Holding your object

Vittayasak Rujivorakul

Lecture 7-2

Holding your object

Vittayasak Rujivorakul

Lecture 7-2

Arrays ในจาวา

- ง่ายในการเข้าถึง (access) ทำให้ทำงานอย่างมีประสิทธิภาพ
- เป็นที่เก็บที่เหมาะสมในกรณีที่รู้ขนาดแน่นอน (fixed size container) มี field length ใช้บอกขนาดของ array
- จาวาตรวจสอบ index ของ array เสมอเพื่อไม่ให้มีการทำงานนอกเขต
- เป็น object ที่ถูกสร้างใน heap (เช่นเดียวกับ object อื่นโดยใช้คำสั่ง new)
- สามารถ return Array จาก method ได้เช่นเดียวกันกับ object อื่น การจัดการต่าง ๆ ในแง่การส่ง/รับจาก method จึงง่าย
- array ใช้เก็บได้ทั้ง object และ primitive โดยที่
  - array ของ object แต่ละ element จะเก็บ reference ที่ใช้อ้างถึง object
  - array ของ primitive แต่ละ element จะเก็บค่าของ primitive นั้น

Collection

- Collection class ในจาวา (Vector, Stack, Hashtable) ใช้เก็บ object ได้อย่างเดียว
- การใช้ collection class ในจาวา จะทำให้เสียข้อมูลของชนิดของ object ไป (ใช้ class Object ซึ่งเป็น root ของทุก class ในการเก็บ)
  - ทำให้ไม่รู้ชนิดของ object ที่เก็บภายใน และต้องทำ casting ให้เป็นชนิดเดิมก่อนจะใช้งานได้
  - ทำให้สามารถเก็บ object หลายชนิดปนกัน (มีทั้งข้อดีและข้อเสีย)
- ในกรณีต้องการเก็บค่าของ primitive ต้องอาศัยกลุ่มของ class ที่เรียกว่า wrapper เช่น Integer, Double, Boolean, etc.
- Collection class สามารถ resize ได้โดยอัตโนมัติ

ข้อด้อยของการไม่รู้ชนิด

```
import java.util.*;
class Cat {
   private int catNumber;
   Cat (int i) { catNumber = i; }
   void print( ) {
      System.out.println("Cat #"+catNumber);
   }
}
class Dog {
 private int dogNumber;
   Cat (int i) { dogNumber = i; }
   void print( ) {
      System.out.println("Dog #"+dogNumber);
   }
}
```

```
public class CatAndDogs {
   public static void main(String[ ] args) {
      int i;
      Vector cats = new Vector( );
      for(i = 0; i < 7; i++)
        cats.addElement(new Cat(i));
      cats.addElement(new Dog(7));

      for(i = 0; i < cats.size(); i++) {
        if (cats.elementAt(i)  instanceof  Cat)
          ((Cat)cats.elementAt(i)).print( );
      }
    }
}
```

บางกรณี (e.g.,String) อาจใช้งานได้

```
import java.util.*;
class Cat {
   private int catNumber;
   Cat (int i) { catNumber = i; }
   public String toString( ) {
      return "Cat #"+catNumber;
   }
}
class Dog {
 private int dogNumber;
   Cat (int i) { dogNumber = i; }
   public String toString( ) {
      return "Dog #"+dogNumber;
   }
}
```

```
public class WorksOh {
   public static void main(String[ ] args) {
      int i;
      Vector cats = new Vector( );
      for(i = 0; i < 3; i++)
        cats.addElement(new Cat(i));
      cats.addElement(new Dog(3));
      for(i = 0; i < cats.size(); i++) {
        // no need for casting
       System.out.println(" "+cats.elementAt(i);
      }
    }
}
```


การสร้าง Vector ให้รู้จักชนิด

```
import java.util.*;
class Cat {
   private int catNumber;
   Cat (int i) { catNumber = i; }
   void print( ) {
      System.out.println("Cat #"+catNumber);
   }
}
class CatTrap {
   static void gotha(Cat c) {
      c.pirnt( );
   }
}
```

```
public class CatVector {
   private Vector v = new Vector( );
   public void addElement(Cat c) {
      v.addElement(c);
   }
   public Cat elementAt(int index) {
      return (Cat)v.elementAt(index);
   }
   public static void main(String [] args) {
      CatVector cats = new CatVector();
      for(int i=0; i < 7; i++)
        cats.addElement(new Cat(i));
      for(int i = 0; i < cats.size(); i++)
        CatTrap.gotcha(cats.elementAt(i));
    }
}
```


Enumerators (iterators)

- เป็น object ที่ใช้ move ใน sequence ของ object และเลือก object แต่ละอันใน sequence ได้โดยที่ไม่ต้องรู้ว่า structure ของ sequence นั้นเป็นอย่างไร (light-weight object)
- การสร้าง
  - เรียก method element ( ) ใน Collection เพื่อสร้าง Enumeration หากเรียก nextElement ( ) method ของ Enumeration นี้จะได้ element ตัวแรกใน sequence
  - เรียก nextElement ( ) เพื่อทำการเรียก next object ใน sequence
  - เรียก method hasMoreElement ( ) เพื่อตรวจสอบว่ายังมี object ใน sequence

Types of Collections

- Vector : collection ของ object ที่ไม่รู้ขนาด
  - addElement (Object), elementAt (int), elements ( )
- BitSet : Vector ของ bits
  - set (int), clear (int), get (int)
- Stack : LIFO vector
  - push (Object), pop ( ), empty ( )
- Hashtable : association ของตัวเลขกับ object
  - containsKey ( ), get (int)

Enumerators Revisted

```
import java.util.*;
class Cat {
   private int catNumber;
   Cat (int i) { catNumber = i; }
   void print( ) {
      System.out.println("Cat #"+catNumber);
   }
}
class Dog {
 private int dogNumber;
   Cat (int i) { dogNumber = i; }
   void print( ) {
      System.out.println("Dog #"+dogNumber);
   }
}
```

```
Class PrintData {
   static void print(Enumeration e) {
      while(e.hasMoreElements())
        System.out.println
          (e.nexElement().toString());
   }
}
class Enumerator2 {
   public static void main( String [] args){
     Vector v = new Vector();
     for(int i=0; i<5; i++)
        v.addElement(new Cat.(I));
     for(int i=0; i<5; i++)
        h.put(new Integer(i), new Dog(i));
     PrintData.print(v.elements());
     PrintData.print(h.elements());
   }
}
```


Utilities ใน Collection class

enumeration(Collection)	สร้าง Enumeration สำหรับ arg
max(Collection) **min(Collection)**	ให้ element ที่มีค่าสูงสุด หรือ ต่ำสุดใน arg โดยใช้ comparison ของ object ใน Collection
max(Collection,Comparator) **min(Collection,Comparator)**	ให้ element ที่มีค่าสูงสุด หรือ ต่ำสุดใน arg โดยใช้ Comparator
nCopies(int n,Object o)	คืนค่า immutable List ขนาด n ชี้โดย o
subList(List, int min, int max)	คืนค่า List ย่อยระบุโดย List ที่มี index เริ่มต้นชี้โดย min และสิ้นสุดที่ index ของ max

การทำให้ Collection หรือ Map อ่านได้อย่างเดียว

```
/ / create a read-only version of a Collection of map
import java. util. * ;
public class ReadOnly    {
    public static void main (String [ ] args)   {
          Collection c  =  new ArrayList ( ) ;
          Collection1. fill ( c ) ;  / / insert data
          c = Collections . unmodifiableCollection ( c ) ;
          Collection1. print ( c ) ;
          / / c.add ("one") ;   / / Can' t change it
          / * can do the same with List, Set or Map  *  /
    }
}
```


การ Synchronize Collection หรือ Map

```
/ /  Synchronization
/ /  a way to synchronize an entire container
import java. Util. * ;
public class Synchronization   {
    public static void main (String [ ] args)    {
          Collection  c  =  Collection. synchronizedCollection (
                          new ArrayList ( ) ) ;
          List list  =  Collection. synchronizedList  (
                          new ArrayList ()) ;
          Set s  =  Collection. synchronizedSet (new HashSet  ( ) ) ;
          Map m  =  Collection. synchronizeMap (new HashMap ( ) ) ;
    }
}
```