Part 9

Java Programming Language
Mr.Rungrote Phonkam
rungrote@it.kmitl.ac.th

Contents

1 Exception Handling
- 1.1. Implicitly Exception
- 1.2. Explicitly Exception

2. Handle Exception

3. Threads

1 Exception Handling

- ข้อผิดพลาดที่เกิดขึ้นขณะใช้งานโปรแกรม (Run-time)
- JVM จะดูแลเมื่อเกิดเอ็กเซพชันขึ้น ก็จะส่งเมสเสจออกมาที่คอนโซล
- ผู้เขียนโปรแกรม สามารถดักจับการเกิดข้อผิดพลาดนี้ แล้วกระทำการอย่างใดอย่างหนึ่ง แทนการกระทำของ JVM
- กลไก Exception Handling นี้มีส่วนช่วยให้ โปรแกรมภาษาจาวา มีความทนทานในการใช้งาน (Robust)
- สาเหตุของการเกิด Exception คือ Implicitly Exception และ Explicitly Exception

1.1. Implicitly Exception

Implicitly Exception

คือสาเหตุที่เกิดจากการทำงาน ของสเตจเม็นต์บางคำสั่ง ภายในตัวโปรแกรมเอง

- เช่นการหารค่าใดๆด้วยค่า 0
- การอ้างถึงตำแหน่งในอะเรย์ที่ไม่มีอยู่
- การแปลงตัวอักษรที่ไม่ใช่ตัวเลข

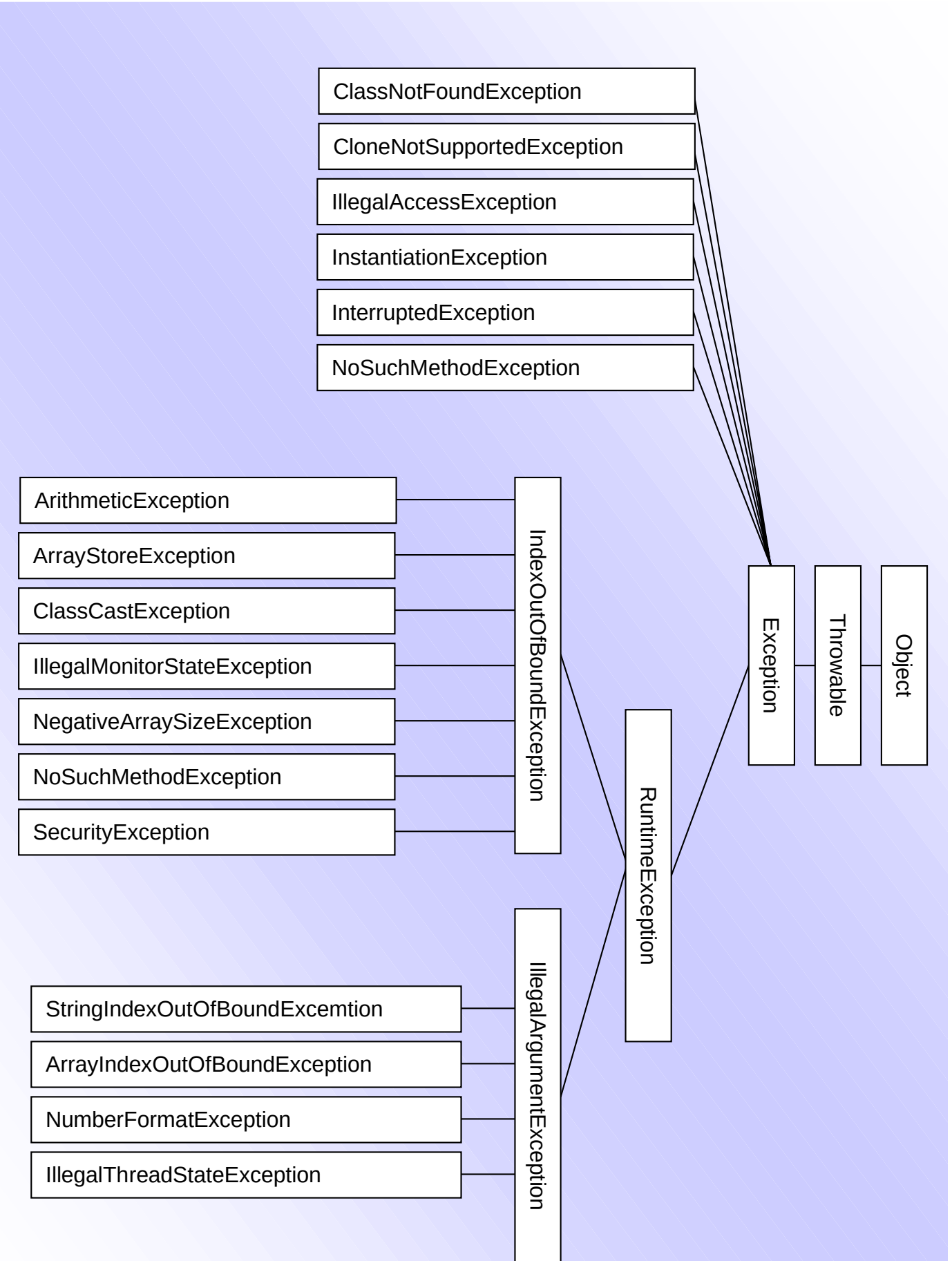
1.1. Implicitly Exception

Division by Zero

```
class DivisionByZero
{    public static void main(String args[])
            {           int a=10, b = 0;
                        System.out.println(a/b);
            }
}
```

java DivisionByZero

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at DivisionByZero.main(5_14.java:4)
```


ClassNotFoundException
CloneNotSupportedException
IllegalAccessException
InstantiationException
InterruptedException
NoSuchMethodException
ArithmeticException
ArrayStoreException
ClassCastException
IllegalMonitorStateException
NegativeArraySizeException
NoSuchMethodException
SecurityException
IndexOutOfBoundException
StringIndexOutOfBoundExcemtion
ArrayIndexOutOfBoundException
NumberFormatException
IllegalThreadStateException
IllegalArgumentException
RuntimeException
Exception
Throwable
Object

1.1. Implicitly Exception

Out of Rang

```
class OutOfRange
{    public static void main(String args[])
            {           int a[] = { 10, 20, 30 };
                        System.out.println(a[3]);
            }
}
```

java OutOfRange

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
at OutOfRange.main(5_15.java:4)
```

1.2. Explicitly Exception

Explicitly Exception

คือสาเหตุที่เกิดโดยความจงใจ ของโปรแกรมเมอร์เอง

- สร้างเอ็กเซ็พชั่นเมื่อข้อมูลที่ได้รับไม่เป็นที่ต้องการ เช่น ต้องการ ค่าจำนวนบวก แต่ได้ค่าลบ
- สร้างเอ็กเซพชั่นสำหรับเงื่อนไข

- การสำหนดเอ็กเซ็พชั่นต้องสร้างคลาสที่มี parent จากคลาสชื่อ Exception
- การสร้างเอ็กเซ็พชั่นทำโดยการใช้งานคีย์เวิร์ด throw ร่วมกับอินสแตนซ์ของคลาสดังกล่าว

1.2. Explicitly Exception

รูปแบบ

```
throw Exception_Instance
```

เมื่อ

throw	คือคีย์เวิร์ดสำหรับทำให้มีการโยนเอ็กเซพชัน
Exception_Instance	คืออินสแตนซ์ที่เกิดจากคลาส Exception หรือคลาสใดๆที่สืบทอดมาจากคลาส Exception

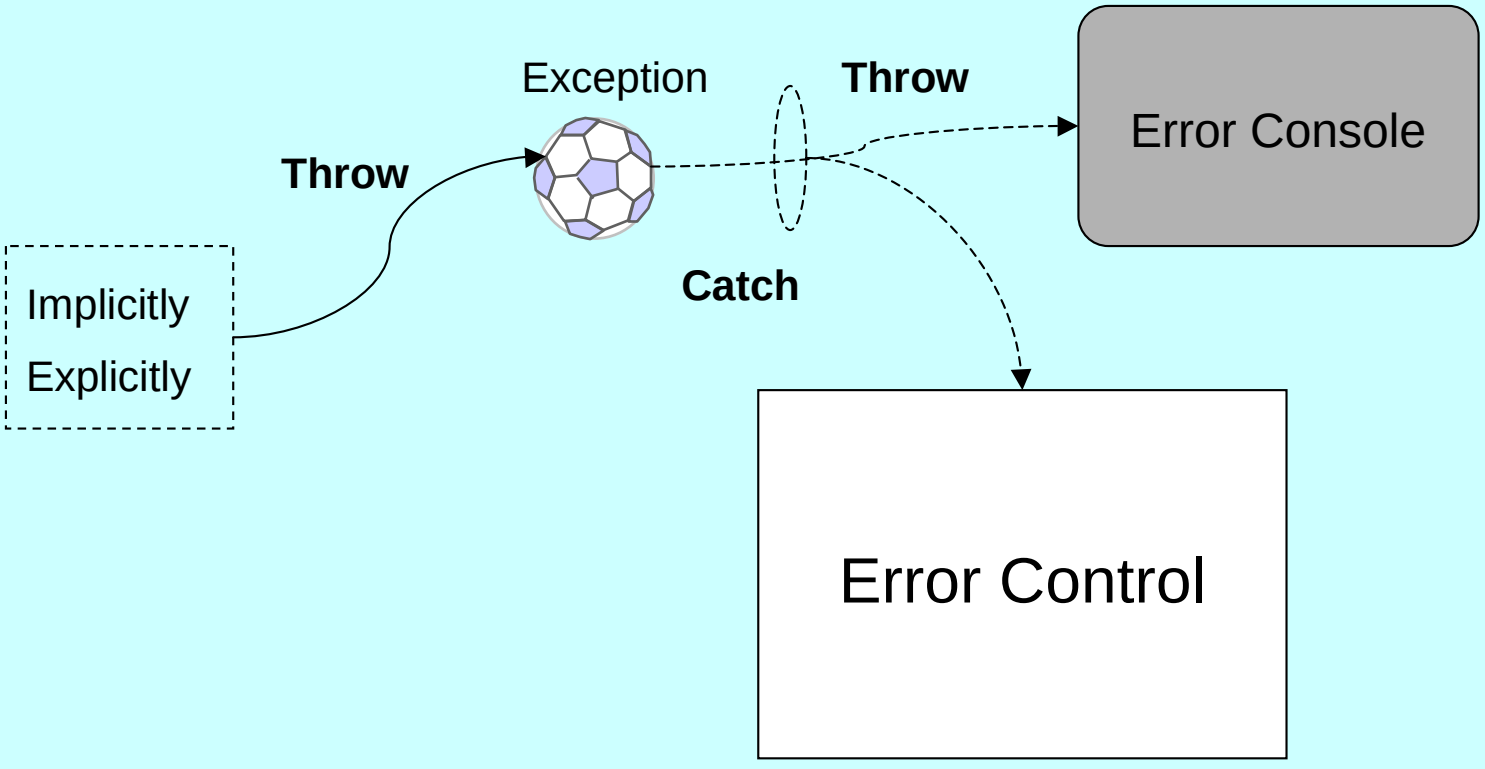
1.2. Explicitly Exception

```
class LowerBoundAge extend Exception  {      }
class CheckLowerAge
{    public static void main(String args[])
     {       try
             {            if (Integer.parseInt(args[0]) <= 0)
                                  throw (new LowerBoundAge());
                          else System.out.println("Legal Age");
             }
             catch (LowerBoundAge e)
             {            System.out.println("Illegal Age");                    }
     }
}
```

```
java CheckLowerAge 10
Legal Age
java CheckLowerAge -16
Illegal Age
```

2. Handle Exception

คือการดักจับ เอ็กเซพชั่นใดๆที่เกิดขึ้น โดยใช้กลุ่มคำสั่ง try block



2. Handle Exception

รูปแบบ

```
try   {  Implicitly_Exception / Explicitly_Exception  }
catch  ( Exception_Class1 )
{      Resolve_Statement1      }
catch  ( Exception_ClassN )
{      Resolve_StatementN      }
finally
{      Final_Statemet          }
```

2. Handle Exception

เมื่อ

try คือคำสั่งสำหรับดักฟังกลุ่มสเตจเม็นต์ที่คาดว่าจะเกิดเอ็กเซพชันขึ้น

Implicitly_Exception หรือ *Explicitly_Exception*

คือกลุ่มสเตจเม็นต์ที่อาจมีการโยนอินสแตนซ์ประเภทเอ็กเซพชัน

catch คือคำสั่งที่ดักจับอินสแตนซ์ใดๆ ใน Exception_Class

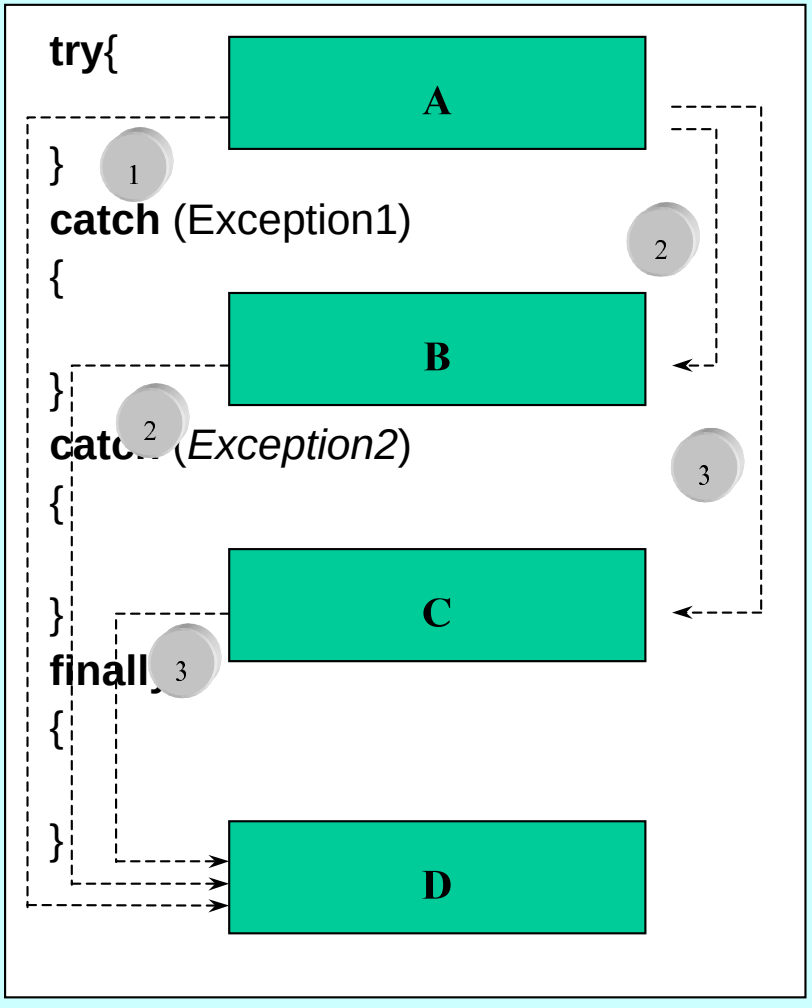
Exception_Class คือชื่อคลาสที่ตรงกับอินสแตนซ์ที่ต้องการดักจับ

Resolve_Statement คือกลุ่มสเตจเม็นต์เพื่อทำงาน

finally คือคำสั่งสำหรับกำหนดให้ทำสเตจเม็นต์ใน Final_Statement ทุกๆกรณี

Final_Statemet คือกลุ่มสเตจเม็นต์ที่ต้องการให้ทำงานไม่ว่าจะเกิดเอ็กเซพชันหรือไม่ก็ตาม

2. Handle Exception



2. Handling Exception

```
class Division1
{    public static void main(String args[])
     {       try
             {  int a = Integer.parseInt(args[0])/Integer.parseInt(args[1]);
                System.out.println(args[0] + "/" + args[1] + " = " + a);
             }
             catch (ArithmeticException e)
             {  System.out.println("Your division value must not equal 0");     }
     }
}
```

java Division1 4 5
4/5 = 0
java Division1 4 0
Your division value must not equal 0

2. Handling Exception

```
class Division2
{    public static void main(String args[])
     { try  {  int a = Integer.parseInt(args[0])/Integer.parseInt(args[1]);
                 System.out.println(args[0] + "/" + args[1] + " = " + a);
               }
       catch (ArithmeticException e)
       {       System.out.println("Your division value must not equal 0");     }
       catch (ArrayIndexOutOfBoundsException e)
       {  System.out.println("Usage: java Division2 FirstValue SecondValue");         }
     }
}
```

```
java Division2
Usage: java Division2 FirstValue SecondValue
java Division2 10
Usage: java Division2 FirstValue SecondValue
java Division2 10 5
10/5 = 2
```

2. Handling Exception

```
class Division3
{    public static void main(String args[])
     {       try
             {  int a = Integer.parseInt(args[0])/Integer.parseInt(args[1]);
                System.out.println(args[0] + "/" + args[1] + " = " + a);
             }
             catch (ArithmeticException e)
             {  System.out.println("Your division value must not equal 0");     }
             catch (ArrayIndexOutOfBoundsException e)
             {   System.out.println("Usage: java Division2 FirstValue
                                                  SecondValue");    }
             catch (NumberFormatException e)
             {   System.out.println("You must passed Integer Value");          }
             }
}
```

2. Handling Exception

```
java Division3 a b
You must passed Integer Value
java Division3 100 25
100/25 = 4
```

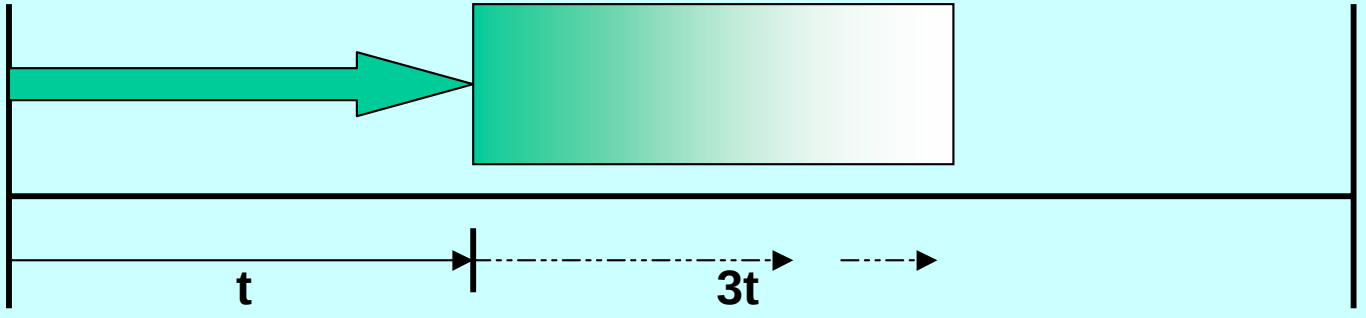
2. Handling Exception

```
class Division4
{    public static void main(String args[])
     {  try
        {  int a = Integer.parseInt(args[0])/Integer.parseInt(args[1]);
           System.out.println(args[0] + "/" + args[1] + " = " + a);
        }
        catch (Exception e)
        {     System.out.println("Uncompleted….");          }
        finally
        {     System.out.println("Finish Process");          }
     }
}
```

```
java Division4
Uncompleted
Finish Process
java Division4 256 2
256/2 = 128
```

โปรแกรมทำงานในหนึ่งโปรเซส

- กินเวลาในการทำงาน เมื่อโปรเซสมีการคำนวณที่สลับซับซ้อน
- กินเวลาในการทำงาน เมื่อโปรเซสทำงานติดต่อกับอุปกรณ์ภายนองที่ช้ากว่า
- ทำให้ผู้ใช้เข้าใจว่าโปรแกรมหยุดทำงานหรือแฮงก์ ทั้งที่จริงเป็นการรอ

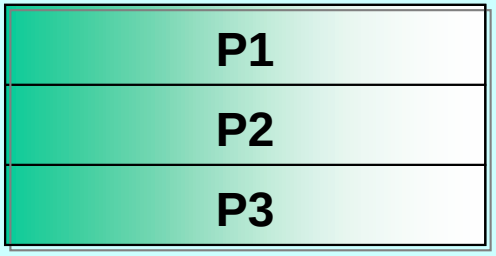


3. Threads

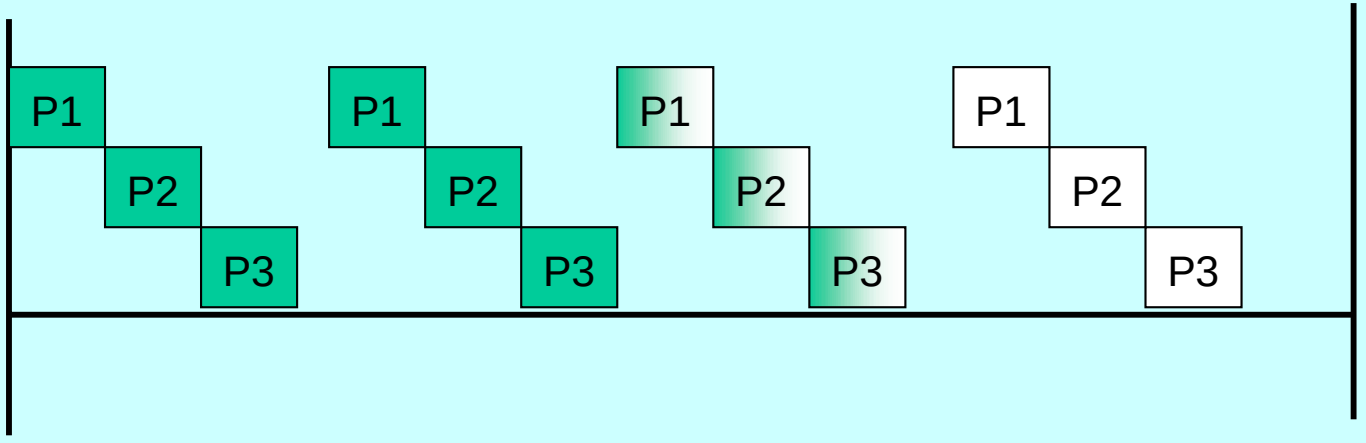
โปรแกรมทำงานในหลาย Thread

- แบ่งการทำงานออกเป็น Thread ย่อยๆ
- ให้เวลาในการประมวลผลของ CPU ในแต่ละ Thread แยกจากกัน
- ทำให้การทำงานไม่หยุดชะงักในเวลานาน เช่น Thread ที่คำนวณซับซ้อนยังคำนวณอยู่ ในขณะที่ Thread ในการติดต่อกับผู้ใช้ก็สามารถใช้งานได้
- ใช้สร้างโปรแกรมในลักษณะ Parallel Processing

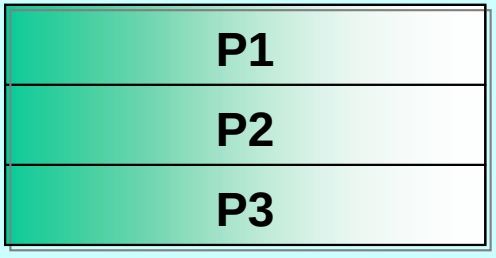
3. Threads



Three Threads in a program

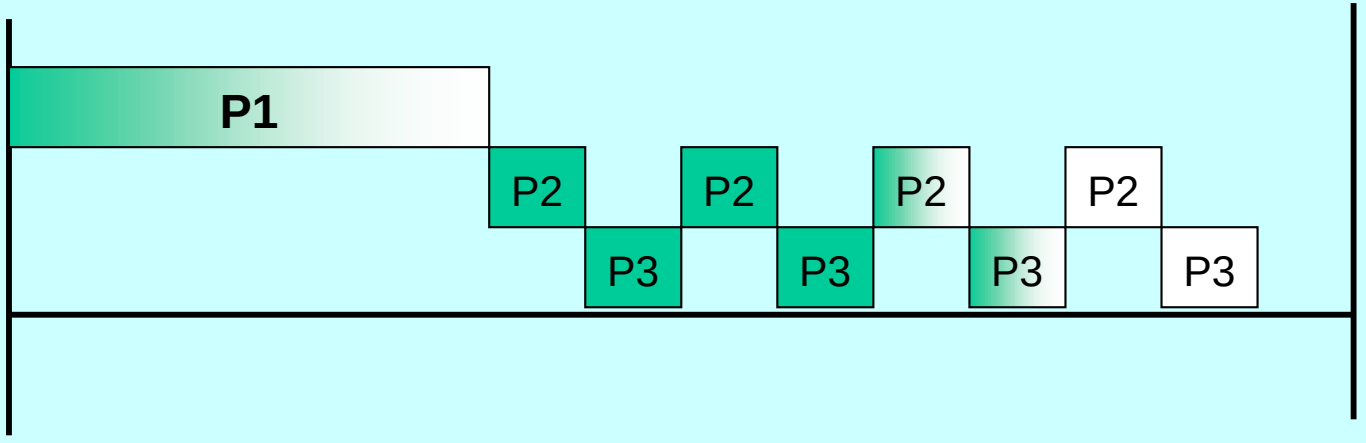


3. Threads



Three Threads in a program

P1 มี Priority สูงสุด P2, P3 มี Priority เท่ากัน



3. Threads

การสร้าง Thread

แบบที่ 1 สร้างคลาสสืบทอดจากคลาส Thread

รูปแบบ

```
class Dog extends Thread {
        public void run ( ) { … }
}

Dog d1 = new Dog();
```

3. Threads

การสร้าง Thread

แบบที่ 2 สร้างอินเตอร์เฟสคลาสจากคลาส Runnable
และใช้งานในคอนสตรักเตอร์ของคลาส Thread
รูปแบบ

```
class Cat implements Runnable {
        public void run ( ) { … }
}
Thread t1 = new Thread( new Cat());
```

3. Threads

เมธรอดใน Thread

ชื่อ	ความหมาย
start, stop	เริ่มทำงาน และหยุดการทำงาน
suspend	หยุดการทำงานชั่วคราว
resume	ทำงานต่อจากช่วงที่แบ้ว
getPriority	ดึงค่า priority อยู่ระว่าง 0-10
setPriority	ตั้งค่า priority
yield	
setName	ตั้งชื่อของ instance ใน Thread นั่นๆ

3. Threads

```
import java.io.*;
class thread5a {
   public static void main(String[] a) {
      ExtnOfThread t1 = new ExtnOfThread();      t1.start();
      Thread t2 = new Thread (new ImplOfRunnable());      t2.start();
   }
}
class ExtnOfThread extends Thread {
public void run() {
         System.out.println("extension of Thread running");
         setPriority( getPriority() +1 );
   }
}
class ImplOfRunnable implements Runnable {
public void run() {
         System.out.println("Implementation of Runnable running");
   }
}
```

3. Threads

3. Threads

```
class Pear implements Runnable {
   String name;
   Pear(String s) { name=s; }
   public void run() {   System.out.println( name );   }
}
public class thread5b {
   public static void main(String[] a) throws InterruptedException{
      Pear p1 = new Pear("comice");
      Pear p2 = new Pear("belvedere");
      Pear p3 = new Pear("brimley");
      Thread t1 = new Thread(p1, "comice");
      new Thread(p2, "comice").start();
      Thread t3 = new Thread(p3, "comice");
      t1.start();      t3.start();
    for(int i=0; i<10; i++ ) {
      t1.sleep( (int)(Math.random()*100) );
      t3.sleep( (int)(Math.random()*100) );
      t1.stop();      t1.start();    }
   }
}
```

3. Threads

3. Threads

```
class testRange extends Thread {
   static long possPrime;
   long from, to;
     testRange(long argpossPrime, int argFrom) {
       possPrime=argpossPrime;
       if (argFrom==0) from=2; else from=argFrom;
       to=argFrom+99;
   }
    public void run() {
      for (long i=from; i<=to && i<possPrime; i++) {
         if (possPrime%i == 0) {  // i divides possPrime exactly
               System.out.println("factor "+i+" found by thread "+getName());
               this.stop();        }
         yield();
      }
   }
}
```

มีต่อ

3. Threads

```
public class testPrime {
   public static void main(String s[]) {
        if (s.length!=1)
            System.out.println("invoke like this: java testPrime 1027");
        long possPrime = Long.parseLong(s[0]);
        int centuries = (int)(possPrime/100) +1;
         for(int i=0;i<centuries;i++) {
            new testRange(possPrime, i*100).start();
            System.out.println("starting a thread: ");       }
    }
}
```

3. Threads

```
java testPrime 2048
factor 2 found by thread Thread-4
factor 512 found by thread Thread-9
factor 10242 found by thread Thread-14
factor 128 found by thread Thread-5
factor 256 found by thread Thread-6
```