

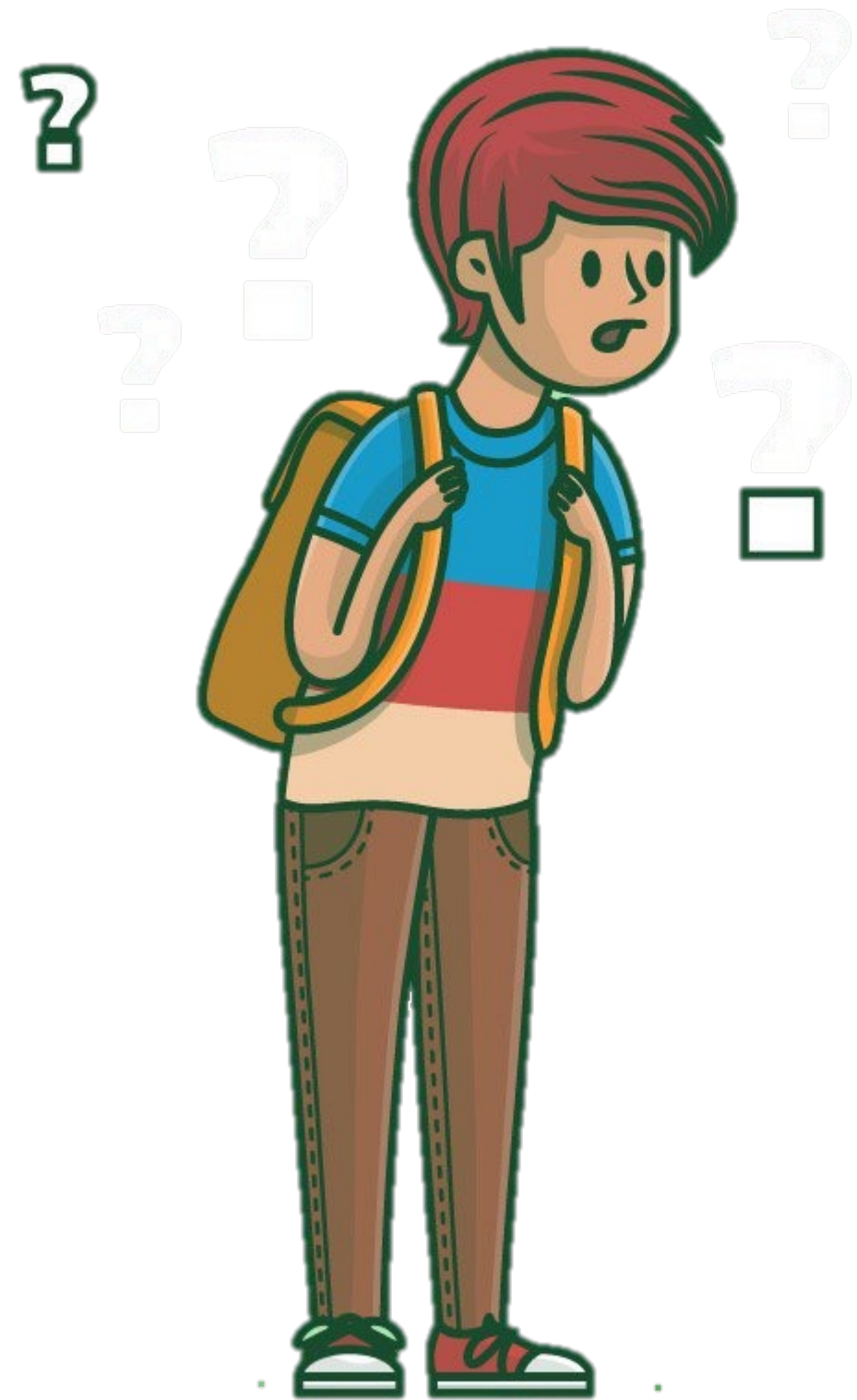
บทที่ 8: อาร์เรย์และคอลเล็กชัน (Arrays & Collections)

บรรยายโดย ผศ.ดร.ธราวิเชษฐ์ ธิติจรูญโรจน์

คณะเทคโนโลยีสารสนเทศ

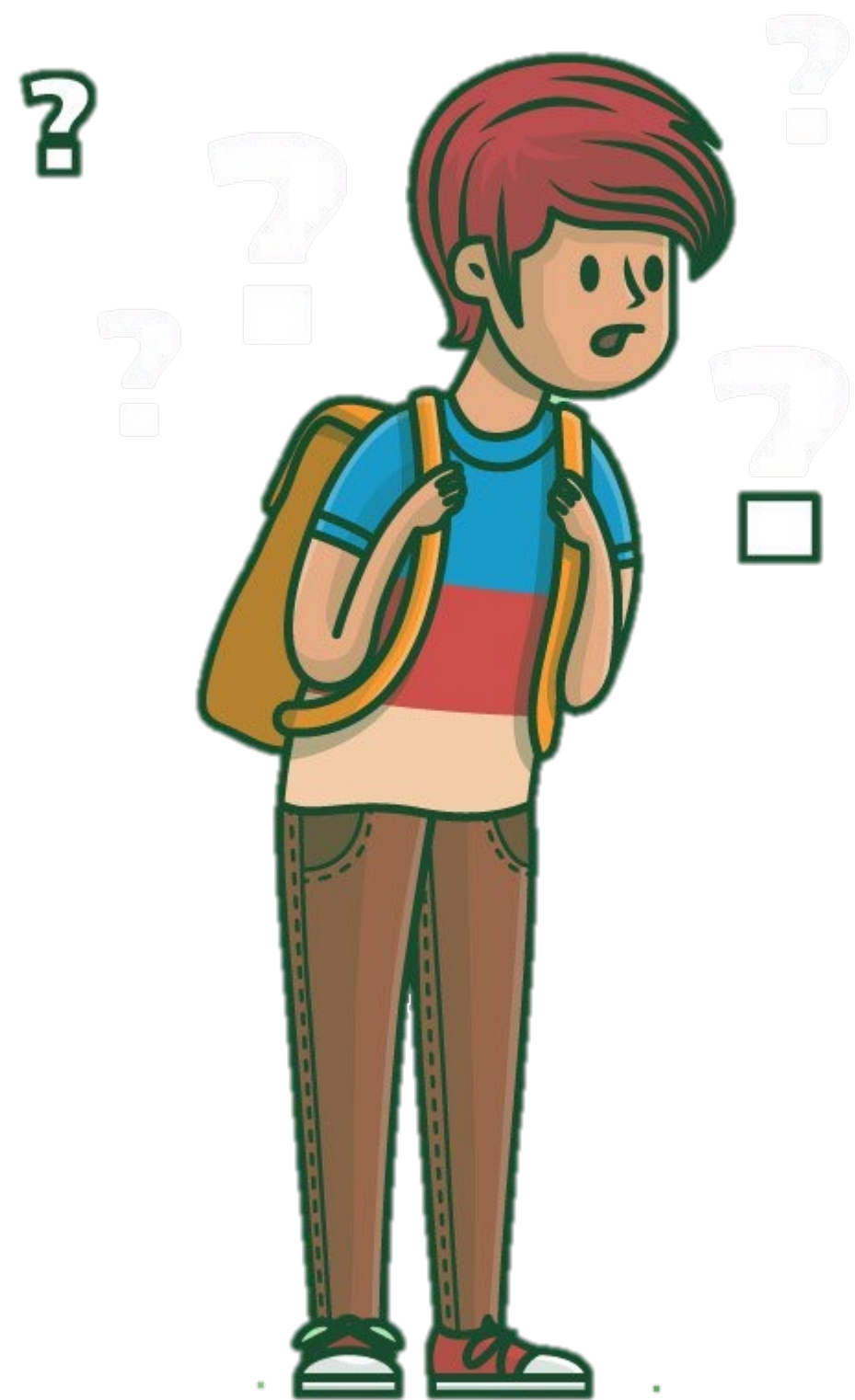
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อ



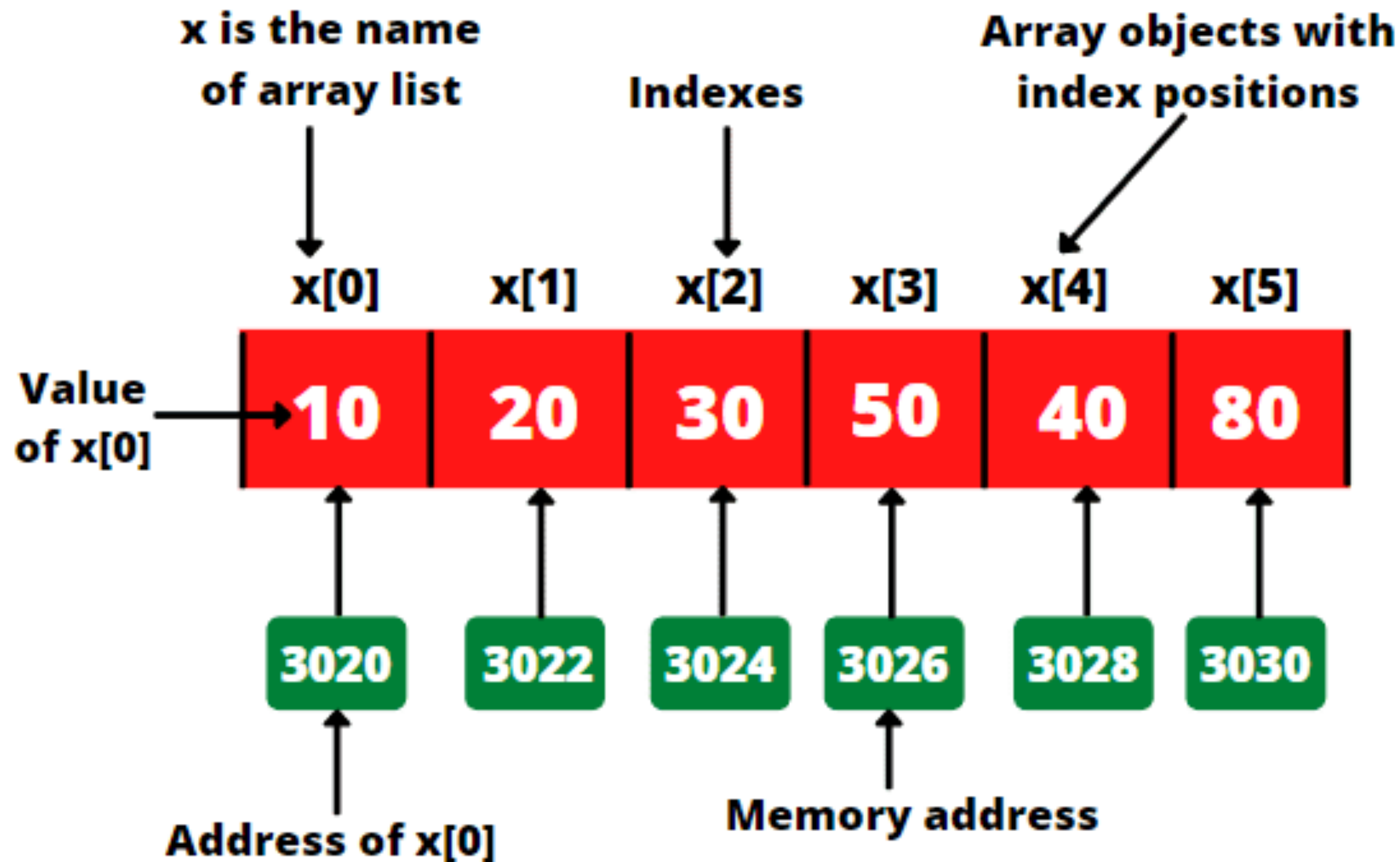
- Array
- Collection
 - Set
 - List
 - Map
- Generic

หัวข้อ



- Array
- Collection
 - Set
 - List
 - Map
- Generic

ตัวแปรอาร์เรย์



ตัวแปรอาร์เรย์ คือ ตัวแปรที่เป็นชนิดข้อมูลแบบอ้างอิงที่ใช้เก็บข้อมูลชนิดเดียวกันได้หลายค่า

ตัวอย่าง ตัวแปรอาร์เรย์ที่ชื่อ x มีการเก็บข้อมูลชนิด int มีสมาชิกจำนวน 6 ตัว โดยมีหมายเลขสมาชิกตั้งแต่ 0 ถึง 5

ประเภทของอาร์เรย์

- อาร์เรย์ข้อมูล**ชนิดพื้นฐาน** : อาร์เรย์ที่สามารถใช้เก็บข้อมูลที่มีชนิดข้อมูลแบบพื้นฐานชนิดใดชนิดหนึ่งได้หลายค่า เช่น
 - อาร์เรย์ของข้อมูลชนิด boolean
 - อาร์เรย์ของข้อมูลชนิด int
- อาร์เรย์ข้อมูล**ชนิดคลาส** : อาร์เรย์ที่สามารถใช้เก็บข้อมูลที่เป็นออบเจ็กต์ของคลาสใดๆ ได้หลายออบเจ็กต์ เช่น
 - อาร์เรย์ของข้อมูลชนิด String
 - อาร์เรย์ของข้อมูลชนิด Student

การประกาศตัวแปรอาร์เรย์

คล้ายกับการประกาศตัวแปรทั่วไป แต่ตัวแปรอาร์เรย์จะมีเครื่องหมาย **[]** อยู่หน้าหรือหลังชื่อตัวแปร เช่น

- `boolean[] b_data;` หรือ `boolean b_data[];`
- `int[] num;` หรือ `int num[];`
- `String[] s_arr;` หรือ `String s_arr[];`
- `Student[] stds;` หรือ `Student stds[];`

การสร้างอาร์เรย์

การสร้างอาร์เรย์จะอาศัยคำสั่ง **new**

```
variableName = new dataType[size];
```

ตัวอย่างเช่น

- `b_data = new boolean[5];`
- `num = new int[4];`
- `s_arr = new String[10];`
- `stds = new Student[200];`

การกำหนดค่าให้สมาชิกในอาร์เรย์

สมาชิกของอาร์เรย์*จะมีค่าตามค่าเริ่มต้น* (Default) ของชนิดข้อมูลนั้น ๆ เช่น

- สมาชิกของอาร์เรย์ของข้อมูลชนิด int จะมีค่าเริ่มต้นเป็น 0 ทั้งหมด
- สมาชิกของอาร์เรย์ของข้อมูลชนิดคลาส จะมีค่าเริ่มต้นเป็น null ทั้งหมด

ซึ่งเราสามารถกำหนดค่าให้สมาชิกของอาร์เรย์ด้วยตัวเองได้

```
variableName[index] = value;
```

ตัวอย่างเช่น

- `b_data[0] = true;`
- `num[1] = 20;`
- `s_arr[2] = "IT KMITL";`
- `stds[0] = new Student("Sompong", "IT");`

อาร์เรย์ของข้อมูลชนิดพื้นฐาน

```
public class Main {  
    public static void main(String args[]) {  
        // int []x;  
        // x = new int[3];  
  
        int []x = new int[3];  
        System.out.print("x = " + x + ", x[0] = "+x[0]);  
        System.out.println(", x[1] = "+ x[1] +", x[2] = "+x[2]);  
  
        boolean []y = new boolean[3];  
        System.out.print("y = " + y + " y[0] = "+y[0]);  
        System.out.println(" y[1] = " + y[1] + " y[2] = "+y[2]);  
  
        char []z = new char[3];  
        System.out.print("z = " + z + " z[0] = "+z[0]);  
        System.out.println(" z[1] = " + z[1] + " z[2] = "+z[2]);  
  
        String []s = new String[3];  
        System.out.print("s = " + s + " s[0] = "+s[0]);  
        System.out.println(" s[1] = " + s[1] + " s[2] = "+s[2]);  
    }  
}
```

Output - JavaApplication15 (run) x

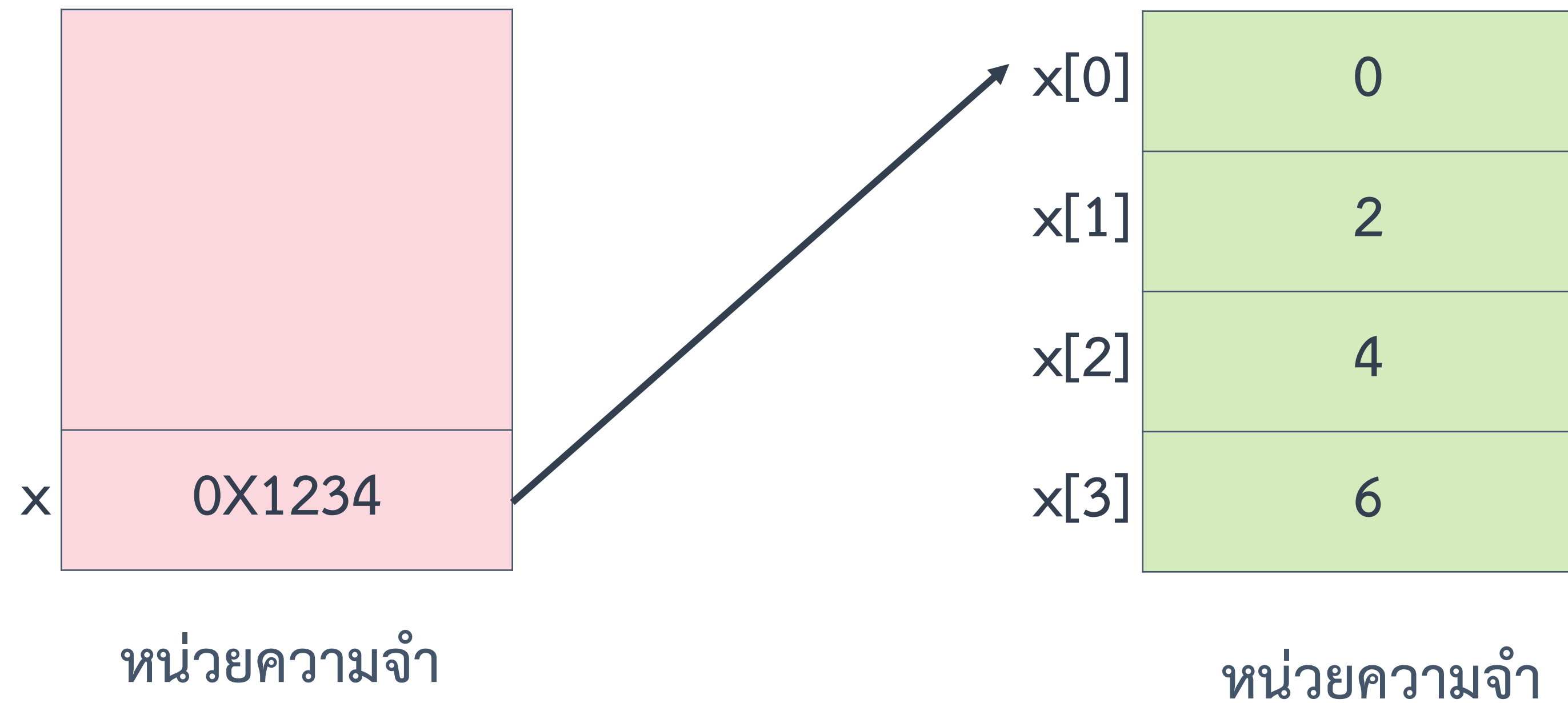
run:
x = [I@28a418fc x[0] = 0 x[1] = 0 x[2] = 0
y = [Z@27716f4 y[0] = false y[1] = false y[2] = false
z = [C@8efb846 z[0] = [] z[1] = [] z[2] = []
s = [Ljava.lang.String;@4f023edb s[0] = null s[1] = null s[2] = null
BUILD SUCCESSFUL (total time: 0 seconds)

อาร์เรย์ของข้อมูลชนิดพื้นฐาน

```
public class Main {  
    public static void main(String args[]) {  
        int []x;  
        x = new int[4];  
  
        x[0] = 0;  
        x[1] = 2;  
        x[2] = 4;  
        x[3] = 6;  
  
        System.out.println("x = "+x) ;  
        System.out.println("x[0] = "+x[0]) ;  
        System.out.println("x[1] = "+x[1]) ;  
        System.out.println("x[2] = "+x[2]) ;  
        System.out.println("x[3] = "+x[3]) ;  
    }  
}
```

อาร์เรย์ของข้อมูลชนิดพื้นฐาน

อาร์เรย์ในภาษาจาวาจะ*เป็นตัวแปรแบบอ้างอิง*ชนิดหนึ่ง (เช่นเดียวกับออปเจ็ค) ซึ่งจะอาศัยคำสั่ง `new` จะจองพื้นที่ในหน่วยความจำสำหรับเก็บค่าของสมาชิกของอาร์เรย์ ส่วนตัวแปรอาร์เรย์จะเก็บตำแหน่งอ้างอิงไปยังสมาชิกของอาร์เรย์



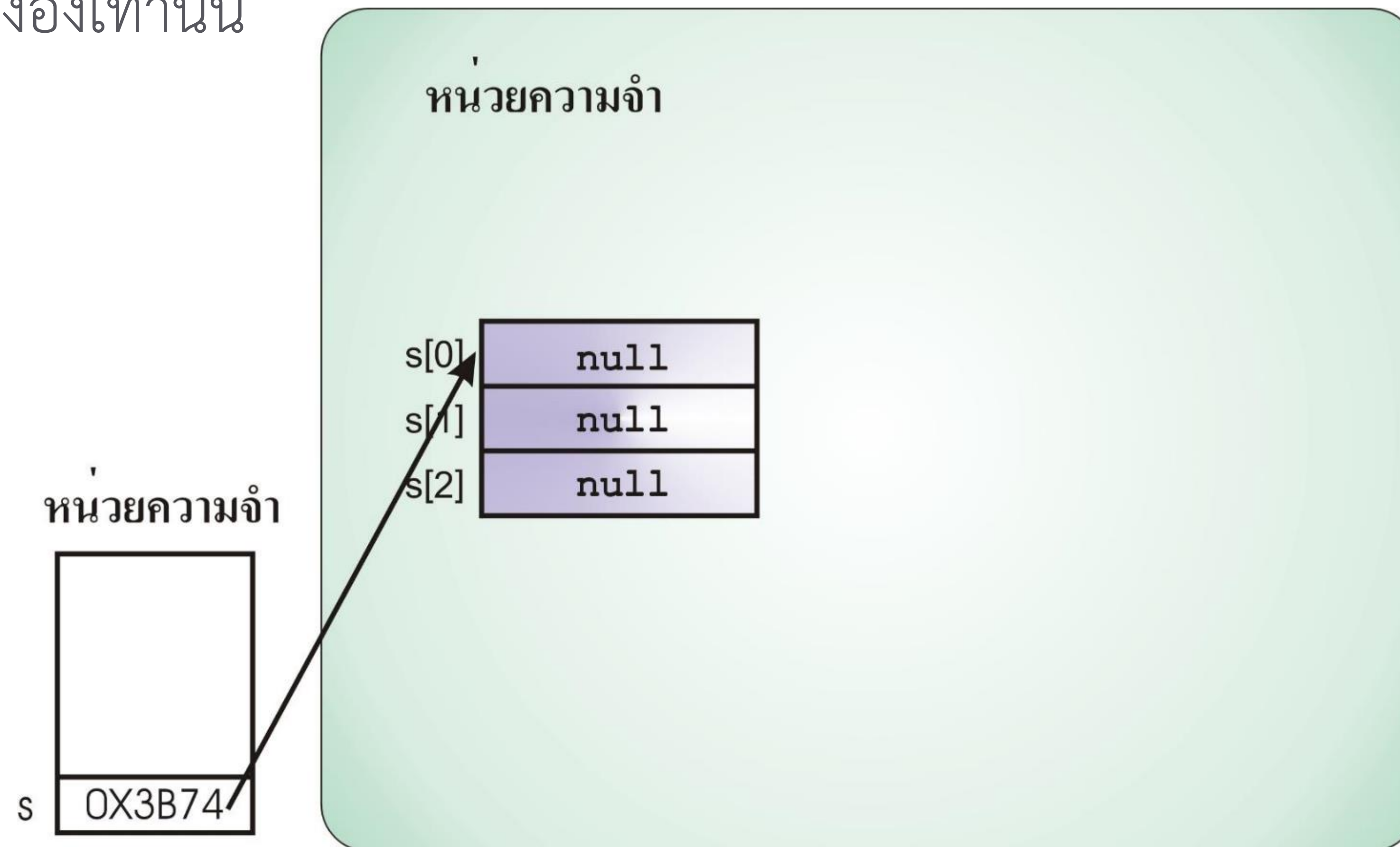
อาร์เรย์ของข้อมูลชนิดคลาส

```
public class Student {  
    private String id;  
    private String name;  
    private double gpa;  
    public Student(String id,  
        String name, double gpa) {  
        this.id = id;  
        this.name = name;  
        this.gpa = gpa;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

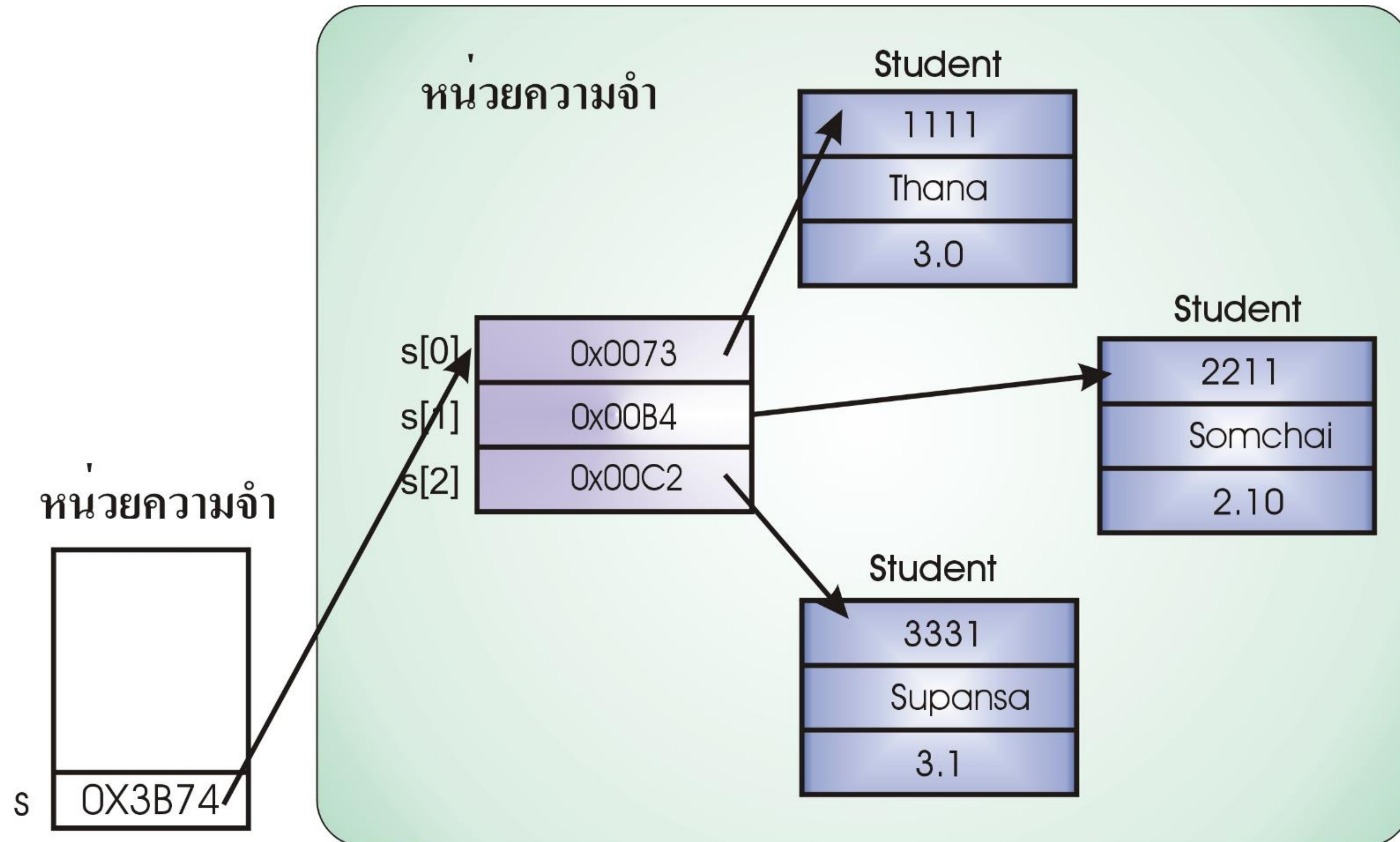
```
public class Main {  
    public static void main(String args[]) {  
        Student []s;  
        s = new Student[3];  
        s[0] = new Student("1111", "Thana", 3.0);  
        s[1] = new Student("2211", "Somchai", 2.10);  
        s[2] = new Student("3331", "Supansa", 3.1);  
  
        System.out.println("s size = " + s.length);  
        System.out.println("s[0] = "+s[0].getName());  
        System.out.println("s[1] = "+s[1].getName());  
        System.out.println("s[2] = "+s[2].getName());  
    }  
}
```

อาร์เรย์ของข้อมูลชนิดคลาส

คำสั่ง `new` จะจองเนื้อที่ในหน่วยความจำสำหรับเก็บค่าของสมาชิกของอาร์เรย์ ซึ่งจะเป็นเพียงแค่ตำแหน่งอ้างอิงเท่านั้น



อาร์เรย์ของข้อมูลชนิดคลาส



การกำหนดค่าเริ่มต้นให้สมาชิกอาร์เรย์

การสร้างอาร์เรย์พร้อมกับกำหนดค่าเริ่มต้นให้กับสมาชิกของอาร์เรย์

```
dataType []variableName = {value1, value2, .., valueN};
```

ตัวอย่างเช่น

- `int []num = {10, 20, 30, 40};`
- `Student []s = {new Student("1111", "Thana", 3.0),
new Student("2211", "Somchai", 2.10),
new Student("3331", "Supansa", 3.1)};`

การใช้คำสั่ง for กับตัวแปรอาร์เรย์

การใช้คำสั่ง for เพื่อการอ้างอิงและอ้างอิงสมาชิกในอาร์เรย์

ตัวอย่างเช่น

```
for(int i = 0; i < 5; i++) {  
  
    System.out.println(x[i]);  
  
}
```

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

ตัวอย่างการใช้ for



```
public class Main {  
    public static void main(String args[]) {  
        int []x;  
        x = new int[4];  
        for (int i=0; i<x.length; i++) {  
            x[i] = i*2;  
        }  
  
        System.out.println("x = " + x);  
  
        for (int i=0; i<x.length; i++) {  
            System.out.println("x["+i+"] = "+x[i]);  
        }  
    }  
}
```

ArrayIndexOutOfBoundsException

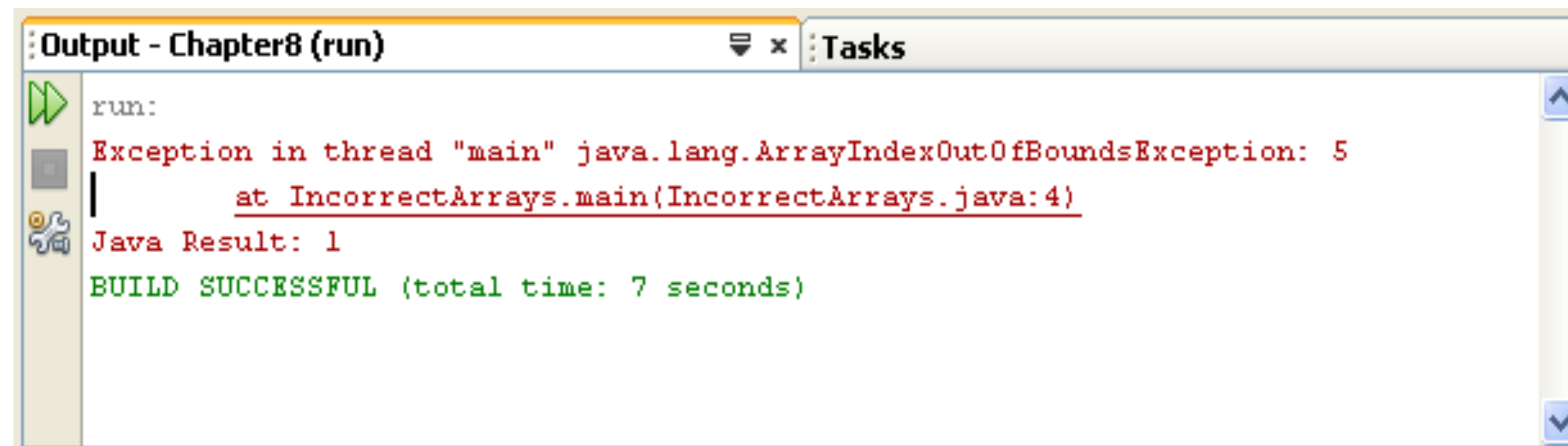
เป็นข้อผิดพลาดที่เกิดขึ้น เมื่อมีการอ้างอิงสมาชิกของอาร์เรย์นอกเหนือจากช่วงลำดับ (index) ของตัวแปรอาร์เรย์ ตัวอย่างเช่น

```
int []x = new int[4]
```

ซึ่งช่วง index ที่อ้างอิงสมาชิกได้ คือ 0 – 3 หากเกิดการอ้างอิงสมาชิกด้วย index อื่น เช่น `x[4]` จะทำให้เกิดข้อผิดพลาด `ArrayIndexOutOfBoundsException`

ตัวอย่างการเกิด ArrayIndexOutOfBoundsException

```
public class IncorrectArrays {  
    public static void main(String args[]) {  
        int []x = {4,3,5,1,8};  
        System.out.println(x[5]);  
    }  
}
```



The screenshot shows an IDE window titled "Output - Chapter8 (run)". The output text is as follows:

```
run:  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5  
    at IncorrectArrays.main(IncorrectArrays.java:4)  
Java Result: 1  
BUILD SUCCESSFUL (total time: 7 seconds)
```

อาร์เรย์หลายมิติ

	0	1	2	3
0				
1				
2				
num				

ในภาษาจาวาเราสามารถที่จะประกาศอาร์เรย์ได้มากกว่า 1 มิติ

ตัวอย่างการประกาศอาร์เรย์สองมิติ

```
int [ ][ ] num;
```

เป็นการประกาศตัวแปร num เป็นตัวแปรอาร์เรย์สองมิติ

ตัวอย่างการสร้างอาร์เรย์สองมิติ

```
variableName = new dataType[row][col];
```

```
num = new int[3][4]
```

เป็นการสร้างตัวแปรอาร์เรย์สองมิติ num ซึ่งประกอบด้วยข้อมูลจำนวน 3 แถวๆ ละ 4 หลัก

อาร์เรย์สองมิติที่แต่ละแถวมีหลักไม่เท่ากัน

การสร้างอาร์เรย์ 2 มิติในภาษาจาวา **ไม่จำเป็นที่จำนวนคอลัมน์ของแต่ละแถวจะต้องเท่ากัน**

ตัวอย่าง

```

int [][]x = new int[3][];
x[0] = new int[4];
x[1] = new int[2];
x[2] = new int[3];
  
```

	0	1	2	3
0				
1				
2				

x

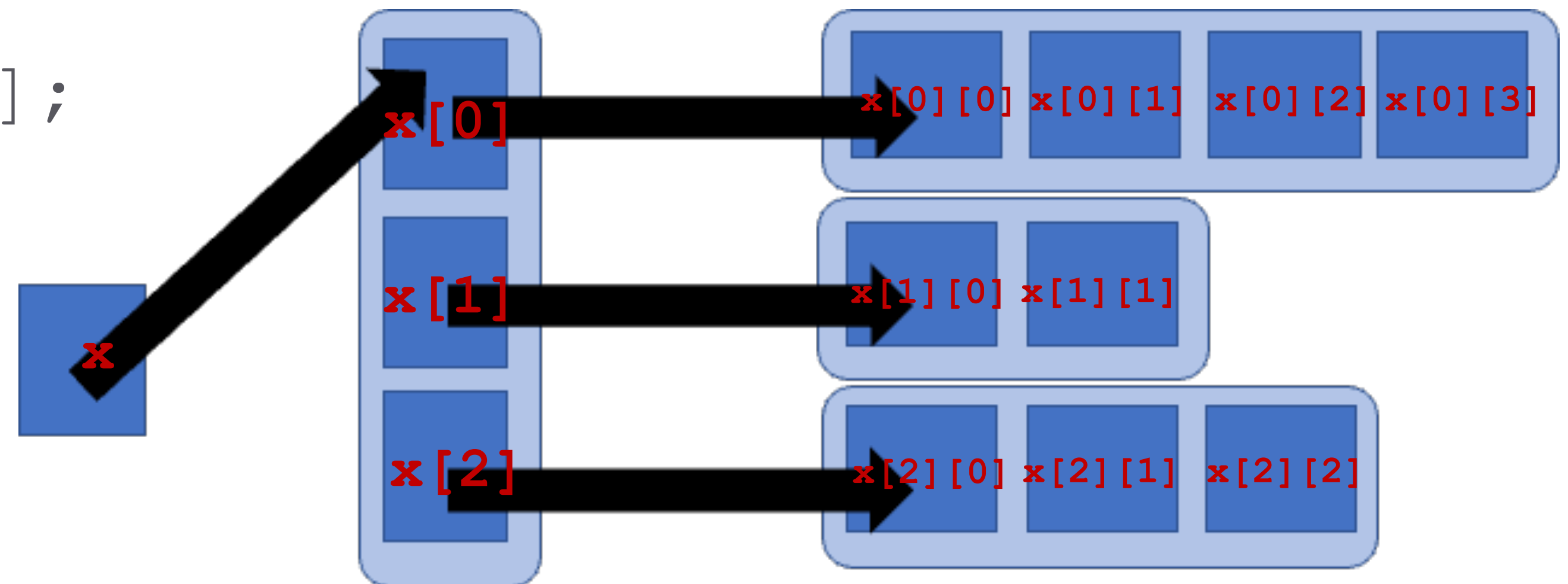
อาร์เรย์สองมิติที่แต่ละแถวมีหลักไม่เท่ากัน

การสร้างอาร์เรย์ 2 มิติในภาษาจาวา **ไม่จำเป็นที่จำนวนคอลัมน์ของแต่ละแถวจะต้องเท่ากัน**

ตัวอย่าง

```

int [][]x = new int[3][];
x[0] = new int[4];
x[1] = new int[2];
x[2] = new int[3];
  
```



ตัวอย่างอาร์เรย์สองมิติ

```

public class Main {
    public static void main(String args[]) {
        int x[][] = new int[3][];
        x[0] = new int[4];
        x[1] = new int[2];
        x[2] = new int[3];
        for(int i=0; i<x.length; i++) {
            for(int j=0; j<x[i].length; j++) {
                x[i][j] = (i+j)*2;
            }
        }
        for(int i=0; i<x.length; i++) {
            for(int j=0; j<x[i].length; j++) {
                System.out.print(x[i][j]+" ");
            } System.out.println();
        }
    }
}
  
```

x	0	1	2	3
0	0	2	4	6
1	2	4		
2	4	6	8	


```

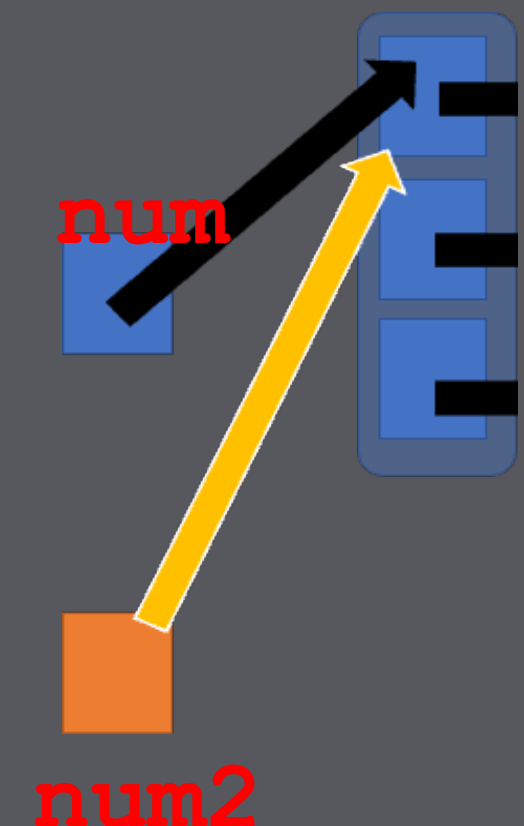
public class Main {
    public static void main(String args[]) {
        int num[] = {5,4,1,3};
        System.out.println("address 1: "+ num);
        for (int i=0; i<4;i++)
            System.out.println(num[i]);
        System.out.println("-----");
        int num2[];
        num2 = num;    // int num2[] = num;
        System.out.println("address 1: "+ num);
        System.out.println("address 2: "+ num2);
        for (int i=0; i<4;i++)
            System.out.println(num2[i]);
        System.out.println("-----");
        num2[1] = num2[1]*10;
        for (int i=0; i<4;i++)
            System.out.println("num1: "+ num[i]+" num2: "+num2[i]);
    }
}

```

```

address 1: [I@2a139a55
5
4
1
3
-----
address 1: [I@2a139a55
address 2: [I@2a139a55
5
4
1
3
-----
num1: 5 num2: 5
num1: 40 num2: 40
num1: 1 num2: 1
num1: 3 num2: 3

```



การคัดลอกค่าข้อมูลของอาร์เรย์

เราไม่สามารถเปลี่ยนแปลงขนาดของอาร์เรย์ แต่สามารถคัดลอกค่าข้อมูลสมาชิกของอาร์เรย์ได้
โดยใช้คำสั่ง `System.arraycopy()` ;

```
public class Main {  
    public static void main(String args[]) {  
        String []scr = {"Copy","an","array","from",  
                        " source"," to"," destination."};  
        String []dst = new String[4];  
        System.arraycopy(scr,3,dst,0,4);  
        for(int i=0; i<dst.length; i++) {  
            System.out.print(dst[i]);  
        }  
        System.out.println();  
    }  
}
```

ผลลัพธ์การทำงานโปรแกรม
from source to destination.

การหาขนาดของอาร์เรย์

อาร์เรย์ในภาษาจาวาจะมีคุณลักษณะที่ชื่อ `length` ซึ่งจะมีความเท่ากับจำนวนสมาชิกทั้งหมดของอาร์เรย์นั้น

ตัวอย่าง

```
int x[] = new int[3];  
System.out.print(x.length);           // มีความเท่ากับ 3
```

```
int x[][] = new int[3][4];  
System.out.print(x.length);           // มีความเท่ากับ 3  
System.out.print(x[1].length);        // มีความเท่ากับ 4
```

เมธอดสำหรับอาร์เรย์



เมธอดในคลาส Arrays ที่เกี่ยวข้องกับอาร์เรย์มีดังต่อไปนี้

- **sort(variableName)**

คือ ทำการเรียงลำดับสมาชิกในตัวแปรอาร์เรย์ variableName

- **binarySearch(variableName, target)**

คือ ทำการค้นหา target ในตัวแปรอาร์เรย์ variableName โดยจะคืนค่าตำแหน่งของ target ในตัวแปรอาร์เรย์

- **fill(variableName, value)**

คือ ทำการกำหนดค่า value ให้กับสมาชิกทุกตัวของตัวแปรอาร์เรย์ variableName

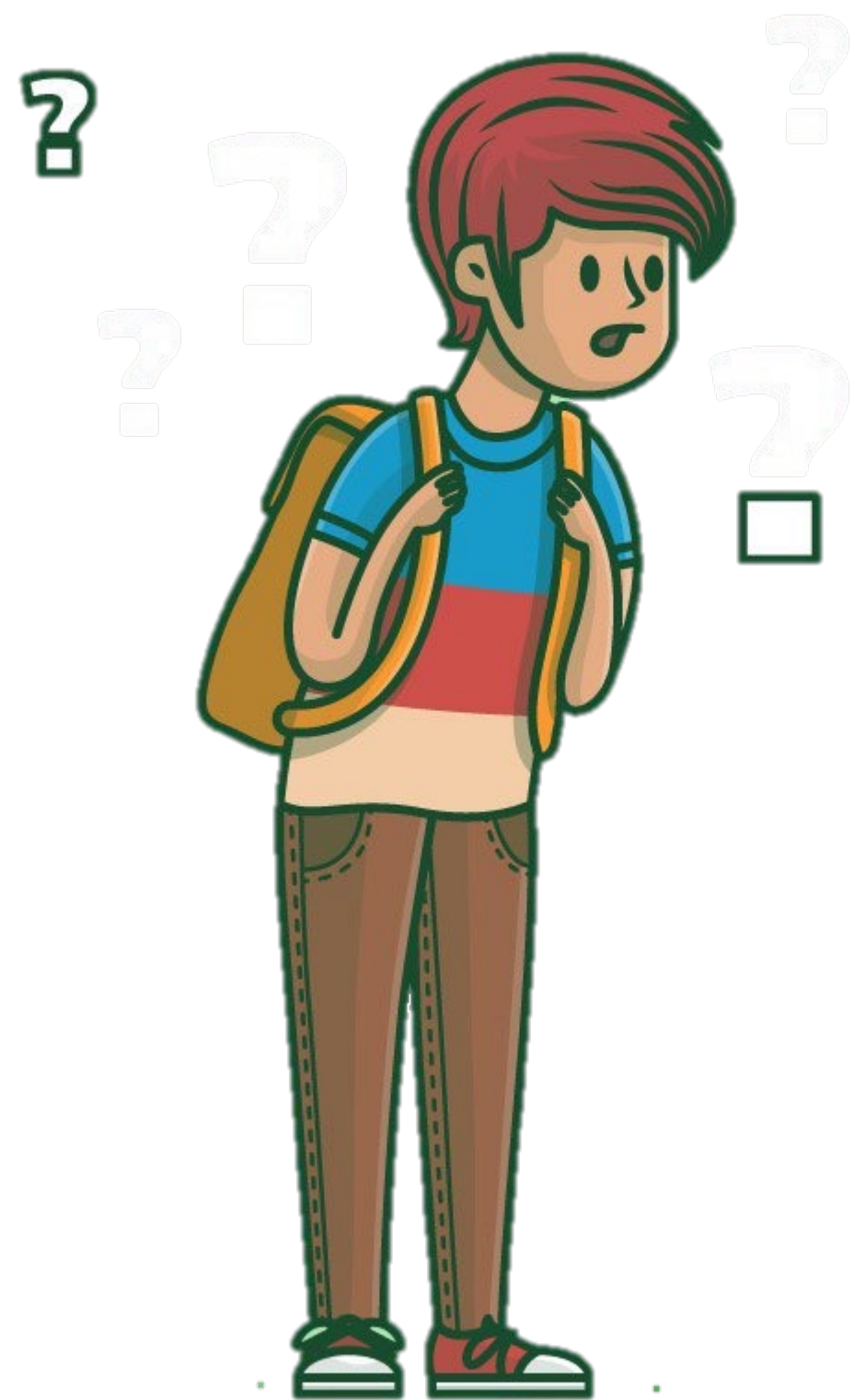
ตัวอย่างการใช้เมธอดสำหรับอาร์เรย์

```
import java.util.Arrays;
public class MethodsArrays {
    public static void main(String args[]) {
        double d[] = {5.3, 3.56, 0.5, 1.65, 7.8};
        Arrays.sort(d);
        for(int i=0; i<d.length; i++) {
            System.out.print(d[i]+" ");
        } System.out.println();
        int pos = Arrays.binarySearch(d,1.65);
        System.out.println("arrays["+pos+"] = 1.65");
        Arrays.fill(d,1.0);
        for(int i=0; i<d.length; i++) {
            System.out.print(d[i]+" ");
        }
        System.out.println();
    }
}
```

ผลลัพธ์การทำงานโปรแกรม

```
0.5 1.65 3.56 5.3 7.8
arrays[1] = 1.65
1.0 1.0 1.0 1.0 1.0
```


หัวข้อ

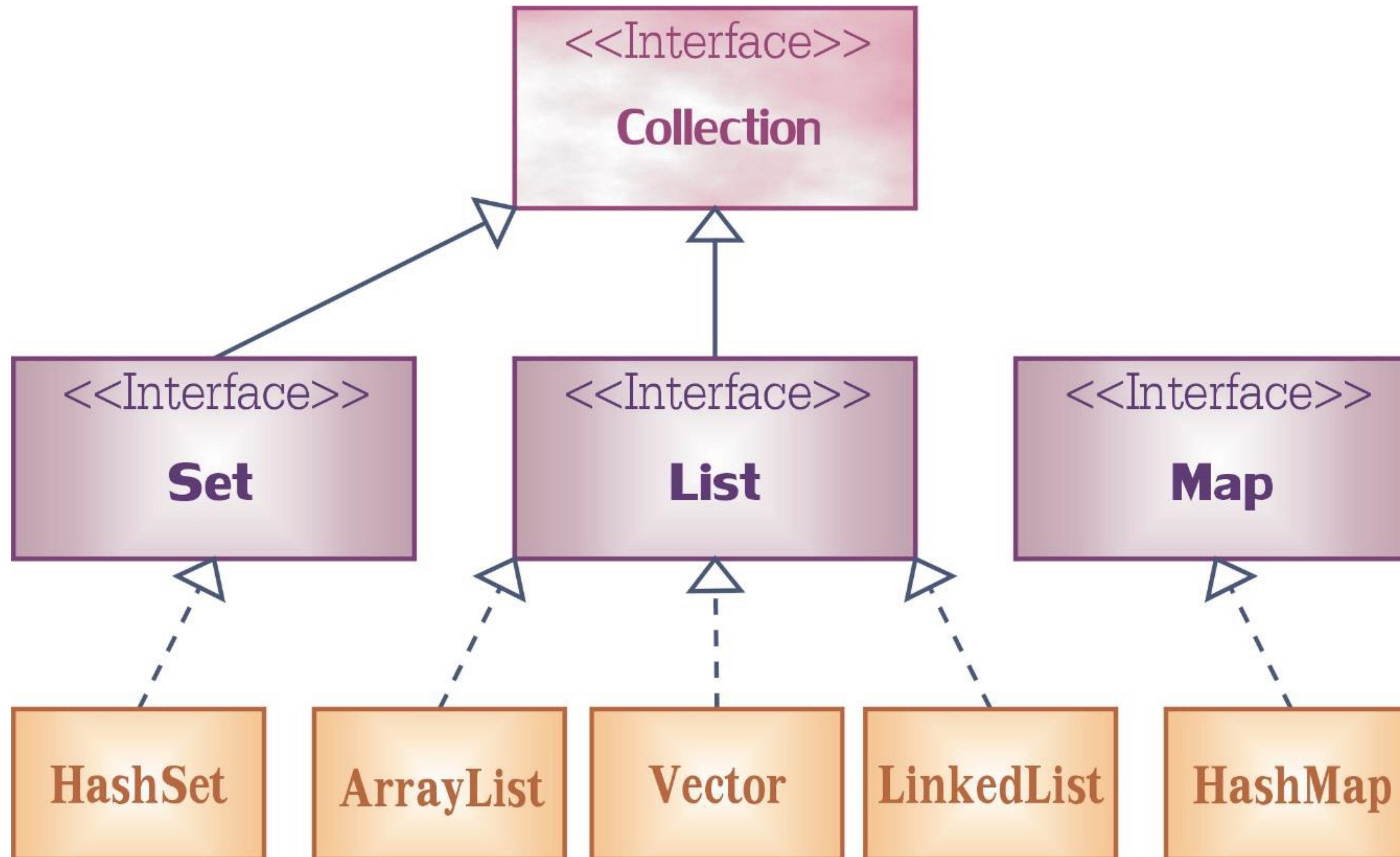


- Array
- Collection
 - Set
 - List
 - Map
- Generic

Collection API

- คลาสใน Collection API สามารถที่จะนำมาใช้เก็บข้อมูลที่เป็นออพเจ็คได้หลายตัว
- โครงสร้างข้อมูลของคลาสใน Collection API จะคล้ายกับของอาร์เรย์ต่างกันตรงที่
 - *ขนาดโครงสร้างข้อมูลของคลาสใน Collection API สามารถที่จะถูกเปลี่ยนแปลงได้*
- Collection API ประกอบไปด้วย **อินเตอร์เฟส** ที่สำคัญ ดังนี้
 - Collection, Set, List, Map
- Collection API ประกอบไปด้วย **คลาส** ที่สำคัญ ดังนี้
 - HashSet, ArrayList, Vector, LinkedList, HashMap

Collection API

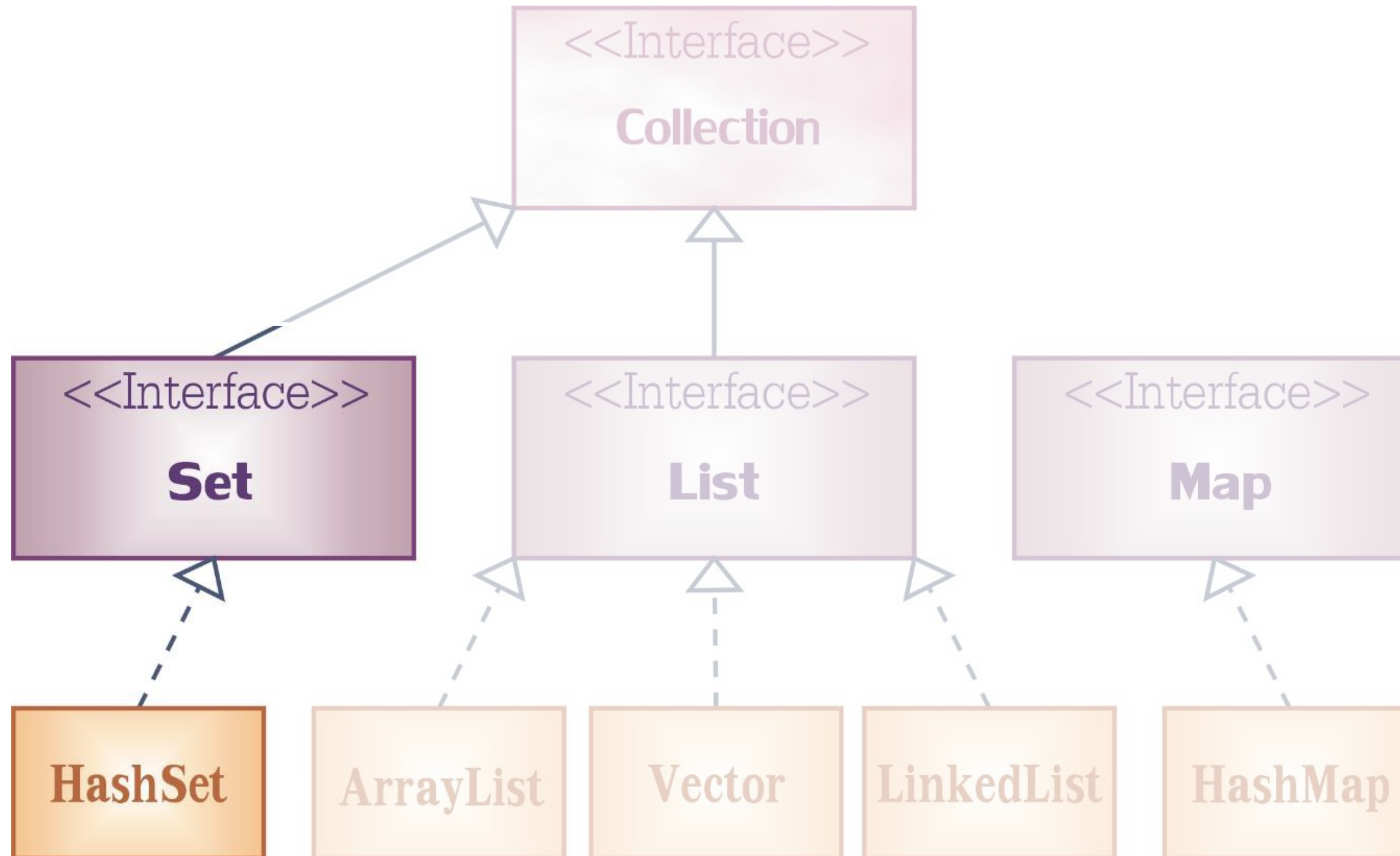


อินเตอร์เฟส Collection

สามารถที่จะระบุหรือไม่ระบุลำดับความสำคัญของสมาชิก และสามารถที่จะกำหนดให้ค่าข้อมูลของสมาชิกซ้ำกันหรือไม่ก็ได้ ซึ่งอินเตอร์เฟส Collection มีเมธอดที่สำคัญ ดังนี้

- `boolean add(Object element)`
- `boolean remove(Object element)`
- `int size()`
- `boolean isEmpty()`
- `boolean contains(Object element)`
- `Iterator iterator()`

Collection API



อินเตอร์เฟส Set



- สืบทอดมาจากอินเตอร์เฟส Collection
- ค่าข้อมูลของสมาชิกจะไม่สามารถซ้ำกันได้ และลำดับของสมาชิกไม่มีความสำคัญ
- คลาสสำคัญที่ implement อินเตอร์เฟส Set คือคลาส HashSet

ตัวอย่างการใช้ HashSet

ผลลัพธ์การทำงานโปรแกรม

```
import java.util.*;
public class SampleSet {
    public static void main(String args[]) {
        HashSet s = new HashSet();
        s.add("C#");
        s.add("Java");
        s.add("Pascal");
        System.out.println("The size of this set is "+s.size());
        System.out.println("The contents are "+s);
        System.out.println("Removing C#");
        s.remove("C#");
        System.out.println("Now this set contains C#: "+ s.contains("C#"));
        s.add("Java");
        System.out.println("Now the size is "+s.size());
        System.out.println("The contents are "+s);
    }
}
```

```
The size of this set is 3
The contents are [Java, Pascal, C#]
Removing C#
Now this set contains C#: false
Now the size is 2
The contents are [Java, Pascal]
```

อินเตอร์เฟส Iterator

- เป็นอินเตอร์เฟสที่มีไว้เพื่อใช้ในการเข้าถึงข้อมูลสมาชิกประเภท Set เนื่องจาก Set ไม่สามารถอ้างอิงสมาชิกทีละตัวได้ โดยมีเมธอดที่สำคัญ ดังนี้
 - `boolean hasNext()`
 - `Object next()`
 - `void remove()`
- ภายในอินเตอร์เฟส Collection จะมีเมธอดที่ชื่อ `iterator()` ซึ่งเป็นเมธอดที่จะส่งค่ากลับเป็น Iterator

ตัวอย่างการใช้ Iterator

```
import java.util.*;
public class c1 {
    public static void main(String args[]) {
        HashSet scrSet = new HashSet();
        scrSet.add("C#");
        scrSet.add("Java");
        scrSet.add("Pascal");

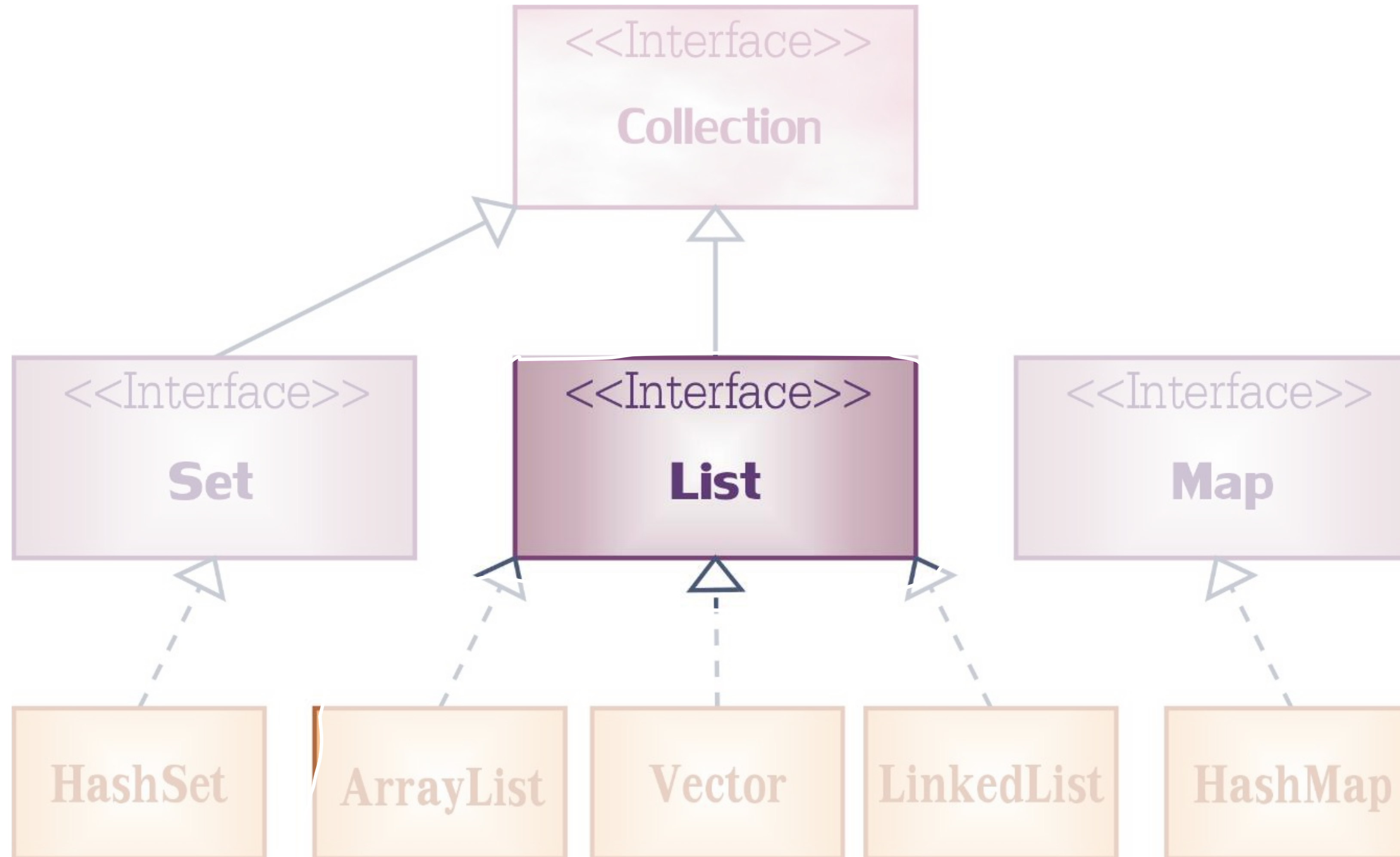
        Iterator it = scrSet.iterator();

        HashSet dstSet = new HashSet();
        while(it.hasNext()) {
            dstSet.add(it.next());
        }
        System.out.println(dstSet);
    }
}
```

ผลลัพธ์การทำงานโปรแกรม

[C#, Java, Pascal]

Collection API



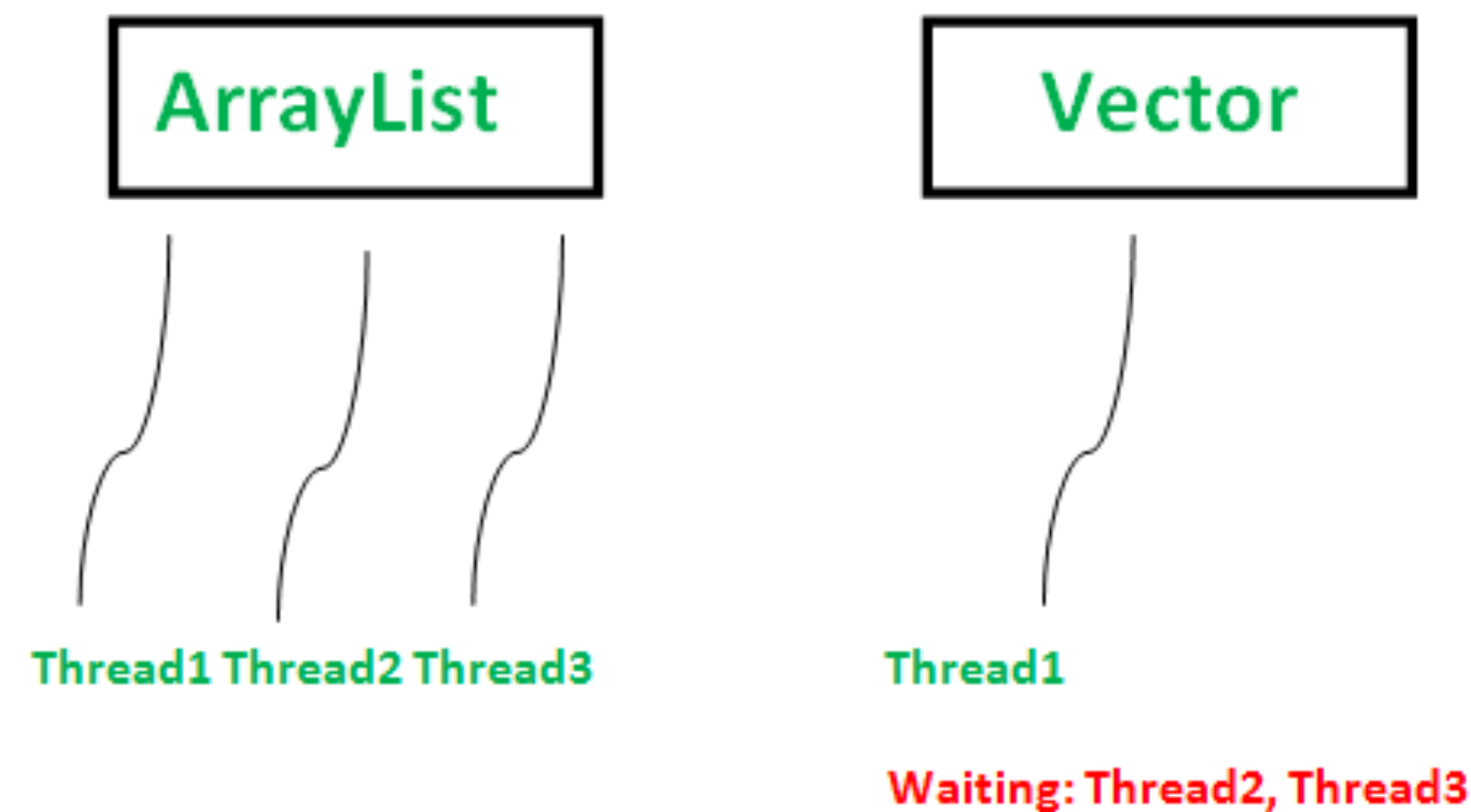
อินเตอร์เฟส List

- สืบทอดมาจากอินเตอร์เฟส Collection
- ค่าข้อมูลของสมาชิกอาจจะสามารถซ้ำกันได้ และลำดับของสมาชิกมีความสำคัญ
- อินเตอร์เฟส List มีเมธอดที่สำคัญที่เพิ่มมาจากอินเตอร์เฟส Collection ดังนี้
 - `void add(int index, Object element)`
 - `Object remove(int index)`
 - `Object get(int index)`
 - `void set(int index, Object element)`
 - `int indexOf(Object element)`
 - `ListIterator listIterator()`
- คลาสสำคัญที่ implement อินเตอร์เฟส List คือคลาส ArrayList, Vector และ LinkedList

Collection API

จากภาพจะพบว่าคลาสที่ implement อินเตอร์เฟส List ได้แก่ คลาส ArrayList, Vector และ LinkedList ซึ่งทั้ง 3 คลาสมีความแตกต่างกันดังนี้

- **ArrayList** ได้พัฒนาต่อยอดมาจาก array ที่มีความสามารถในการปรับขนาดได้ ซึ่งแต่ละสามารถอ้างอิงถึงสมาชิกได้โดยตรงผ่านเมธอด `get()` และ `set()`
- **LinkedList** ได้พัฒนาต่อยอดมาจาก double linked list โดยมีประสิทธิภาพ (ใช้พื้นที่จัดเก็บที่น้อยกว่า) ทางด้านการเพิ่มข้อมูลและลบที่ดีกว่า ArrayList แต่ไม่ค่อยดีในสำหรับการอ้างอิงถึงสมาชิกผ่านเมธอด `get()` และ `set()`
- **Vector** มีความคล้ายคลึงกับ ArrayList แต่ Vector จะมีความสามารถทางด้าน synchronize กล่าวคือ กรณีมีเธรด (หน่วยการทำงานของ Process) เรียกใช้งาน Vector มากกว่า 1 ตัวระบบจะจัดการเข้าใช้งาน โดยที่ ณ ขณะใด ๆ จะมีได้เพียงหนึ่งเธรดเท่านั้นที่สามารถเรียกใช้งาน Vector นั้น ๆ ได้



ข้อแนะนำ

- Vector และ ArrayList ต้องการพื้นที่จัดเก็บข้อมูลเท่ากับจำนวนสมาชิกที่ต้องการจัดเก็บ
- Vector ต้องการพื้นที่จัดเก็บข้อมูลโดยประมาณเท่ากับ 2 เท่าของ array ต่อครั้งในการเพิ่มข้อมูล
- ArrayList ต้องการพื้นที่จัดเก็บข้อมูลโดยประมาณเท่ากับ 1.5 เท่าของ array ต่อครั้งในการเพิ่มข้อมูล อย่างไรก็ตาม พื้นที่จัดเก็บข้อมูลตอนกำหนดค่าเริ่มต้นจะกินขนาดค่อนข้างเล็ก ดังนั้น ถ้าสามารถบรรจุข้อมูลตั้งแต่เริ่มต้นได้ก็สามารถหลีกเลี่ยงปัญหาเรื่องพื้นที่ ๆ ใช้จัดเก็บได้

	ArrayList	LinkedList
<code>get()</code>	$O(1)$	$O(n)$
<code>add()</code>	$O(1)$	$O(1)$ amortized
<code>remove()</code>	$O(n)$	$O(n)$

Collection API

LinkedList มีประสิทธิภาพการเพิ่มข้อมูลและลบที่ดีกว่า **ArrayList** แต่ไม่ค่อยดีในสำหรับการอ้างอิงถึงสมาชิกผ่านเมธอด `get()` และ `set()` ซึ่งสามารถอธิบายด้วยโปรแกรมดังต่อไปนี้

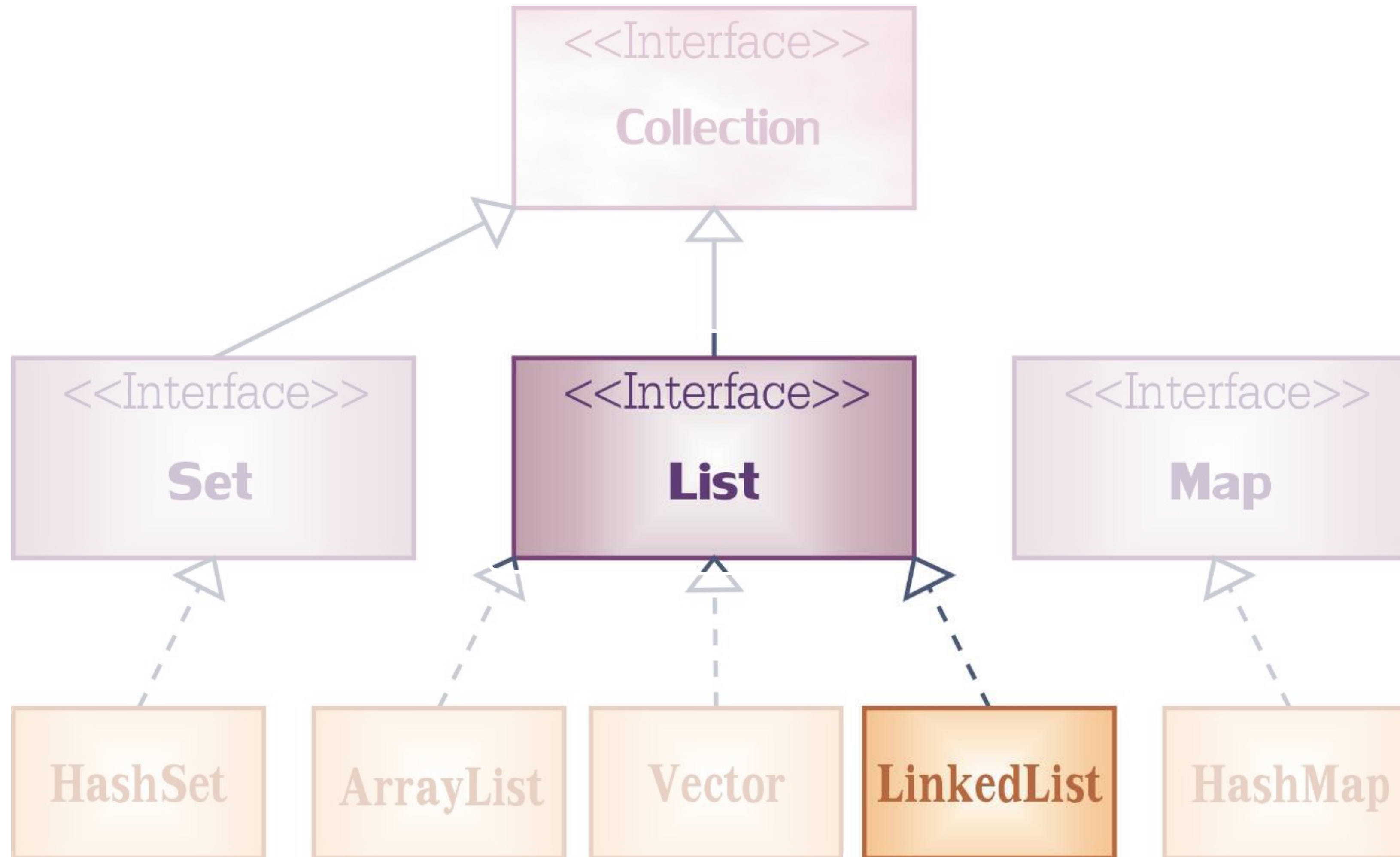
ArrayList

```
public Object get(ArrayList a, int i){  
    return a[i];  
}
```

LinkedList

```
public Object get(LinkedList a, int i){  
    int x = 0;  
    while(a.hasNext()){  
        if(++x == i)  
            return a.next();  
    }  
}
```

Collection API



ตัวอย่างการใช้ LinkedList

```
import java.util.*;
public class SampleList {
    public static void main(String args[]) {
        LinkedList l = new LinkedList();
        l.add("C#");
        l.add("Java");
        l.add("Pascal");
        System.out.println("The size is "+l.size());
        System.out.println("The contents are "+l);
        System.out.println("The first one is "+l.get(0));
        l.add("Java");
        System.out.println("The contents are "+l);
        System.out.println("The index of Java is "+ l.indexOf("Java"));
    }
}
```

ผลลัพธ์การทำงานโปรแกรม

```
The size is 3
The contents are [C#, Java, Pascal]
The first one is C#
The contents are [C#, Java, Pascal, Java]
The index of Java is 1
```

อินเทอร์เฟส ListIterator

เป็นอินเทอร์เฟสที่สืบทอดมาจากอินเทอร์เฟส Iterator ซึ่งใช้ในการเข้าถึงข้อมูลสมาชิกประเภท List โดยมีเมธอดที่สำคัญที่เพิ่มมาจาก Iterator ดังนี้

- `boolean hasPrevious()`
- `Object previous()`
- `void add(Object element)`
- `void set(Object element)`

ภายในอินเทอร์เฟส List จะมีเมธอดที่ชื่อ `listIterator()` ซึ่งเป็นเมธอดที่จะส่งค่ากลับเป็น ListIterator

ตัวอย่างการใช้ ListIterator

```
import java.util.*;
public class c1 {
    public static void main(String args[]) {

        LinkedList l = new LinkedList();
        l.add("C#");
        l.add("Java");
        l.add("Pascal");

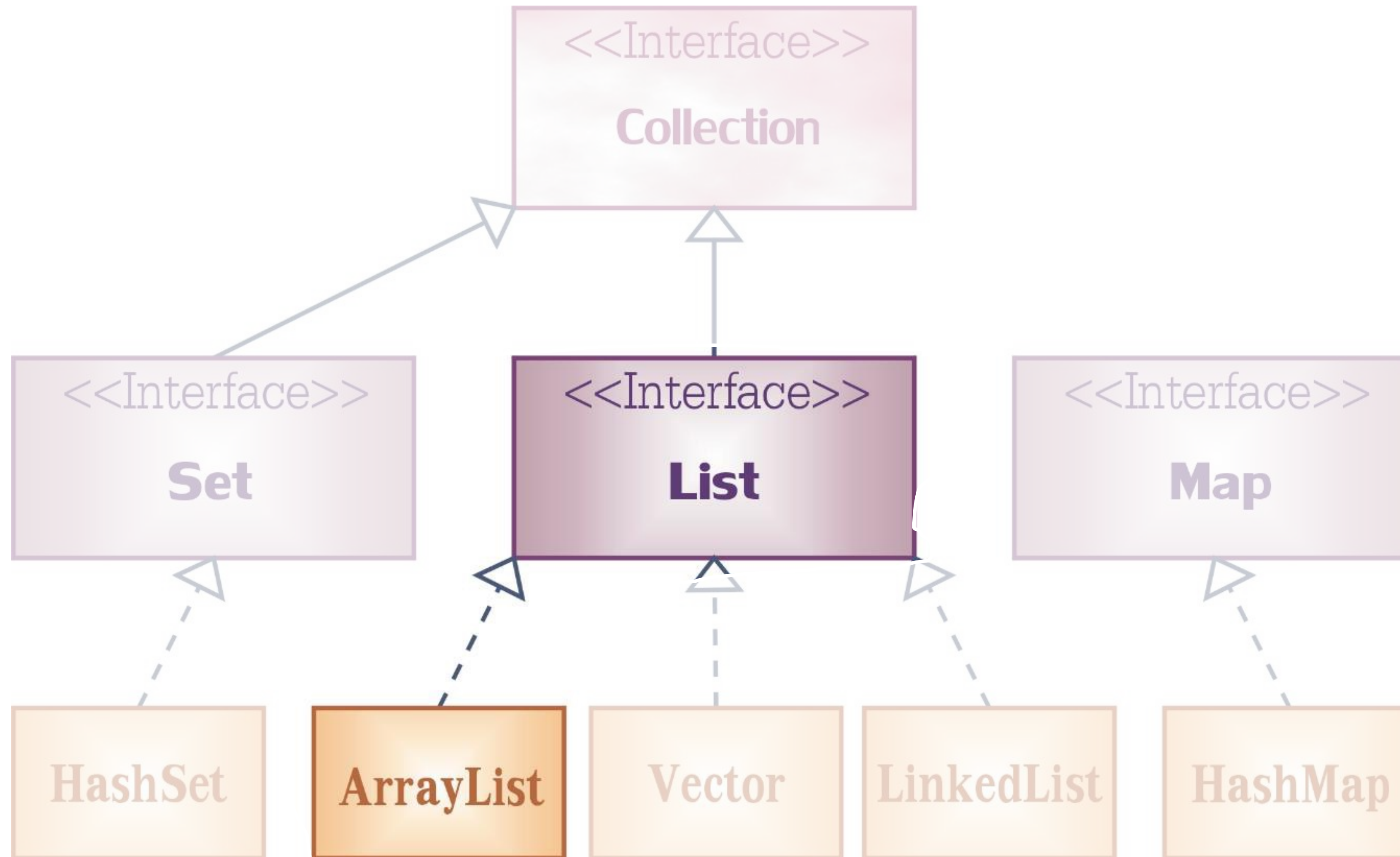
        ListIterator ll = l.listIterator();

        while (ll.hasNext())
            System.out.println(ll.next());
    }
}
```

ผลลัพธ์การทำงานโปรแกรม

```
C#
Java
Pascal
```


Collection API



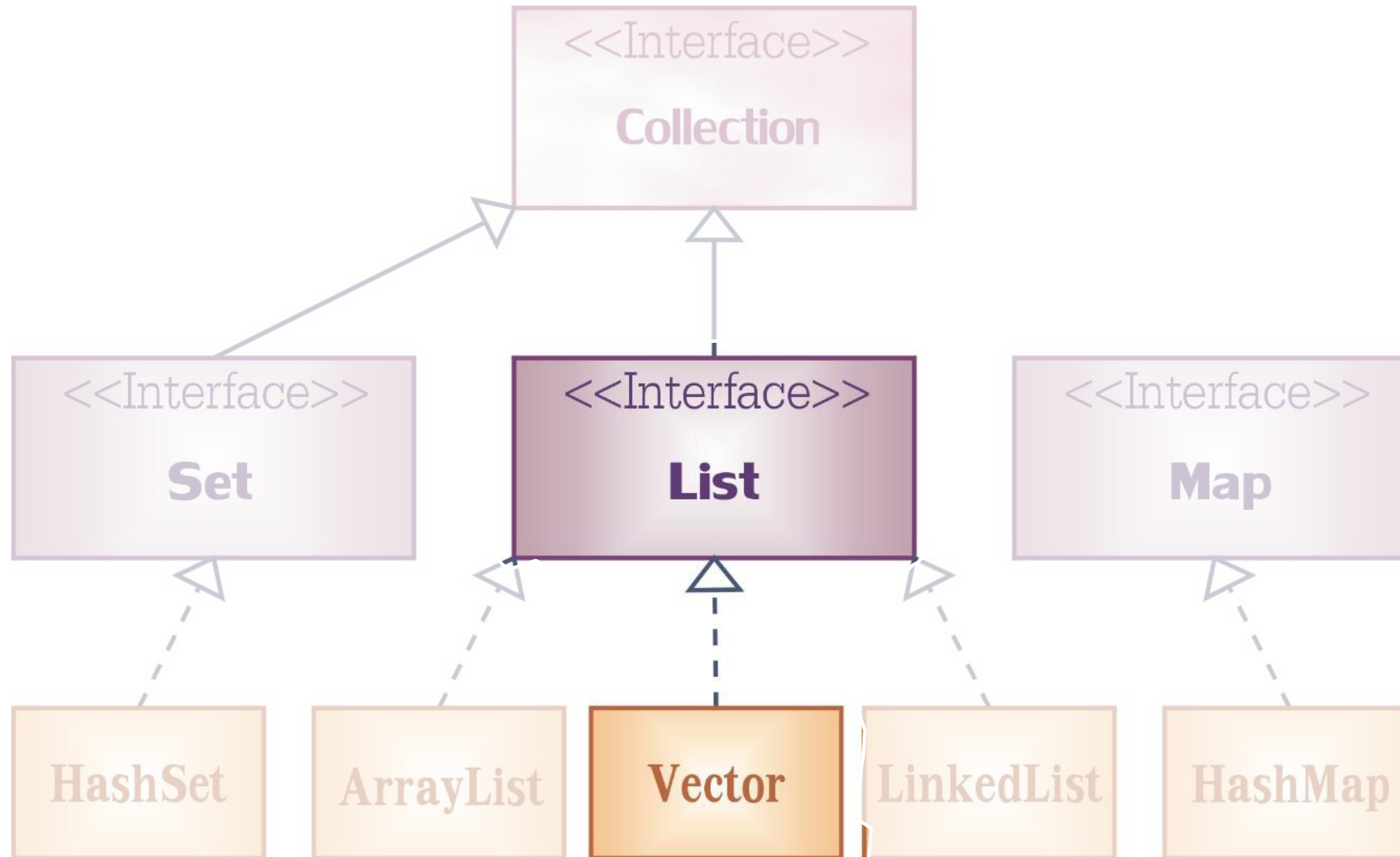
ตัวอย่างการใช้ ArrayList

```
import java.util.*;
public class SampleList {
    public static void main(String args[]) {
        ArrayList l = new ArrayList ();
        l.add("C#");
        l.add("Java");
        l.add("Pascal");
        System.out.println("The size is "+l.size());
        System.out.println("The contents are "+l);
        System.out.println("The first one is "+l.get(0));
        l.add("Java");
        System.out.println("The contents are "+l);
        System.out.println("The index of Java is "+ l.indexOf("Java"));
    }
}
```

ผลลัพธ์การทำงานโปรแกรม

```
The size is 3
The contents are [C#, Java, Pascal]
The first one is C#
The contents are [C#, Java, Pascal, Java]
The index of Java is 1
```

Collection API



คลาส Vector

เป็นคลาสที่ implements อินเตอร์เฟส List มี constructor แบบต่าง ๆ ดังนี้

- `new Vector()`
- `new Vector(int initialCapacity)`
- `new Vector(int initialCapacity, int capacityIncrement)`

นอกจากนี้ การเข้าถึงข้อมูลสมาชิกประเภท `Vector` นิยามอาศัยอินเตอร์เฟส Enumeration ซึ่งหลักการทำงานคล้ายกับอินเตอร์เฟส Iterator โดยมีเมธอดที่สำคัญ ดังนี้

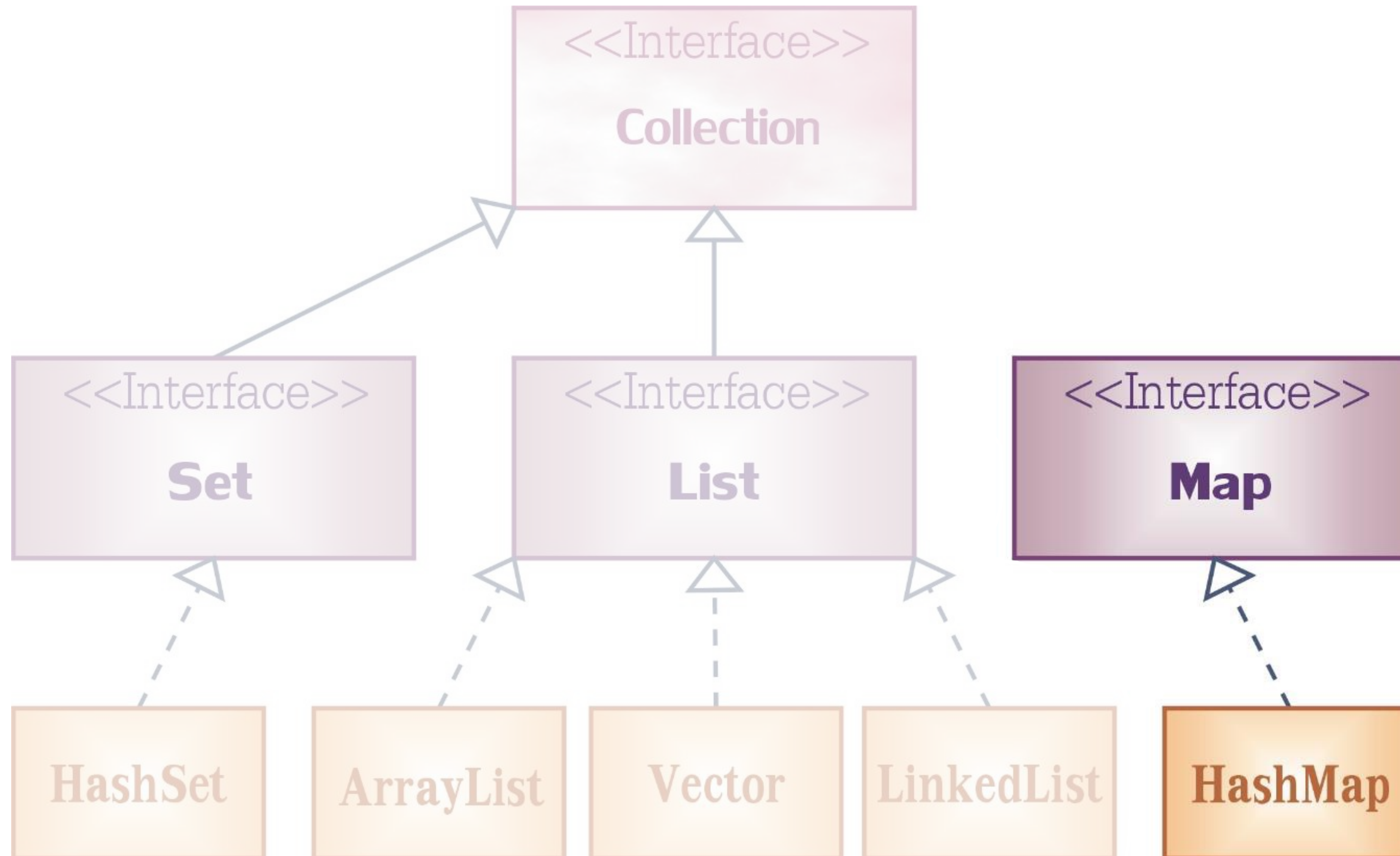
- `boolean hasMoreElement()`
- `Object nextElement()`

ตัวอย่างการใช้ Vector และ Enumeration

```
import java.util.*;

public class SampleEnumeration {
    public static void main(String args[]) {
        Vector v = new Vector();
        v.add("C#");
        v.add("Java");
        v.add("Pascal");
        Enumeration e = v.elements();
        while (e.hasMoreElements()) {
            System.out.print(e.nextElement()+" ");
        }
    }
}
```

Collection API



อินเตอร์เฟส Map

ทำการเก็บค่าคีย์คู่กับค่าข้อมูลของสมาชิกเสมอ โดยที่ค่าคีย์จะต้องไม่ซ้ำกัน แต่ค่าข้อมูลของสมาชิกสามารถที่จะซ้ำกันได้ ซึ่งอินเตอร์เฟส Map มีเมธอดที่สำคัญ ดังนี้

- `Object put(Object key, Object value) // insert or update`
- `Object remove(Object key)`
- `Object get(Object key)`
- `Set entrySet()`
- `Set keySet()`
- `int size()`
- คลาสสำคัญที่ implement อินเตอร์เฟส Map คือคลาส `HashMap`

ตัวอย่างการใช้ Map

```
import java.util.*;
public class SampleMap {
    public static void main(String args[]) {
        HashMap m = new HashMap();
        m.put("1", "C#");
        m.put("2", "Java");
        m.put("3", "Pascal");
        System.out.println("Removing Pascal");
        m.remove("3");
        System.out.println("The size is "+m.size());
        System.out.println("The first one is "+m.get("1"));
        m.put("3", "Java");
        System.out.println("The key of this map are "+ m.keySet());
        System.out.println("The contents are "+ m.entrySet());
    }
}
```

ผลลัพธ์การทำงานโปรแกรม

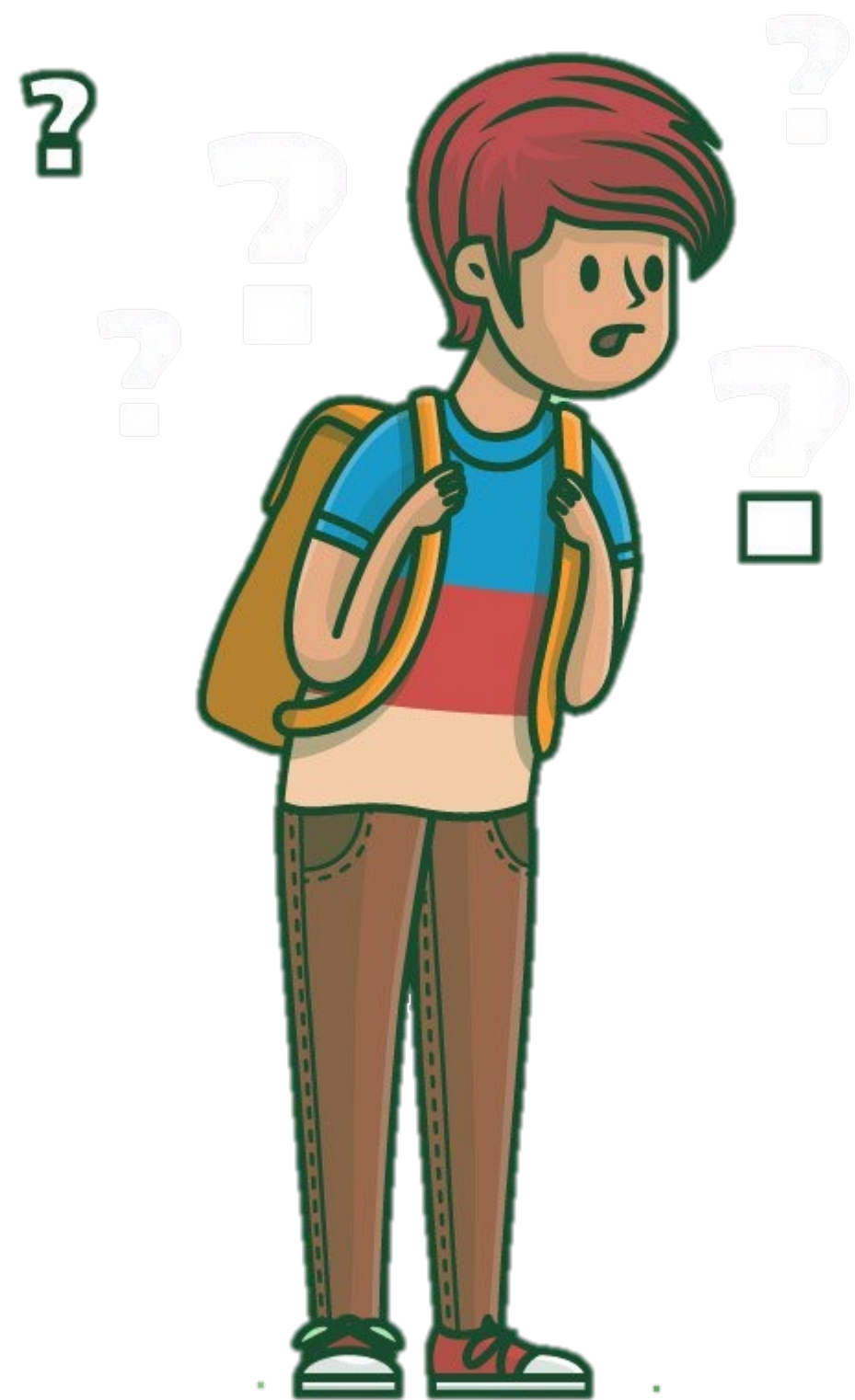
```
Removing Pascal
The size is 2
The first one is C#
The key of this map are [3, 2, 1]
The contents are [3=Java, 2=Java, 1=C#]
```

ตัวอย่างการใช้ Map

```
import java.util.*;
public class SampleMap {
    public static void main(String args[]) {
        HashMap m = new HashMap();
        m.put("A", "ant");
        m.put("B", "bat");
        System.out.println("Map Size: "+m.size());
        System.out.println("Map Value: "+m.get("A"));
        m.put("A", "apple");
        System.out.println("Map Size: "+m.size());
        System.out.println("Map Value: "+m.get("A"));
    }
}
```

```
Map Size:2
Map Value:ant
Map Size:2
Map Value:apple
```

หัวข้อ



- Array
- Collection
 - Set
 - List
 - Map
- Generic

Generic

ให้นักศึกษาลองพิจารณาโปรแกรมต่อไปนี้

```
public class Main {  
    public static void main(String args[]) {  
        ArrayList l = new ArrayList();  
        l.add(12);  
        l.add("Java");  
        l.add(15.6);  
        System.out.println("The size is "+l.size());  
        System.out.println("The contents are "+l);  
        System.out.println("The first one is "+(l.get(0)+2));  
    }  
}
```

ผลลัพธ์การทำงานโปรแกรม

```
The size is 3  
The contents are [12, Java, 15.6]  
Exception in thread "main" java.lang.RuntimeException:  
Uncompilable source code - Erroneous tree type: <any>  
at TestCollection.c1.main(c1.java:14)
```

Generic

จากปัญหาข้างต้นจะพบว่า ถ้าเราใส่ content เป็นชนิดข้อมูลอะไรก็ได้ **Collection** จะเก็บเป็นชนิด **Object** ให้ ดังนั้น เราจำเป็นต้องแปลงชนิดข้อมูลให้เป็น int ก่อนนำมาใช้งาน และเราจะรู้ได้ไงว่าเมื่อไหร่ควรแปลงเป็นชนิดอะไร ?

```
public class Main {  
    public static void main(String args[]) {  
        ArrayList l = new ArrayList();  
        l.add(12);  
        l.add("Java");  
        l.add(15.6);  
        System.out.println("The size is "+l.size());  
        System.out.println("The contents are "+l);  
        System.out.println("The first one is "+((int)l.get(0)+2));  
    }  
}
```

ผลลัพธ์การทำงานของโปรแกรม

```
The size is 3  
The contents are [12, Java, 15.6]  
The first one is 14
```


Generic

ดังนั้น เพื่อที่จะหลีกเลี่ยงปัญหาข้างต้น เราสามารถใช้คำสั่ง Generic ในการกำหนดชนิดข้อมูลของอ็อบเจกต์ที่อยู่ใน Collection ได้ ตัวอย่างเช่น

```
LinkedList<String> myList = new LinkedList<String>();
```

```
List<String> myList = new LinkedList<String>();
```

ตัวอย่างการใช้ Generic



```
public class SampleGeneric {  
  
    public static void main(String args[]) {  
        Set<Integer> scrSet = new HashSet<Integer>(); // ทำไมถึงไม่ Error  
        int[] myInt = {1, 3, 5, 7, 11};  
  
        for (int i : myInt) {  
            scrSet.add(i);  
        }  
  
        for (Integer num : scrSet) {  
            System.out.print(num + " ");  
        }  
        System.out.println();  
    }  
}
```

เปรียบเทียบ Generic ของ List กับไม่ Generic

```
public class c2 {  
    public static void main(String args[]) {  
        ArrayList<Student> l1 = new ArrayList<Student>();  
        l1.add(new Student("1111", "Thana", 3.0));  
        l1.add(new Student("2211", "Somchai", 2.10));  
        l1.add(new Student("3331", "Supansa", 3.1));  
        for(int i= 0;i<l1.size();i++)  
            System.out.println("l1["+ i +"] = "+l1.get(i).getName());  
  
        System.out.println("-----");  
  
        ArrayList l2 = new ArrayList();  
        l2.add(new Student("1111", "Thana", 3.0));  
        l2.add(new Student("2211", "Somchai", 2.10));  
        l2.add(new Student("3331", "Supansa", 3.1));  
        for(int i= 0;i<l2.size();i++)  
            System.out.println("l2["+ i +"] = "+ ((Student)l2.get(i)).getName());  
    }  
}
```

เปรียบเทียบ Array ของวัตถุกับ Generic ของ List



```

public class Main {
    public static void main(String args[]) {
        Student []s = new Student[3];
        s[0] = new Student("1111", "Thana", 3.0);
        s[1] = new Student("2211", "Somchai", 2.10);
        s[2] = new Student("3331", "Supansa", 3.1);
        for(int i= 0;i<s.length;i++)
            System.out.println("s["+ i +"] = "+s[i].getName());

        System.out.println("-----");

        ArrayList<Student> l = new ArrayList<Student>();
        l.add(new Student("1111", "Thana", 3.0));
        l.add(new Student("2211", "Somchai", 2.10));
        l.add(new Student("3331", "Supansa", 3.1));
        for(int i= 0;i<l.size();i++)
            System.out.println("l["+ i +"] = "+l.get(i).getName());

    }
}
    
```

การใช้ for และ Generic

นอกจากนี้ ยังสามารถใช้คำสั่ง for สำหรับการแจกแจงค่าของอ็อบเจกต์ประเภทคอลเล็กชันแทนที่จะใช้อินเตอร์เฟส Iterator ตัวอย่างเช่น

```
ArrayList<String> l = new ArrayList<String>();  
l.add("C#");  
l.add("Java");  
l.add("Pascal");  
  
for (String stringList : l) {  
    System.out.println(stringList);  
}
```