Part 10

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

Contents

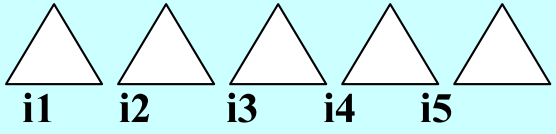
1. Array

แถวลำดับ (Array)

- เก็บข้อมูลในรูปแบบของชุดข้อมูลสามารถทำได้การกำหนดตัวแปรประเภทเดียวกันหลายๆตัว เช่น

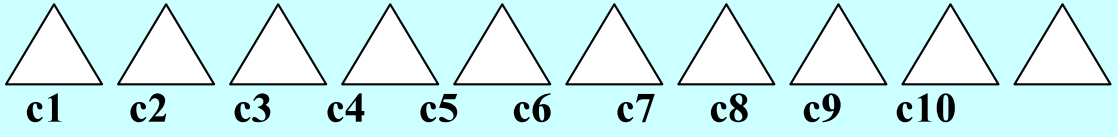
int i1, i2, i3, i4, i5;

คือการเก็บชุดข้อมูลตัวเลขจำนวนเต็ม 5 ตัว



char c1, c2, c3, c4, c5, c6, c7, c8, c9, c10;

คือการเก็บชุดข้อมูลตัวอักษร 10 ตัว



1. Array

- เก็บข้อมูลรูปแบบของชุดข้อมูลที่เป็นประเภทเดียวกัน เช่น ชุดข้อมูลตัวเลขจำนวนเต็ม ชุดข้อมูลตัวอักษร หรือชุดข้อมูลอ้างถึง
- ใน Array หนึ่งชุดสามารถอ้างถึง ใช้หมายเลขอินเด็กซ์(index) ในการอ้างตำแหน่งของการเก็บข้อมูล
- เมื่อกำหนดให้ Array มีข้อมูล n ตัว มีหมายเลขอินเด็กซ์ชี้ข้อมูลตั้งแต่ 0 - n-1 เช่น เมื่อชุดข้อมูลมี 5 ตัว ในชุดนั้นสามารถอ้างตำแหน่งได้ตั้งแต่ 0 ถึง 4
- แต่ละตัวใน Array ที่อ้างด้วยอินเด็กซ์ คือหนึ่งอิลิเม็นต์ (Element)
- การกำหนด Array มี 2 ลักษณะคือ
  - Array เก็บข้อมูลชนิด Primitive Data Type
  - Array เก็บข้อมูล Reference Data Type

2. Array(Primitive)

Array **เก็บข้อมูลชนิด** Primitive Data Type

รูปแบบ

ขั้นที่ 1 Data_Type Arrag_Name[]

ขั้นที่ 2 Arrag_Name **= new** Data_Type**[** Array_Size **]**

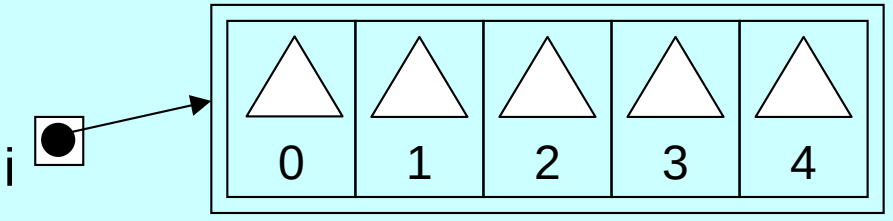
เมื่อ

Arrag_Name	คือชื่ออะเรย์ที่ใช้
Data_Type	คีย์เวิร์ดของชนิดข้อมูล Primitive เช่น int, char, double, ...
Array_Size	ขนาดของอิลิเม็นต์ที่ต้องการ
[]	แสดงการกำหนด Array

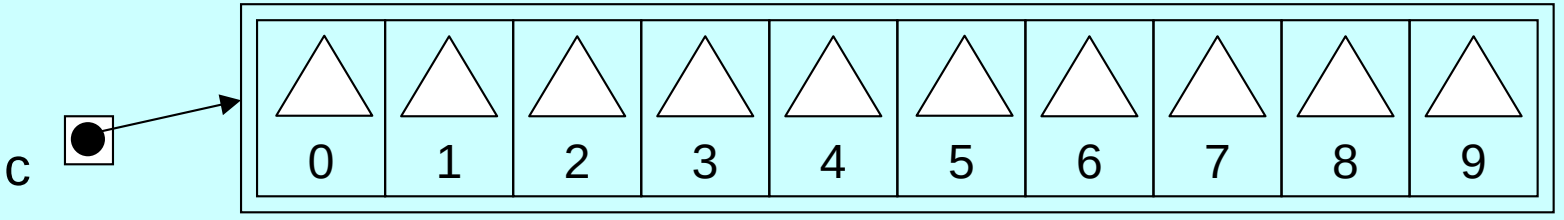
2. Array(Primitive)

ตัวอย่าง

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




```
char c[];
c = new char[10];
```



2. Array(Primitive)

การกำหนด Array สามารถทำได้ในหนึ่งสเตจเม็นต์

รูปแบบ

Data_Type Arrag_Name[ ] = new DataType[Array_Size]

Data_Type [ ]Arrag_Name = new DataType[Array_Size]

ตัวอย่าง

```
int i[] i = new int[5];
char []c = new char[10];
```

2. Array(Primitive)

การกำหนดค่าเริ่มต้นให้กับแต่ละอิลิเม็นต์ใน Array

รูปแบบ

Data_Type Array_Name [ ] = { Value_List }

Data_Type [ ]Array_Name = { Value_List }

เมื่อ

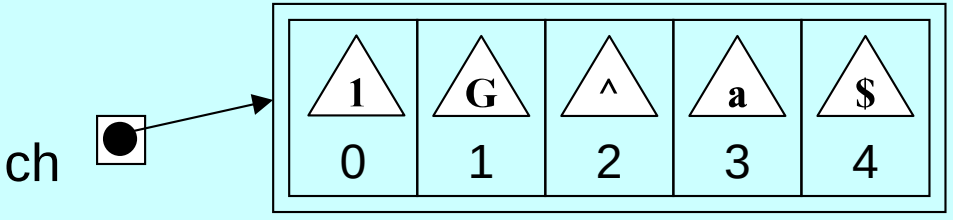
Value_List คือข้อมูลค่าคงที่ที่ตรงกับชนิดข้อมูล Data_Type

ตัวอย่าง

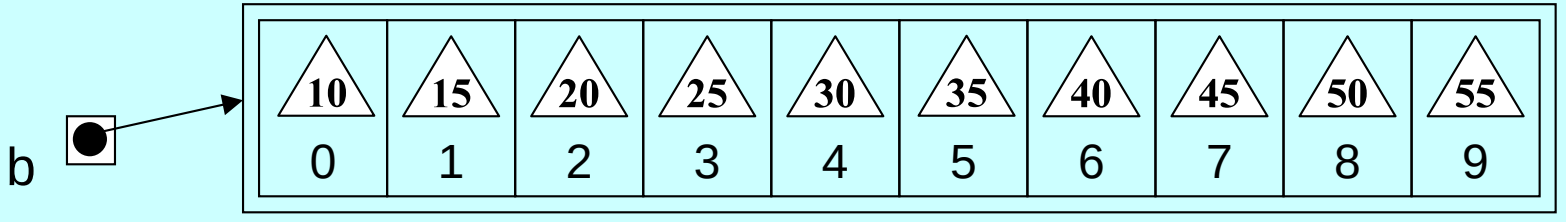
```
char ch[ ] = { '1', 'G', '^', 'a', '$' };

byte b[ ] = { 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 };
```

2. Array(Primitive)

char ch[ ] = { '1', 'G', '^', 'a', '$' };



byte b[ ] = { 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 };



2. Array(Primitive)

การใช้งานข้อมูลในอิลิเม็นต์ของ Array

รูปแบบ

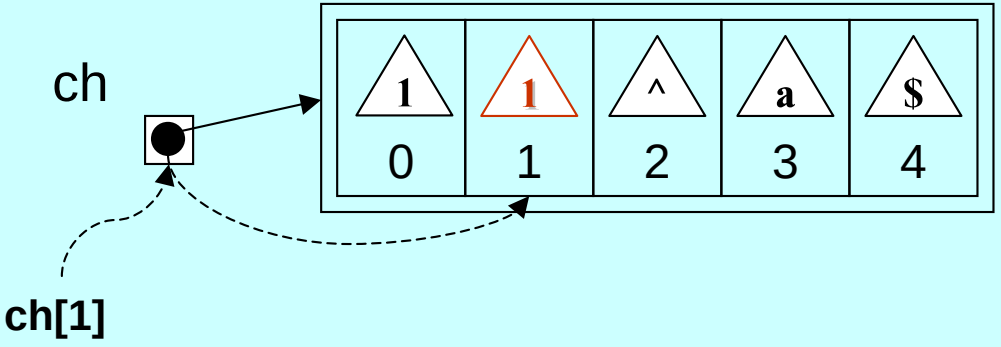
Array_Name [index]

เมื่อ

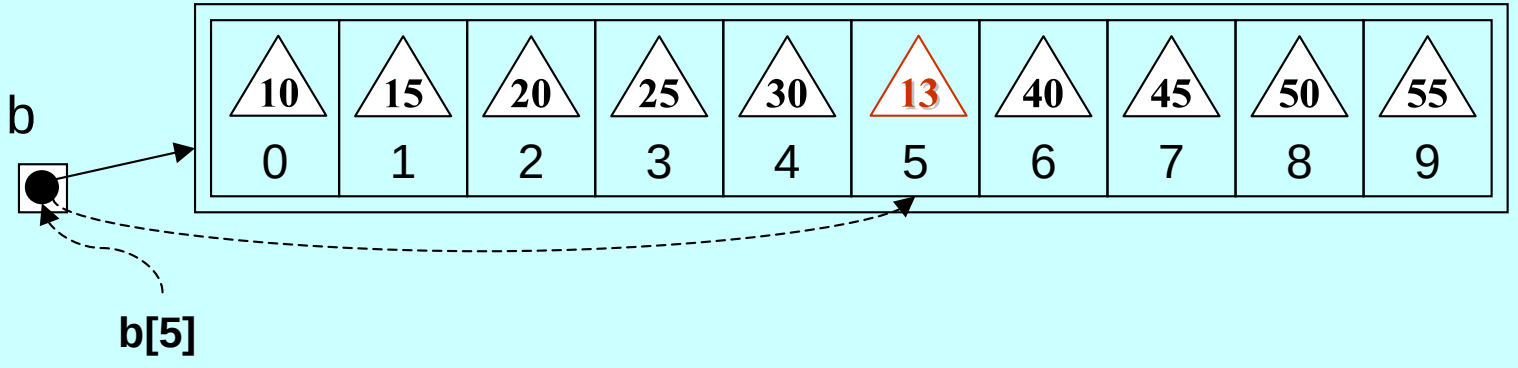
index คือตัวเลขที่อ้างถึงอิลิเม็นต์ใน Array โดยเริ่มที่ 0

2. Array(Primitive)

ch[1] = ch[0]



b[5] = 13



2. Array(Primitive)

การหาจำนวนอิลิเม็นต์ใน Array

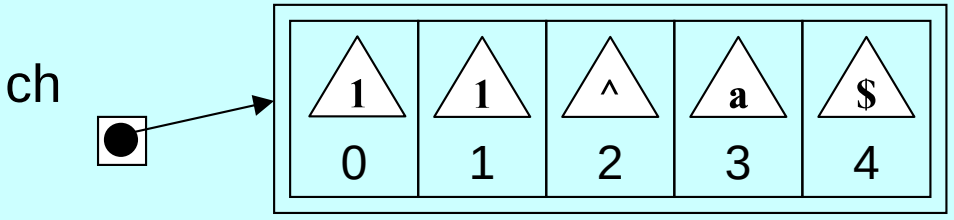
รูปแบบ

Array_Name.length

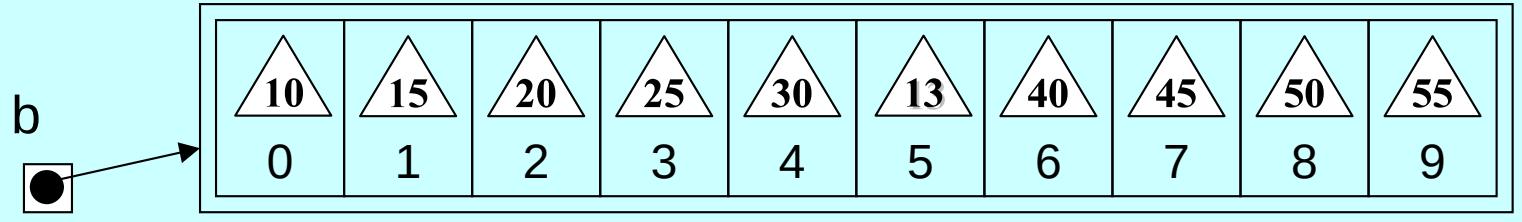
ตัวอย่าง

length คือ Data ในคุณสมบัติของ Array ที่ใช้บอกขนาดของอิลิเม็นต์ในอะเรย์

2. Array(Primitive)



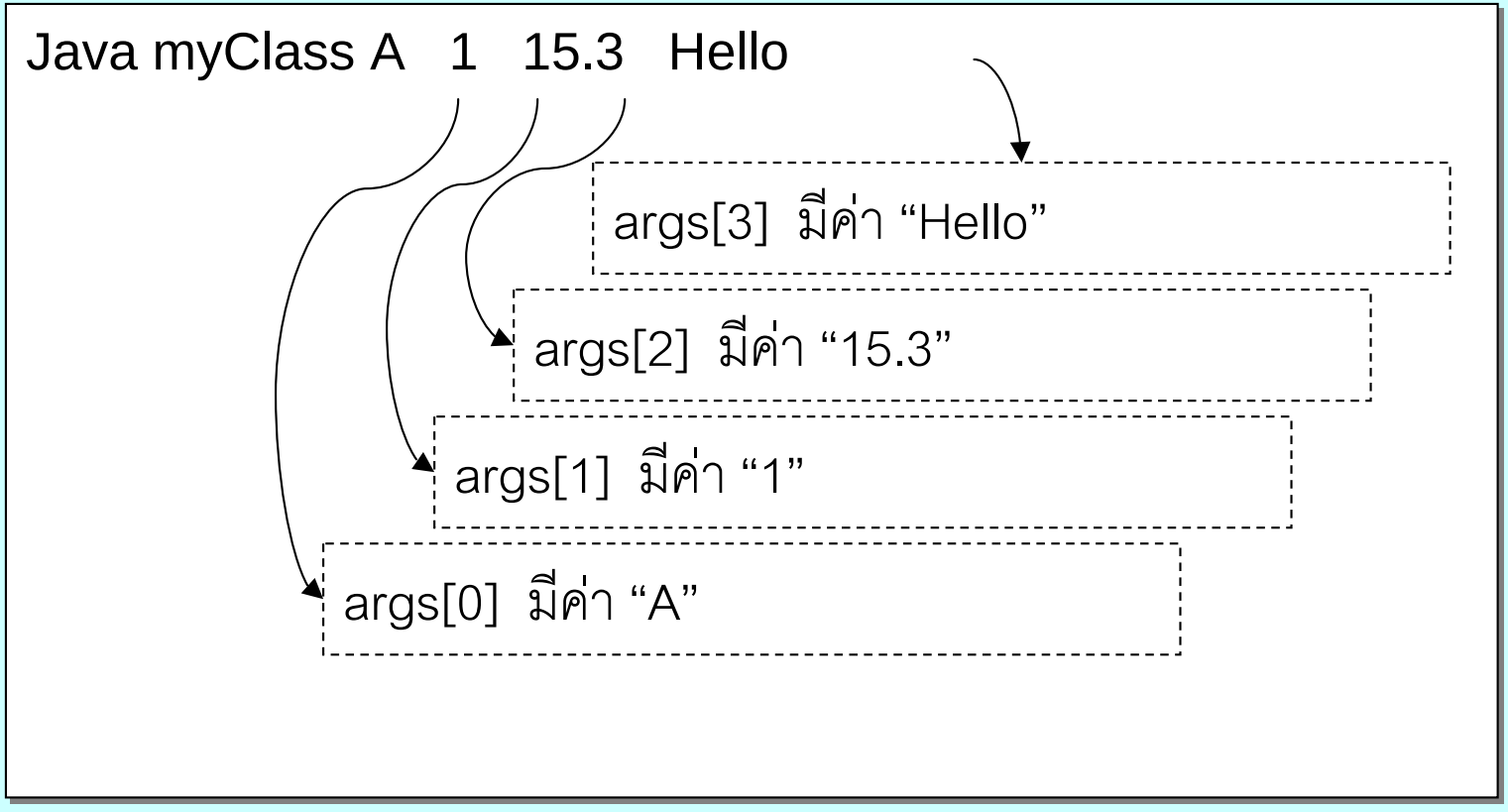
ch.length มีค่าเท่ากับ 5



b.length มีค่าเท่ากับ 10

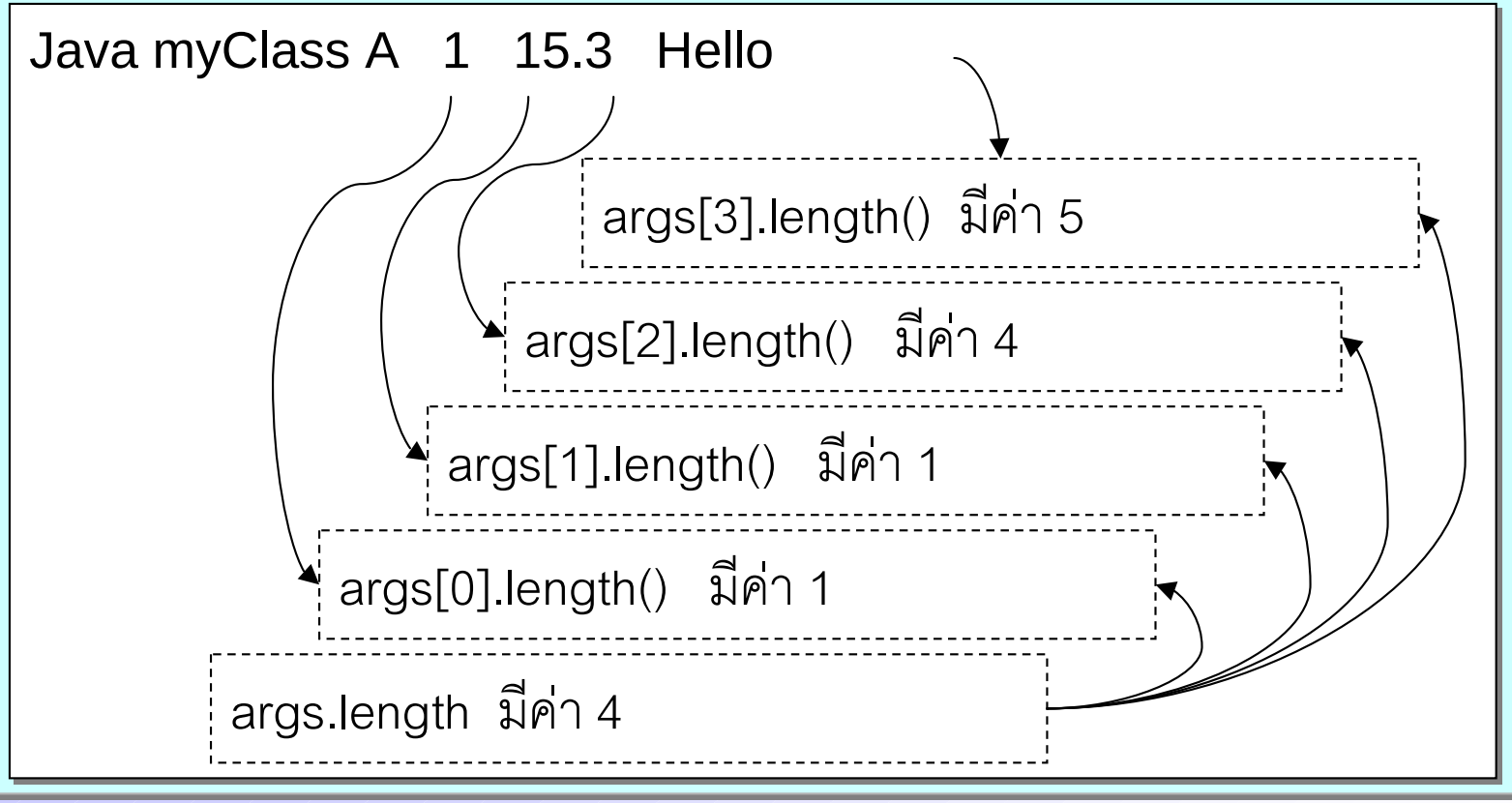
2. Array(Primitive)

```
class myClass {
public static void main(String args[] ) { … }
}
```



2. Array(Primitive)

length ของ Array vs. length() ของ String



3. Array(Primitive)

```
class SoundTrack
{    static public float track_time [ ] =
            { 3.24f, 3.28f, 2.58f, 4.01f, 3.17f, 3.59f, 2.24f, 3.22f };
     static float getLongTrack( )
     {       float max = 0.0f;
             for( int i = 0; i < track_time.length; i++)
                        if (max<track_time[i])
                                    max = track_time[i];
             return max;
     }
     public static void main(String arg[ ])
     {       System.out.print("Long Track Time is " + getLongTrack( ) );      }
}
```

java SoundTrack

Long Track Time is 4.01

3. Array(Reference)

Array **เก็บข้อมูลชนิด** Reference Data Type

คือการกำหนด Array เพื่อเก็บตัวแปรอ้างถึงไปยัง Instance

รูปแบบ

ขั้นที่ 1 Class_Name Arrag_Name[]

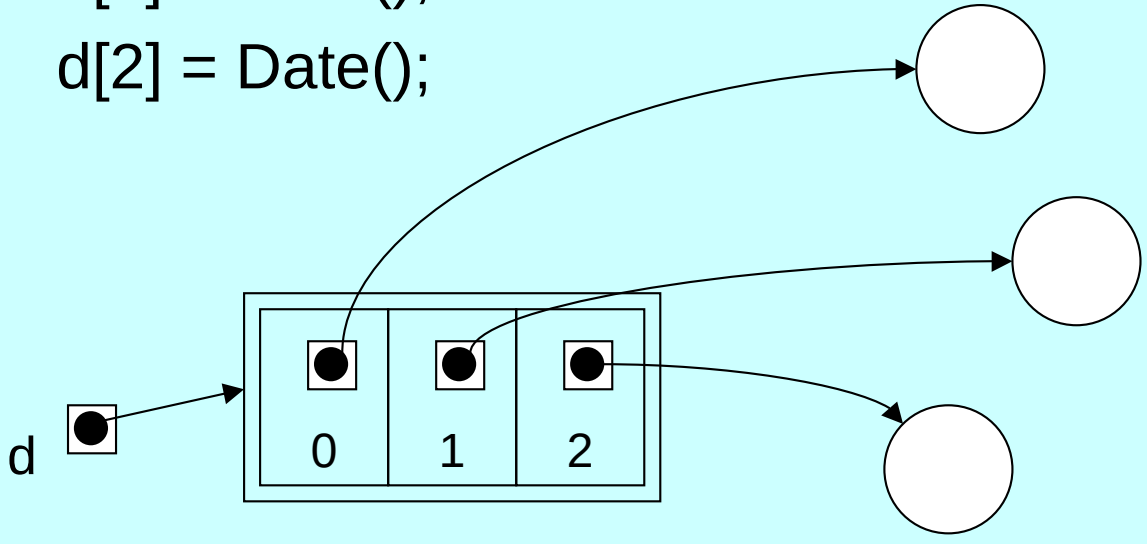
ขั้นที่ 2 Arrag_Name **= new** Class_Name**[** Array_Size **]**

ขั้นที่ 3 Arrag_Name[0] **= new** Constructor()
Arrag_Name[0] **= new** Constructor()
…….
Arrag_Name[Array_Size-1] **= new** Constructor()

3. Array(Reference)

ตัวอย่าง

```
Date d [ ];
d = new Date[3]
d[0] = Date(); // ไม่จำเป็นต้องสร้าง Instance ทุกๆอิลิเม็นต์
d[1] = Date();
d[2] = Date();
```



3. Array(Reference)

ความแตกต่างระหว่าง Array ที่สร้างจาก Primitive และ Reference Data Type

	Primitive	Reference
การประกาศชื่อ	✓	✓
จัดสรรพื้นที่ให้กับอะเรย์	✓	✓
สร้างอินสแตนซ์ให้กับอิลิเม็นต์		✓

3. Array(Reference)

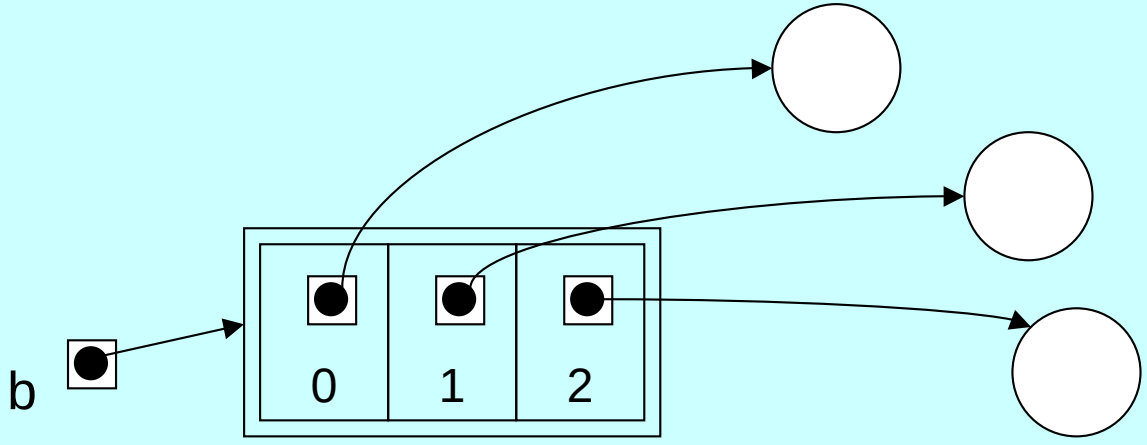
การกำหนดค่าเริ่มต้นให้กับแต่ละอิลิเม็นต์ใน Array

รูปแบบ

Class_Name Array_Name[] = { new Constructor ( ), … }

ตัวอย่าง

Byte b [ ] = { new Byte(1), new Byte(100), new Byte(10) };



3. Array(Reference)

การใช้งาน

การใช้งานดาต้าในอินสแตนซ์ของแต่ละอิลิเม็นต์

Array_Name [Index].Data_Member

การใช้งานเมธรอดในอินสแตนซ์ของแต่ละอิลิเม็นต์

Array_Name [Index].Method_Member (*Parameter_List*)

ตัวอย่าง

```
1: Byte b [ ] = { new Byte(1), new Byte(100), new Byte(10) };
2: d[2].byteValue( );
3: d[0].MAXVALUE;
```

3. Array(Reference)

```
class CDAlbum
{    static Song track_list[ ] = { new Song("I do It for You ", 3.38f),
     new Song("So Much In Love ", 3.20f), new Song("Unbreak My Heart", 4.45f),
     new Song("The Power of Love", 4.17f), new Song("Grovy Kind of Love",
     3.35f) };
     public static void main(String arg[ ])
     {       float total = 0.0f;
             System.out.println("\tSong Name\t\t\tRecord Time");
             System.out.println("\t=========\t\t\t==========");
             for (int i=0; i < track_list.length; i++)
             {           System.out.println("\t" + track_list[i].song_name + "\t\t" +
                                 track_list[i].record_time);
                         total += track_list[i].record_time;
             }
             System.out.println("\t\t\t\t\tTotal = " + total);
     }
}
```

มีต่อ

3. Array(Reference)

```
class Song
{    String          song_name = "";
     float  record_time = 0.0f;
     Song(String name, float time)
     {       song_name = name; record_time = time;       }
}
```

```
java CDAlbum
 Song Name                Record Time
 =========                ===========
 I do It for You.         3.38
 So Much In Love          3.2
 Unbreak My Heart         4.45
 The Power of Love        4.17
 Grovy Kind of Love       3.35
                          Total = 18.55
```

4. Abstract Class

แอบเตรสคลาส

- แอบแตรสคลาสใช้เพื่อให้เป็นรูปแบบ เช่น รูปแบบ Data และ Method (ไม่มีสเตจเม็นต์อยู่ภายใน
- แอบเตรสคลาสไม่สามารถนำไปสร้างอินสแตนซ์ได้จริง
- แอบเตรสคลาสสามารถใช้กำหนดตัวแปรอ้างถึงได้เท่านั้น (Reference Variable) ซึ่งมีคุณสมบัติ Polymorphism
- แอบเตรสคลาสที่ถูกนำไปสร้างคลาสที่สามารถใช้งานได้จริง โดยใช้วิธีการสืบทอดด้วยคีย์เวิร์ด extends

4. Abstract Class

รูปแบบ

```
abstract class Class_Name
{   abstract Data_Define;
    abstract Method_Head;
}
```

เมื่อ

abstract คือคีย์เวิร์ดที่วางไว้หน้าคลาส เพื่อทำให้คลาสมีคุณสมบัติเป็นแอบเตรสคลาส

data_define คือการกำหนด data

Method_Head คือการกำหนดชื่อเมธรอด รูปแบบอาร์กิวเม็นต์ ชนิดการคืนกลับ โดยไม่มีสเตจเม็นอยู่ภายใน

4. Abstract Class

```
abstract class AbstractClass
{    public abstract int SumNumber(int i, int j);              }
class ConcreteClass extends AbstractClass
{    public int SumNumber(int i, int j)
     {       System.out.println(i+j);
             return i+j;
     }
}
class UseConcrete
{    public static void main(String arg[])
     {
     //      AbstractClass obj1 = new AbstractClass();
             ConcreteClass obj2 = new ConcreteClass();
             obj2.SumNumber(900, 99);
     }
}
```

```
java UseConcrete
999
```

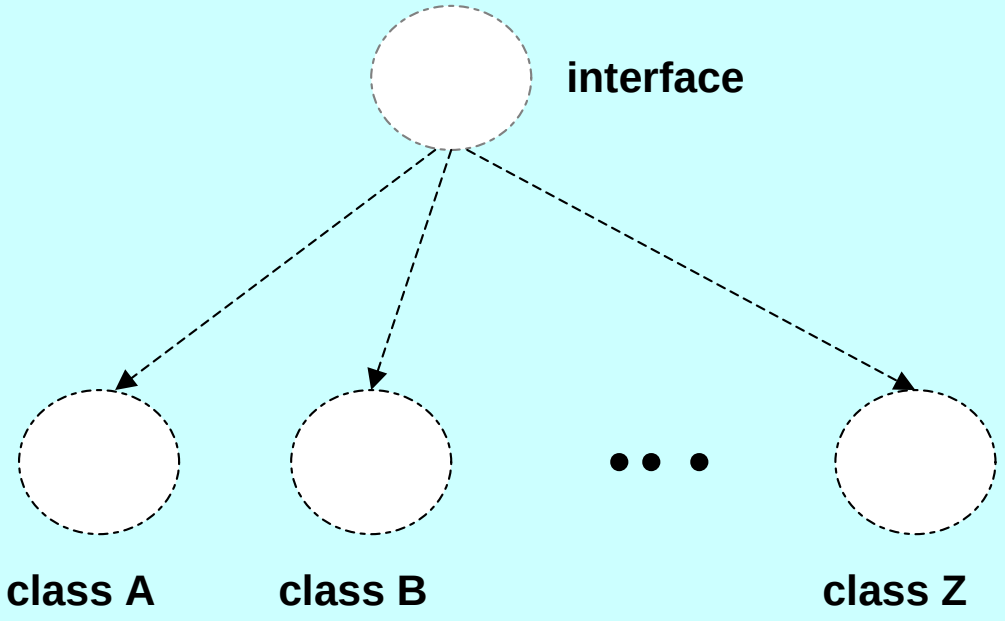
5. Interfaces

อินเตอร์เฟสซ์ (Interfaces)

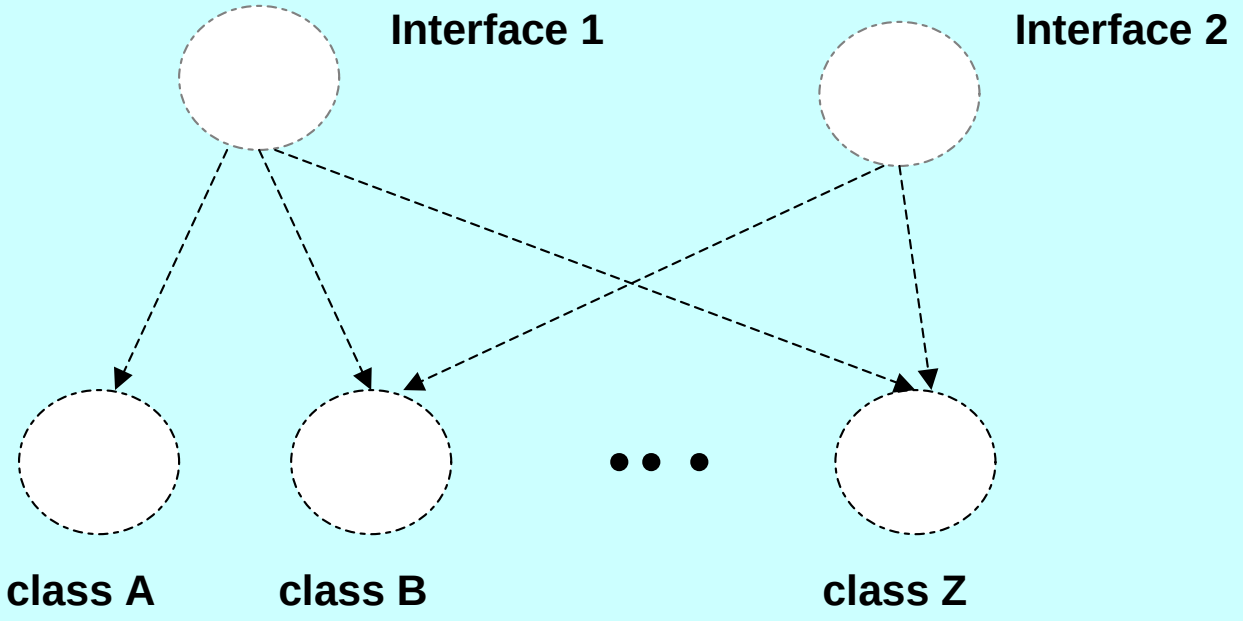
- เนื่องจากการออกแบบคลาสส่วนใหญ่มักจะมีเมธรอดที่คล้ายกัน จาวาสามารถกำหนด อินเตอร์เฟสไว้เป็นมาตรฐานเพื่อนำไปสร้างคลาส
- Data Member จะมีระดับการเข้าถึงเป็นแบบ final และ static ถึงแม้จะไม่ระบุ
- Method Member จะมีระดับการเข้าถึงเป็นแบบ public ถึงแม้จะไม่ระบุ
- การใช้งานอินเตอร์เฟสต้องมีการสร้างคลาส ด้วยการใช้คีย์เวิร์ด implements
- คลาสที่ได้สามารถนำไปสร้าง instance ได้ ในขณะที่อินเตอร์เฟสสร้างไม่ได้
- เหมาะสำหรับเป็นต้นแบบในการสร้างคลาสที่มีลักษณะคล้ายกัน เช่น เมธรอดแบบเดียวกัน

(ในบทเรียน จะมีการใช้งาน Interface ในส่วนของการสร้าง GUI)

5. Interfaces



5. Interfaces



5. Interfaces

รูปแบบ

```
interface Interface_Name
{  Data_Member
   Method_Head
}
```

เมื่อ

Interface_Name	คือชื่ออินเตอร์เฟส
Data_Member	คือดาต้าที่ต้องการระบุให้กับอินเตอร์เฟส
Method_Head	คือชื่อเมธรอด อาร์กิวเม็นต์ ชนิดคืนกลับ

5. Interfaces

รูปแบบ

```
class Class_Name implements Interface_Name
    , Interface_Name
{   Data_Member
    Method_Member
}
```

เมื่อ

implements คือคีย์เวิร์ดในการสร้างคลาสที่มีที่มาจากอินเตอร์เฟส

5. Interfaces

```
interface InterfaceClass
{
    String info = "INTERFACE CLASS";
    public abstract void printInfo();
}
class ConcreteClass implements InterfaceClass
{   public void printInfo()
    {       System.out.println(info);     }
}
class UseConcrete
{   public static void main(String arg[])
    {
    //      InterfaceClass obj1 = new InterfaceClass();
            ConcreteClass obj2 = new ConcreteClass();
            obj2.printInfo();
    }
}
```

```
java UseConcrete
INTERFACE CLASS
```