeDouble(prices[i]);
    dout.writeInt(unit[i]);
}
```

หลังจากนั้นเราก็อ่านข้อมูลออกมาจากไฟล์ ตามลำดับ ถ้าเราลองเปิดดูไฟล์ที่เก็บข้อมูลที่เราสร้างขึ้น เราจะไม่สามารถอ่านได้ เพราะข้อมูลเหล่านี้ถูกเก็บอยู่ในรูปแบบของ binary data

ในการอ่านข้อมูลออกมาจากไฟล์นั้นเราใช้ while loop ในการอ่านด้วยการกำหนดให้เป็น infinite loop โดยเราจะใช้ error ที่เกิดขึ้นเป็นตัวยุติการทำงานของ loop เนื่องจากว่าเรารู้ว่าในการอ่านข้อมูลออกจากไฟล์นั้น ในที่สุดเราก็จะอ่านเจอเครื่องหมายที่บ่งบอกถึงจุดจบของไฟล์ (EOF) ซึ่งอาจเป็นการเขียน code ที่ไม่ค่อยจะสวยเท่าไร แต่ก็ทำให้การทำงานของโปรแกรมเป็นไปตามที่เราต้องการ

```
fin = new FileInputStream(args[0]);
din = new DataInputStream(fin);
System.out.println("Description\tPrice\tUnit\tTotal");
while(true) {
    description = din.readUTF();
    price = din.readDouble();
    unitPrice = din.readInt();
    total = price * unitPrice;
    display(description, price, unitPrice, total);
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
>java DataFile data.dat

Description     Price   Unit    Total
Coffee Mug      5.0      12     60.0
Bookmark        2.0      50     100.0
Note book       10.0     30     300.0
```

จากผลลัพธ์ที่ได้จะเห็นว่าการจัดเรียงของข้อมูลที่ส่งไปยังหน้าจอ ยังดูไม่เรียบร้อยและสวยงามเท่าไร ถ้าต้องการให้การแสดงผลอยู่ในรูปแบบที่สวยงาม เราจะต้องสร้าง class ขึ้นมาใช้เองด้วยการ extends class ที่เราสร้างขึ้นจาก class PrintWritter พร้อมกับทำการสร้าง method output() ที่ทำการแสดงผลข้อมูลด้วยการจัดเรียงทางขวา และทำการ override method print() และ println() เราจะสร้าง class FormatWriter ดังที่แสดงให้ดูในโปรแกรมตัวอย่างที่ได้ดัดแปลงมาจากโปรแกรม DataFile.java

```
//DataFile.java - Sequential Access file
//-(modified for formatted output)

import java.io.*;

//for pretty printing (formatted - right justified)
class FormatWriter extends PrintWriter {
    private int width = 10;

    //default constructor
    FormatWriter(Writer output) {
        super(output);
    }

    //constructor with specified width
    FormatWriter(Writer out, int width) {
        super(out);
        this.width = width;
    }

    //set prefer output style
    private void output(String str) {
        //calculate spaces
        int blanks = width - str.length();
        //fill with blanks before actual data
        for(int i = 0; i < blanks; i++)
            super.print(' ');
        //print data
        super.print(str);
    }
```

```
    //our print() and println() methods with different types of data
    public void print(String str) {
        output(str);
    }

    public void println(String str) {
        output(str);
        super.println();
    }

    public void print(double value) {
        output(String.valueOf(value));
    }

    public void print(int value) {
        output(String.valueOf(value));
    }

    public void println(double value) {
        this.print(value);
        super.println();
    }
}

class DataFile {
    public static void main(String[] args) throws IOException {
        FileOutputStream fout;    //file output stream
        DataOutputStream dout;    //data output stream
        FileInputStream fin;      //file input stream
        DataInputStream din;      //data input stream
        //setting up data to be written to a file
        String[] desc = {"Coffee Mug", "Bookmark", "Note book"};
        double[] prices = {5.00, 2.00, 10.00};
        int[] unit = {12, 50, 30};
        double[] total = new double[3];

        //open file for writing
        try {
            fout = new FileOutputStream(args[0]);
            dout = new DataOutputStream(fout);
            for(int i = 0; i < desc.length; i++) {
                dout.writeUTF(desc[i]);
                dout.writeDouble(prices[i]);
                dout.writeInt(unit[i]);
            }
            dout.close();
        }
        catch(FileNotFoundException e) {
            System.err.println("Cannot open " + args[0]);
            return;
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.err.println("Usage: DataFile file-name.");
            return;
        }

        //open file for input
        //reuse variables desc, prices, and unit
        //calculate total fro each data item
```

```
        try {
            fin = new FileInputStream(args[0]);
            din = new DataInputStream(fin);
            boolean endOfFile = false;   //EOF marker
            int index = 0;               //array index
            while(!endOfFile) {
                try {
                    //store data in arrays before printing
                    desc[index] = din.readUTF();
                    prices[index] = din.readDouble();
                    unit[index] = din.readInt();
                    total[index] = prices[index] * unit[index];
                    index++;
                }
                catch(EOFException e) {
                    endOfFile = true; //stop while loop
                    din.close();      //close the file
                }
            }
            //display data - right justified
            display(desc, prices, unit, total);
        }
        catch(IOException e) {
            System.err.println("Error reading file: " + e.toString());
            System.exit(1);
        }
    }

    //display a record to screen
    private static void display(String[] d, double[] p,
                                int[] u, double[] t) {
        //create object from FileWriter with width of 12
        FormatWriter out = new FormatWriter(
                           new BufferedWriter(
                           new FileWriter(FileDescriptor.out)), 12);
        //display header
        out.print("Description");
        out.print("Prices");
        out.print("Unit");
        out.println("Total");
        //display contents of arrays
        for(int i = 0; i < d.length; i++) {
            out.print(d[i]);
            out.print(p[i]);
            out.print(u[i]);
            out.println(t[i]);
        }
        out.close();  //close the stream
    }
}
```

เราสร้าง class FormatWriter ขึ้นมาใหม่จาก class PrintWriter พร้อมทั้งกำหนดให้มี constructor 2 โดยที่ตัวแรกเป็น default constructor ที่มีหน้าที่เรียก constructor ของ class PrintWriter สำหรับการกำหนดช่องทางผ่านออกของข้อมูล ส่วนตัวที่สองเรากำหนดให้มีความกว้างของเนื้อที่ของข้อมูลที่ต้องการแสดง

```
FormatWriter(Writer out, int width) {
    super(out);
```

```
    this.width = width;
}
```

เราจะใช้ width ในการกำหนดจำนวนของช่องว่างที่เราต้องการแสดงก่อนการแสดงข้อมูลจริง เช่น ถ้าเรากำหนดให้ width มีค่าเป็น 10 และข้อมูลที่ต้องการแสดงมีค่าเป็น 2 ตัวอักษรเราก็จะทำการเขียนช่องว่างจำนวนเท่ากับ 8 ตัว เสร็จแล้วจึงเขียนข้อมูลตามช่องว่างเหล่านี้ เราทำได้ด้วยการเขียน method output() ให้ทำการคำนวณหาจำนวนช่องว่างที่ว่า เมื่อได้แล้วจึงส่งไปยังหน้าจอด้วยการเรียก print() ของ class PrintWriter หลังจากนั้นจึงส่งข้อมูลจริงออกไป

Method output() จะถูกเรียกใช้ใน print() และ println() ที่เราเขียนขึ้นมารองรับข้อมูลต่างชนิดกัน ซึ่งในที่นี้เรามีข้อมูลเพียงสามชนิด คือ string double และ int การเขียน print() และ println() ก็ไม่ยาก ดังนี้

```
public void print(int value) {
    output(String.valueOf(value));
}

public void println(double value) {
    this.print(value);
    super.println();
}
```

เราเพียงแต่เรียก output() ด้วยค่าที่ต้องการแสดงผล แต่เราจะต้องเปลี่ยนค่านี้ให้อยู่ในรูปแบบของ string ก่อนที่จะส่งค่านี้ไปให้ output()

เราเปลี่ยน display() ของเราใหม่ด้วยการรับ parameter ที่เป็น array แทนที่จะเป็นข้อมูลเดี่ยว ๆ เหมือนที่ทำก่อนหน้านี้ พร้อมทั้งประกาศตัวแปรจาก class FormatWriter ที่เราเขียนขึ้น ดังนี้

```
private static void display(String[] d, double[] p,
                            int[] u, double[] t) {
    //create object from FileWriter with width of 12
    FormatWriter out = new FormatWriter(
                       new BufferedWriter(
                       new FileWriter(FileDescriptor.out)), 12);
    //display header
    out.print("Description");
    out.print("Prices");
    out.print("Unit");
    out.println("Total");
    //display contents of arrays
    for(int i = 0; i < d.length; i++) {
        out.print(d[i]);
        out.print(p[i]);
        out.print(u[i]);
        out.println(t[i]);
    }
    out.close();  //close the stream
}
```

เพื่อให้การส่งข้อมูลออกทาง standard output เป็นไปได้ เราต้องสร้าง object จากสมาชิกที่ชื่อ out ของ class FileDescriptor (สมาชิกของ FileDescriptor มีอยู่สามตัวคือ in out และ err ซึ่งหมายถึงช่องทางของ input output และ error ตามลำดับ) หลังจากนั้นเราก็ส่ง object ตัวนี้ไปให้ BufferedWriter เพื่อสร้าง buffer สำหรับรองรับข้อมูลที่จะเกิดขึ้น และ buffer ตัวนี้จะเป็น parameter ตัวแรกที่ถูกส่งไปให้ constructor ของ FormatWriter และ 12 (ซึ่งเป็น width) จะถูกส่งไปเป็น parameter ตัวที่สอง ทำให้ object out ของเราสามารถใช้ช่องทางผ่านออกได้ หลังจากนั้นเราก็แสดงข้อมูลด้วยการเรียกใช้ print() และ println() ผ่าน object ตัวนี้

ในตัว main() ของเรา เราเปลี่ยนแปลง code ให้ทำการอ่านข้อมูลมาเก็บไว้ใน array สำหรับการส่งไปแสดงผลใน display()

```
    …
    …
boolean endOfFile = false;   //EOF marker
int index = 0;               //array index
while(!endOfFile) {
    try {
        //store data in arrays before printing
        desc[index] = din.readUTF();
        prices[index] = din.readDouble();
        unit[index] = din.readInt();
        total[index] = prices[index] * unit[index];
        index++;
    }
    catch(EOFException e) {
        endOfFile = true; //stop while loop
        din.close();      //close the file
    }
}
//display data - right justified
display(desc, prices, unit, total);
    …
    …
```

เรายังใช้ EOF error เป็นตัวยุติการทำงานของ loop เหมือนเดิม เพียงแต่แทนที่จะใช้ค่า -1 เราก็เลือกที่จะใช้ตัวแปร boolean เป็นตัวกำหนดการทำงานของ loop แทน

ผลลัพธ์ที่ได้จากการ run คือ

```
>java DataFile data.dat

 Description     Prices  Unit   Total
  Coffee Mug        5.0  12.0    60.0
    Bookmark        2.0  50.0   100.0
   Note book       10.0  30.0   300.0
```

เรายังสามารถที่จะทำการแสดงผลที่อยู่ในรูปแบบที่เราต้องการ ได้อีกหลายรูปแบบ ซึ่งก็ต้องขึ้นอยู่กับผู้เขียนโปรแกรมว่าต้องการให้ผลลัพธ์ออกมาในลักษณะไหน ผู้อ่านคงต้องใช้เวลาพอสมควรในการศึกษาและทดลองเขียน code ที่ทำให้การแสดงผลมีความสวยงามมากขึ้น

ตัวอย่างต่อไปนี้จะเป็นการแสดงถึงความแตกต่างในการใช้ writeUTF() กับการใช้ writeChars()

```
//WriteCharsAndWriteUTF.java - Differences between the to methods

import java.io.*;

class WriteCharsAndWriteUTF {
    public static void main(String[] args) throws IOException {
        File f = new File(args[0]);
        DataOutputStream out = new DataOutputStream(
                               new BufferedOutputStream(
                                new FileOutputStream(f)));
        out.writeUTF("If it does not work, blame the computer.");
        int utfSize = out.size();
```

```
        out.writeChars("If it does not work, blame the computer.");
        int charsSize = out.size() - utfSize;

        out.close();
        System.out.println("writeUTF() writes " + utfSize +
                           " bytes.");
        System.out.println("writeChars writes " + charsSize +
                           " bytes.");
    }
}
```

ผลลัพธ์ที่ได้จากการ run คือ

```
>java WriteCharsAndWriteUTF difs.dat
writeUTF() writes 42 bytes.
writeChars writes 80 bytes.
```

โปรแกรมของเราเขียน string จำนวน 40 ตัวอักษรไปยังไฟล์ด้วยการใช้ writeUTF() และ writeChars() จากผลของการ run จะเห็นว่า writeUTF() ใช้เพียงแค่ 1 byte ต่อตัวอักษรบวกกับอีก 2 byte สำหรับเก็บความยาวของ string ส่วน writeChars() ใช้ 2 byte ต่อตัวอักษร

การสร้างและใช้ Random-access file

ตัวอย่างก่อนหน้านี้เราเข้าหาข้อมูลในไฟล์แบบ ก่อน-หลัง หรือที่เรียกว่า sequential access ซึ่งเป็นวิธีการที่ค่อนข้างที่จะใช้เวลามาก ถ้าข้อมูลมีอยู่เป็นจำนวนมาก มีวิธีการอีกวิธีหนึ่งที่ทำให้การเข้าหา หรือค้นหาข้อมูลทำได้รวดเร็ว นั่นก็คือการเข้าหาแบบที่ไม่ต้องเข้าหาตามลำดับ ก่อน-หลัง หรือที่เรียกว่า Random-access

คำว่า random-access ในที่นี้หมายความว่าข้อมูลที่ถูกดึงออก หรือ นำเข้าไม่มีส่วนเกี่ยวข้องกับข้อมูลที่ถูกดึงออก หรือ นำเข้าก่อนหน้านี้ เช่น ในการค้นหาเบอร์โทรศัพท์ของใครสักคนผ่านทาง operator ผู้ค้นหา ณ เวลาปัจจุบันไม่มีความเกี่ยวข้องหรือความสัมพันธ์ใด ๆ กับการค้นหาที่ผ่านมาก่อนหน้า

ในการจัดการกับข้อมูลของ random-access file นั้นจะใช้ตัวชี้ตำแหน่ง (logical pointer) เป็นตัวกำหนดตำแหน่งของการนำข้อมูล เข้า-ออก ซึ่งการกำหนดตำแหน่งจะคำนวณจากจุดเริ่มต้นของไฟล์ (ไม่ใช่ตำแหน่งที่เก็บจริงในหน่วยความจำสำรอง) เราเรียกระยะทางจากจุดเริ่มต้นนี้ว่า offset Java กำหนดการเข้าหาตำแหน่งเหล่านี้ด้วยการใช้คำสั่ง seek เพราะฉะนั้นในการเข้าหาข้อมูล ถ้าเรารู้ขนาดของข้อมูลที่ตายตัว (fixed length record) เราก็สามารถเข้าหาข้อมูล ณ ตำแหน่ง p ด้วย p * length

เราจะเริ่มต้นด้วยการสร้าง record ตัวอย่าง โดยกำหนดให้ record ของเราประกอบด้วย

ชื่อ (first name)
นามสกุล (last name)
ปีที่เกิด (year of birth)
เงินเดือน (salary)

เราจะกำหนดให้ record เกิดจาก class ที่มีสมาชิกดังที่กล่าวข้างต้น พร้อมกับ method ต่าง ๆ ที่เกี่ยวข้องกับการทำงานกับ record นั้น ๆ

```
//RandomAccessFileDemo.java

import java.io.RandomAccessFile;
import java.io.*;

class RandomAccessFileDemo {
    public static void main(String[] args) throws IOException {
        //data to be written to a file
```

```
        String[] names = {"John", "Paul", "Stephen", "Hendrix"};
        String[] lasts = {"Kawakami", "Collins", "Anderson", "McCoy"};
        int[] years = {1965, 1978, 1985, 1972};
        double[] pays = {2400.0, 500.0, 5000.0, 9500.0};

        //use to randomly select the position in a file
        //to write to/read from e.g. random[0] is at the second
        //position, random[2] is at the frist position etc.
        int []random = {2, 3, 0, 1};

        //writing to a file
        RandomAccessFile f = new RandomAccessFile("emp.dat", "rw");
        try {
            for(int i = 0; i < 4; i++) {
                Employee emp = new Employee(names[i], lasts[i],
                                            years[i], pays[i]);
                emp.write(f, random[i]);
            }
        }
        catch(IOException e) {
            System.err.println("Error writing file");
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            if(f != null) {
                f.close();
            }
        }

        //reading from a file
        RandomAccessFile  fin = null;
        try {
            fin = new RandomAccessFile("emp.dat", "r");
            Employee em = new Employee();
            System.out.println("Number of records = " +
                               em.size(fin) + "\n");
            for(int i = 0; i < 4; i++) {
                em.read(fin, random[i]);
                System.out.print("Record #" + (i+1) + ": ");
                em.show();
            }
        }
        catch(IOException e) {
            System.err.println("Error reading file");
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            if(fin != null)
                fin.close();
        }
    }
}
```

โปรแกรมตัวอย่างของเราใช้ array 4 ตัวเป็นตัวเก็บ record ที่ต้องการเขียนไปยังไฟล์ และเพื่อเป็นการแสดงให้เห็นถึงการเข้าหาไฟล์ในรูปแบบของความเป็น random เราจึงใช้ array random เป็นตัวกำหนดถึงตำแหน่งของ record ที่เราต้องการจะเขียน ซึ่งเราได้กำหนดให้

Record ตัวแรกเขียนในตำแหน่งที่ 2 (random[0])
Record ตัวที่สองเขียนในตำแหน่งที่ 3 (random[1])
Record ตัวที่สามเขียนในตำแหน่งที่ 0 (random[2])
Record ตัวที่สุดท้ายเขียนในตำแหน่งที่ 1 (random[3])

เราเก็บข้อมูลเหล่านี้ด้วยการสร้าง object จาก class Employee ซึ่งเป็น class ที่ทำหน้าที่หลักในการเขียน และอ่าน record ไปยัง และออกจากไฟล์ ตามลำดับ ซึ่งมี code ดังนี้

```
//Employee.java - Simple record for random-access file

import java.io.*;
import java.io.RandomAccessFile;
import java.lang.String;

class Employee {
    private String firstName;//15 characters (30 bytes)
    private String lastName; //15 characters (30 bytes)
    private int year;        //4 bytes
    private double salary;   //8 bytes
    private static final int RECORD_SIZE = 72;

    //default constructor
    Employee() {
        firstName = "";
        lastName = "";
        year = 0;
        salary = 0.0D;
    }

    //setting up fields
    Employee(String firstName, String lastName,
              int year, double salary) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.year = year;
        this.salary = salary;
    }

    //write a record to a file with a given position
    public void write(RandomAccessFile file, int position)
            throws IOException {
        try {
            //locate a position
            file.seek((long)(position * RECORD_SIZE));
            //write equal-length strings
            writeString(file, firstName, 15);
            writeString(file, lastName, 15);
            //write int and double
            file.writeInt(year);
            file.writeDouble(salary);
        }
        catch(IOException e) {
            System.err.println("Error writing file");
            e.printStackTrace();
            System.exit(1);
        }
    }
```

```
    //reading a record from a file with a given position
    public void read(RandomAccessFile file, int position)
            throws IOException {
        try {
            //locate position to read
            file.seek((long)(position * RECORD_SIZE));
            //reading strings
            firstName = readString(file, 15);
            lastName = readString(file, 15);
            //reading int and double
            year = file.readInt();
            salary = file.readDouble();
        }
        catch(IOException e) {
            System.err.println("Error reading file");
            e.printStackTrace();
            System.exit(1);
        }
    }

    //helper method to write a string with specified length
    public void writeString(DataOutput out, String s, int len)
            throws IOException {
        for(int i = 0; i < len; i++) {
            if(i < s.length())
                out.writeChar(s.charAt(i));
            else
                out.writeChar(0);
        }
    }

    //helper method to read a string
    public String readString(DataInput in, int len)
            throws IOException {
        String s = "";
        int i = 0;
        while(i < len) {
            char c = in.readChar();
            if(c != 0)
                s += c;
            i++;
        }
        return s;
    }

    //number of records in file
    public int size(RandomAccessFile file) throws IOException{
        int size = 0;
        try {
            size = (int)file.length() / RECORD_SIZE;
        }
        catch(IOException e) {
            System.err.println("Error reading file.");
            System.exit(1);
        }
        return size;
    }
```

```
    //display a record to screen
    public void show() {
        System.out.print(firstName + " ");
        System.out.print(lastName + " ");
        System.out.print(year + " ");
        System.out.println(salary);
    }
}
```

method หลัก ๆ ที่อยู่ใน class Employee คือ

```
write(RandomAccessFile, position)
read(RandomAccessFile, position)
writeString(DataOutptu, String, length)
readString(DataInput, length)
size(RandomAccessFile)
```

เนื่องจากว่าเราต้องรู้ขนาดของ record ของเราว่ามีความยาวเท่าไร เราจึงจะสามารถที่จะเลื่อน logical pointer ของเราไปยังตำแหน่งต่าง ๆ ที่เราต้องการได้ ดังนั้นเราจึงต้องกำหนดขนาดของ record ภายในโปรแกรมของเราเพื่อให้การทำงานสะดวกขึ้น ทั้งนี้การกำหนดขนาดก็ต้องขึ้นอยู่กับการเลือกใช้วิธีการเขียนของเราว่าใช้ method ตัวไหนเป็นตัวเขียน ถ้าเราใช้ writeUTF() เราก็ต้องกำหนดขนาดแบบหนึ่ง แต่ถ้าเราใช้ writeChars() เราก็ต้องกำหนดขนาดอีกแบบหนึ่ง (ดูความแตกต่างของการใช้จากตาราง) สำหรับโปรแกรมที่เราเขียนขึ้น เรากำหนดให้ขนาดของ record เป็น 72 byte ก็เพราะว่าเราใช้ writeChar() เป็นตัวเขียน ดังนั้น ทั้งชื่อและนามสกุลใช้ 30 byte ส่วน year และ salary ใช้ 4 และ 8 byte ตามลำดับ

หน้าที่ของ write() คือ การนำข้อมูลจาก field ต่าง ๆ ของ Employee object เข้าสู่ไฟล์ในตำแหน่งที่ได้กำหนดไว้ ด้วยการใช้ seek() เป็นตัวกำหนดตำแหน่ง

```
file.seek((long)(position * RECORD_SIZE));
```

เราคูณ position ด้วย RECORD_SIZE ก็เพื่อที่จะหาตำแหน่งได้ถูกต้อง เนื่องจากว่าเรากำหนดให้ field 2 ตัวแรกมีความยาวเท่ากับ 15 ตัวอักษร (30 byte) หลังจากนั้นเราก็เขียน field ทั้งสองไปยังไฟล์ด้วยการเรียกใช้ writeString() ซึ่งทำหน้าที่ในการเขียนจำนวน byte ที่ได้กำหนดไว้ และจะเขียนค่า 0 ในตำแหน่งที่ว่างอยู่ ถ้าความยาวของ field ที่เขียนมีน้อยกว่า 15 ตัวอักษร

```
//write equal-length strings
writeString(file, firstName, 15);
writeString(file, lastName, 15);
```

เมื่อเราเขียน field ทั้งสองเสร็จเราก็เขียน อีกสอง filed ที่เหลือด้วย writeInt() และ writeDouble() โดยไม่ต้องทำอะไรพิเศษ

```
//write int and double
file.writeInt(year);
file.writeDouble(salary);
```

code ของ writeString() ก็ไม่ยากอะไร เราเขียนข้อมูลทีละตัวไปยังไฟล์จนกว่าจะหมดข้อมูลที่อยู่ใน string ถ้าความยาวของ string ที่เราเขียนมีน้อยกว่าความยาวที่กำหนดไว้ เราก็เขียนค่า 0 ลง ณ ตำแหน่งนั้น

```
public void writeString(DataOutput out, String s, int len)
        throws IOException {
    for(int i = 0; i < len; i++) {
        if(i < s.length())
            out.writeChar(s.charAt(i));
        else
```

```
            out.writeChar(0);
    }
}
```

ในการอ่าน filed กลับออกมาก็เหมือนกันหลังจากที่ใช้ seek() หาตำแหน่งได้แล้ว เราก็เรียกใช้ readString() ในการอ่านสอง field แรก และใช้ readInt() กับ readDouble() ในการอ่านอีกสอง field ที่เหลือ

Code ของ readString() ก็ง่าย ๆ เราทำการอ่านทุกตัว แต่จะเก็บไว้เฉพาะตัวที่ไม่ใช่ 0 เพราะ 0 เป็นค่าที่เราใช้แทนตำแหน่งที่ว่างในการเขียน string เข้าสู่ไฟล์

```
public String readString(DataInput in, int len) throws IOException {
    String s = "";
    int i = 0;
    while(i < len) {
        char c = in.readChar();
        if(c != 0)
            s += c;
        i++;
    }
    return s;
}
```

เราใช้ readChar() ในการอ่านข้อมูลแต่ละตัว และเราจะตรวจสอบว่าค่าที่อ่านได้เป็น 0 หรือไม่ ถ้าไม่เราก็เชื่อมค่านี้เข้าสู่ string s พร้อมกับเลื่อนไปอ่านข้อมูลตัวต่อไป ทำการอ่านและเชื่อมจนกว่าจะหมดข้อมูล สำหรับการหาจำนวนของ record ที่มีอยู่ในไฟล์นั้นเราหาได้ด้วยการใช้ ขนาดของไฟล์หารด้วยขนาดของ record ที่เราได้กำหนดไว้

```
public int size(RandomAccessFile file) throws IOException{
    int size = 0;
    try {
        size = (int)file.length() / RECORD_SIZE;
    }
    catch(IOException e) {
        System.err.println("Error reading file.");
        System.exit(1);
    }
    return size;
}
```

เราใช้ readChar() และ writeChar() ในการเขียนและอ่านข้อมูล ดังนั้นเราต้องคำนึงถึงจำนวนของ byte ที่เราต้องใช้สำหรับ record แต่ละตัวว่าถูกต้องตามที่ได้กำหนดหรือไม่ สิ่งที่ต้องคำนึงถึงในเรื่องของการใช้ random access file ก็คือ field แต่ละ field ควรเป็น field ที่มีขนาดตายตัว เพราะจะทำให้การค้นหาตำแหน่งทำได้อย่างถูกต้อง

ผลลัพธ์ที่เราได้จากการ run คือ

```
Number of records = 4

Record #1: John Kawakami 1965 2400.0
Record #2: Paul Collins 1978 500.0
Record #3: Stephen Anderson 1985 5000.0
Record #4: Hendrix McCoy 1972 9500.0
```

เรามาลองดูตัวอย่างการปรับปรุงข้อมูล (update) ที่อยู่ในไฟล์กัน

การ update ข้อมูลก็คล้าย ๆ กับการเขียนข้อมูลเข้าสู้ไฟล์ เราต้องหาตำแหน่งของ record ที่ต้องการ update ให้เจอ ซึ่งเมื่อเจอแล้วการเปลี่ยนแปลงข้อมูลก็สามารถทำได้ เราเลือกที่จะเปลี่ยนแปลง record ของเราทุก field เพื่อเป็นการแสดงถึงวิธีการในการ update แต่ในทางปฏิบัติจริง คงไม่มีใครทำแบบนี้ คำสั่งหลัก ๆ ในการ update คือ

```
f.seek((long)(recNum - 1) * em.recSize());
em.write(f, recNum - 1);
```

record ของเราถูกเก็บในตำแหน่งที่น้อยกว่าที่เป็นจริงอยู่หนึ่งค่า ดังนั้นการใช้ seek() เราจึงต้องหักออกหนึ่งค่า เราได้เขียน method ขึ้นใหม่อีกหลายตัวเพื่อช่วยให้การกำหนด และ การดึงข้อมูลออกจาก record ทำได้โดยไม่มีปัญหา method ที่เพิ่มขึ้นใน class Employee คือ

```
public void setFirstName(String name) {
    firstName = name;
}

public void setLastName(String name) {
    lastName = name;
}

public void setSalary(double pay) {
    salary = pay;
}

public int recSize() {
    return RECORD_SIZE;
}

public void setYear(int y) {
    year = y;
}
```

สำหรับกระบวนการหลัก ๆ ที่เกี่ยวข้องกับการ update ข้อมูลก็ไม่มีอะไรมาก หลังจากที่รับข้อมูลใหม่มาจาก keyboard แล้ว พร้อมกับหาตำแหน่งที่ต้องการ update เจอเราก็ทำการเขียนข้อมูลใหม่ทับลงที่เดิม

```
    …
System.out.print("Enter record number: ");
buffer = new BufferedReader(new InputStreamReader(System.in));
str = buffer.readLine();
recNum = Integer.parseInt(str);
System.out.print("Enter first name: ");
emp.setFirstName(buffer.readLine());
System.out.print("Enter last name: ");
emp.setLastName(buffer.readLine());
System.out.print("Enter year of birth: ");
emp.setYear(Integer.parseInt(buffer.readLine()));
System.out.print("Enter salary: ");
emp.setSalary(Double.parseDouble(buffer.readLine()));
```

และ code สำหรับการเขียนทับที่เราได้พูดก่อนหน้านี้

```
    …
f.seek((long)(recNum - 1) * em.recSize());
em.write(f, recNum - 1);
```

ผลลัพธ์ของการ run

```
>java Update temp.dat
```

```
Enter record number: 1
Enter first name: My name
Enter last name: Is changing
Enter year of birth: 1965
Enter salary: 2500

Record #1: My name Is changing 1965 2500.0
Record #2: Hendrix McCoy 1972 9500.0
Record #3: John Kawakami 1965 2400.0
Record #4: John Kowalski 1990 5400.0

>java Update temp.dat
Enter record number: 3
Enter first name: Jennie
Enter last name: Saiudom
Enter year of birth: 1995
Enter salary: 500

Record #1: My name Is changing 1965 2500.0
Record #2: Hendrix McCoy 1972 9500.0
Record #3: Jennie Saiudom 1995 500.0
Record #4: John Kowalski 1990 5400.0
```

โปรแกรมตัวอย่างนี้ update ทุก field ที่มีอยู่ใน record ด้วยการกำหนดข้อมูลที่เข้ามาใหม่ให้กับ object จาก class Employee เสร็จแล้วก็เขียน object ใหม่นี้ลงในไฟล์ ซึ่งการกระทำแบบนี้เราไม่สามารถที่จะ update บาง field ที่มีอยู่ได้ เราต้อง update ทั้งหมด สำหรับการ update เฉพาะ field ที่ต้องการนั้น จะทิ้งไว้ให้เป็นแบบฝึกหัดท้ายบทต่อไป

ตัวโปรแกรมทั้งหมด มีดังนี้คือ

```
//Update.java - Updating Random-access file

import java.io.RandomAccessFile;
import java.io.*;
import java.lang.Integer;

class Update {
    public static void main(String[] args) throws IOException {
        RandomAccessFile f = null;
        Employee em = new Employee();

        if(args.length < 1) {
            System.err.println("Usage: Update file-name");
            System.exit(1);
        }

        try {
            f = new RandomAccessFile(args[0], "rw");
            int recNum = getNewData(f, em);  //get new data
            updateData(f, recNum, em);       //update record
            showData(f, args[0]);            //display records
        }
        catch(IOException e) {
            System.err.println("Error processing file");
            e.printStackTrace();
            System.exit(1);
        }
    }
```

```
    //get new data from keyboard
    private static int getNewData(RandomAccessFile f, Employee emp)
                    throws IOException {
        BufferedReader buffer = null;
        InputStreamReader iStream = null;
        String str = "";
        int recNum = 0;

        try {
            System.out.print("Enter record number: ");
            buffer = new BufferedReader(
                      new InputStreamReader(System.in));
            str = buffer.readLine();
            recNum = Integer.parseInt(str);
            System.out.print("Enter first name: ");
            emp.setFirstName(buffer.readLine());
            System.out.print("Enter last name: ");
            emp.setLastName(buffer.readLine());
            System.out.print("Enter year of birth: ");
            emp.setYear(Integer.parseInt(buffer.readLine()));
            System.out.print("Enter salary: ");
            emp.setSalary(Double.parseDouble(buffer.readLine()));
        }
        catch(IOException e) {
            System.err.println("Error readin file");
            e.printStackTrace();
            System.exit(1);
        }
        return recNum; //record number
    }

    //update record
    private static void updateData(RandomAccessFile f,
                                   int recNum, Employee em)
                    throws IOException {
        try {
            //locate record
            f.seek((long)(recNum - 1) * em.recSize());
            //write the whole record
            em.write(f, recNum - 1);
        }
        catch(IOException e) {
            System.err.println("Error writing file");
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            if(f != null)
                f.close();
        }
    }

    //display all records in this file
    private static void showData(RandomAccessFile f, String fName)
                    throws IOException {
        try {
            f = new RandomAccessFile(fName, "r");
            Employee em = new Employee();
```

```
            int num = em.size(f);
            for(int i = 0; i < 4; i++) {
                em.read(f, i);
                System.out.print("Record #" + (i+1) + ":");
                em.show();
            }
        }
        catch(IOException e) {
            System.err.println("Error reading file");
            e.printStackTrace();
            System.exit(1);
        }
        finally {
            if(f != null)
                f.close();
        }
    }
}
```

สรุป

เราได้แสดงตัวอย่างของการทำงานกับ text file และ binary file การอ่าน การเขียน รวมไปถึงกระบวนการต่าง ๆ ที่เกี่ยวข้องกับไฟล์ โดยสรุปแล้วเราได้พูดถึง

- ✓ การตรวจสอบข้อมูลที่เกี่ยวข้องกับไฟล์ด้วย File และ method ต่าง ๆ เช่น ifFile() canRead() เป็นต้น
- ✓ การใช้ FileReader FileWriter FileInputStream FileOutputStream
- ✓ การใช้ DataOutputStream BufferedOutputStream
- ✓ การใช้ StreamTokenizer และ StringTokenizer
- ✓ การสร้างและใช้งาน text file
- ✓ การใช้ delimiter ในการอ่านข้อมูลจาก text file
- ✓ การสร้างและใช้งาน binary file
- ✓ การสร้างและใช้งาน Random-access file
- ✓ การ update ข้อมูลใน random-access file

แบบฝึกหัด

1. จงเขียนโปรแกรมที่ทำหน้าที่แสดงข้อมูลเกี่ยวกับไฟล์ทุกตัวที่มีอยู่ใน directory นั้น ๆ เช่น ถ้าเจอไฟล์ให้แสดงถึง ขนาด วันที่สร้าง ถ้าเจอ directory ให้แสดงถึงจำนวนไฟล์ที่มีอยู่ใน directory นั้น พร้อมกับวันที่ directory นี้ถูกสร้างขึ้น

2. จงเขียนโปรแกรมที่ copy ข้อมูลจากไฟล์หนึ่งไปยังอีกไฟล์หนึ่งผ่านทาง command-line

3. จงเขียนโปรแกรมที่สร้างข้อมูลชนิด int ด้วยการสุ่มจำนวนเท่ากับ 100 ตัว หลังจากนั้นให้นำข้อมูลทั้งหมดไปเก็บไว้ใน text file โดยกำหนดให้จำนวนของข้อมูลต่อแถวเท่ากับ 10 ตัว

4. จงเขียนโปรแกรมที่อ่านข้อมูลจากไฟล์ที่เขียนขึ้นในข้อ 3 เสร็จแล้วให้คำนวณหาผลรวมของข้อมูลในแต่ละแถว หลังจากนั้นให้เขียนข้อมูลในแต่ละแถวรวมทั้งผลรวมที่หาได้ลงในไฟล์ตัวเดิม (ข้อมูลในแต่ละแถวจะมีจำนวนเท่ากับ 11 ตัวโดยที่ตัวสุดท้ายในแถวเป็นผลรวมของข้อมูลในแถวนั้น)

5. จงปรับปรุงโปรแกรมที่เขียนขึ้นในข้อ 4 โดยให้มีการคำนวณหาค่าเฉลี่ยของข้อมูลทุกตัวในแนวตั้ง (column) นำผลลัพธ์ที่ได้พร้อมทั้งข้อมูลในแต่ละแถวไปเก็บไว้ในไฟล์ตัวใหม่

6. จงเขียนโปรแกรมที่อ่านข้อมูลในรูปแบบที่กำหนดให้ด้านล่างนี้ หลังจากนั้นให้แสดงข้อมูลที่อ่านได้ไปยังหน้าจอ โดยไม่ต้องแสดง delimiter ที่ใช้ ให้กำหนดจำนวนของข้อมูลที่มีอยู่ในไฟล์จำนวนเท่ากับ 10 แถว

John Longman+25.0+30.50+48.00
Peter Wang+35.00+50.00+98.00
…
…
Mike Kawakami+90.00+80.00+85.00

7. จงเขียนโปรแกรมที่สร้าง binary file จาก class Student ที่กำหนดให้นี้ โดยทำการอ่านข้อมูลเบื้องต้นมาจาก keyboard ซึ่งมีข้อมูลดังนี้

   ชื่อต้น เป็น String จำนวน 15 ตัวอักษร
   นามสกุล เป็น String จำนวน 15 ตัวอักษร
   เกรด เป็น array ที่เก็บ double เช่น 85.0 90.5 ทั้งนี้จำนวนของเกรดที่มีอยู่จะมีกี่ตัวก็ได้
   เสร็จแล้วให้ทำการอ่านข้อมูลกลับออกมาจากไฟล์ พร้อมทั้งคำนวณหาเกรดเฉลี่ยของข้อมูล (นักศึกษา) ทุกตัว เมื่อได้เกรดเฉลี่ยแล้วให้แสดงผลออกทางหน้าจอ

   ```
   class Student {
         String firstName;
         String lastName;
         double[] grades;
   }
   ```

8. จงเขียนโปรแกรมที่สร้าง random-access file จากโครงสร้างของ Student ที่มีอยู่ในข้อ 7 โดยให้ user เป็นผู้กำหนดตำแหน่งของ record ที่ต้องการเขียนเอง ให้เขียน method ที่แสดงข้อมูลของ record (ที่กำหนดมาจาก keyboard) ไปยังหน้าจอ

9. จงปรับปรุงโปรแกรม Update ในบทนี้ให้มีการ update ได้เฉพาะ field ที่เป็น salary เท่านั้น

10. จงปรับปรุงโปรแกรมในข้อ 6 ด้วยการเพิ่ม method ในการคำนวณหาผลรวมของข้อมูลในแต่ละแถว หลังจากนั้นให้นำข้อมูลทั้งหมดไปเก็บไว้ในไฟล์ตัวใหม่ โดยให้ผลรวมที่หาได้เป็นข้อมูลตัวสุดท้ายในแถวนั้น

ตารางต่าง ๆ

ตารางที่ 1 Unicode characters – 128 ตัวแรก											
dec	hex	char	Dec	hex	char	dec	hex	char	dec	hex	char
0	0000	<NUL>	33	0021	!	66	0042	B	99	0063	c
1	0001	<SOH>	34	0022	"	67	0043	C	100	0064	d
2	0002	<STX>	35	0023	#	68	0044	D	101	0065	e
3	0003	<ETX>	36	0024	$	69	0045	E	102	0066	f
4	0004	<EOT>	37	0025	%	70	0046	F	103	0067	g
5	0005	<ENQ>	38	0026	&	71	0047	G	104	0068	h
6	0006	<ACK>	39	0027	'	72	0048	H	105	0069	i
7	0007	<BEL>	40	0028	(	73	0049	I	106	006A	j
8	0008	<BS>	41	0029	)	74	004A	J	107	006B	k
9	0009	<HT>	42	002A	*	75	004B	K	108	006C	l
10	000A	<LF>	43	002B	+	76	004C	L	109	006D	m
11	000B	<VT>	44	002C	'	77	004D	M	110	006E	n
12	000C	<FF>	45	002D	-	78	004E	N	111	006F	o
13	000D	<CR>	46	002E	.	79	004F	O	112	0070	p
14	000E	<SO>	47	002F	/	80	0050	P	113	0071	q
15	000F	<SI>	48	0030	0	81	0051	Q	114	0072	r
16	0010	<DLE>	49	0031	1	82	0052	R	115	0073	s
17	0011	<DC1>	50	0032	2	83	0053	S	116	0074	t
18	0012	<DC2>	51	0033	3	84	0054	T	117	0075	u
19	0013	<DC3>	52	0034	4	85	0055	U	118	0076	v
20	0014	<DC4>	53	0035	5	86	0056	V	119	0077	w
21	0015	<NAK>	54	0036	6	87	0057	W	120	0078	x
22	0016	<SYN>	55	0037	7	88	0058	X	121	0079	y
23	0017	<ETB>	56	0038	8	89	0059	Y	122	007A	z
24	0018	<CAN>	57	0039	9	90	005A	Z	123	007B	{
25	0019	<EM>	58	003A	:	91	005B	[	124	007C	|
26	001A	<SUB>	59	003B	;	92	005C	\	125	007D	}
27	001B	<ESC>	60	003C	<	93	005D	]	126	007E	~
28	001C	<FS>	61	003D	=	94	005E	^	127	007F	<DEL>
29	001D	<GS>	62	003E	>	95	005F	_	128	0080	
30	001E	<RS>	63	003F	?	96	0060				
31	001F	<US>	64	0040	@	97	0061	a			
32	0020	blank	65	0041	A	98	0062	b			

ตารางที่ 2 Constants ที่ใช้บ่อย ๆ ในโปรแกรม	
ชื่อ	ความหมาย
Math.E	ค่าของ e (natural logarithm)
Math.PI	ค่าของ PI (3.145)
Float/Double.NEGATIVE_INFINITY	infinity ที่เป็นลบ
Float/Double.POSITIVE_INFINITY	infinity ที่เป็นบวก
Float/Double.NaN	ไม่ใช่ตัวเลข
Byte/Short/Integer/Long/Float/Double.MAXIMUM_VALUE	ค่าสูงสุด
Byte/Short/Integer/Long/Float/Double.MINIMUM_VALUE	ค่าต่ำสุด
System.in	ทางเข้าข้อมูล (keyboard)
System.out	ทางออกข้อมูล (จอ)
System.err	ทางออกของ error (จอ)

ตารางที่ 3 Numeric Types				
ชนิด	bit	ค่าเบื้องต้น	ค่าต่ำสุด	ค่าสูงสุด
byte	8	0	-128	127
short	16	0	-32768	32767
int	32	0	-2147483648	2147483647
long	64	0	-9223372036854775808	9223372036854775807
float	32	0.0f	1.40129846432481707e-45f	3.40282346638528860e+38f
double	64	0.0d	4.94065645841246544e-324	1.79769313486231570e+308

ตารางที่ 4 Escape Sequences	
Escape Sequence	ความหมาย
\b	back space
\t	tab
\n	LF (ขึ้นบรรทัดใหม่)
\f	FF (ขึ้นหน้าใหม่)
\r	CR (กลับไปยังจุดเริ่มต้น)
\"	double quote
\'	single quote
\\	back slash
\uxxxx	Unicode จาก \u ตามด้วยเลขฐาน 16 สี่ตัว
\xxx	เลขฐานแปด เช่น \101

ตารางที่ 5 Format Syntax สำหรับตัวเลข			
form	ค่า	ผลลัพธ์	ความหมาย
0000	314	0314	0 –ใช้แทน digit
###0	314	314	# - ใช้แทน digit
##0.0#	3.10	3.1	. - จุดทศนิยม
#,##0	3000	3,000	, - ใช้แบ่งกลุ่มตัวเลข
##0.00%	3.1415	314.15%	% - คูณตัวเลขด้วย 100 เพื่อแสดง %
\u00A4##0	314	$314	\u00A4 – แสดงสกุลเงินตาม locale ของประเทศ