Java Collections

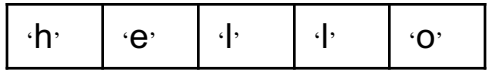
By Assoc. Prof. Rangsit Sirirangsi

Collections

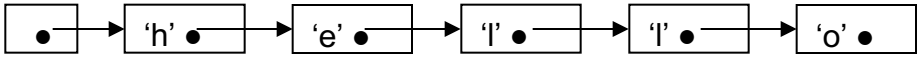
- Collection (บางครั้งถูกเรียกว่า *container*) ใช้สำหรับจัดกลุ่มสมาชิกในรูปของข้อมูลให้อยู่ในหน่วยเดียวกัน
- Collection ประกอบด้วยบริการต่าง ๆ เช่น การเพิ่มหรือลบข้อมูล และการจัดการกับสมาชิกต่าง ๆ ภายใน collection
- Collection ในจาวาได้ถูกจัดเตรียมไว้ในรูปของอินเตอร์เฟสและคลาสต่าง ๆ ภายในแพคเกจ java.util
- การใช้ Collection ในทางปฏิบัติผู้ใช้เพียงแต่เรียกใช้เมธอดให้เหมาะสมเท่านั้นก็พอ ไม่จำเป็นต้องรู้ถึงการทำงานภายใน
- ส่วน Collection Framework เป็นการรวบรวมสถาปัตยกรรมที่ใช้สำหรับการนำเสนอและเข้าถึงสมาชิกภายใน Collections

Storing data

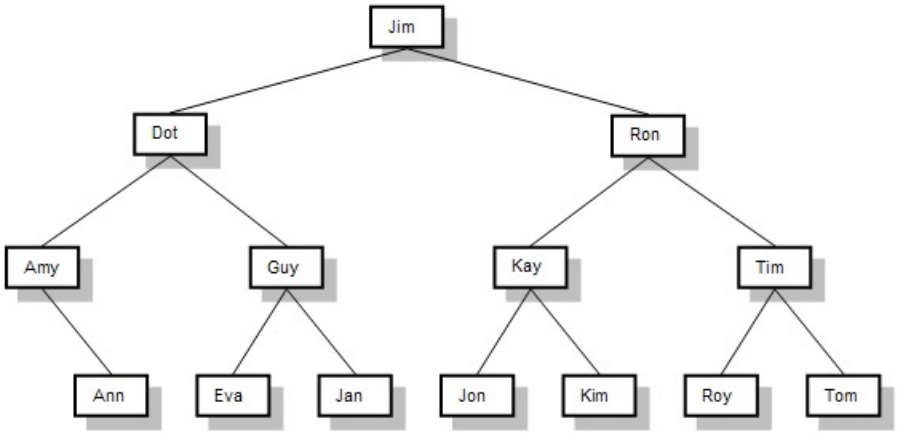
- ข้อมูลสามารถจัดเก็บในรูปของโครงสร้างข้อมูลได้หลายชนิด เช่น
- **Array Lists**



- **Sequential Lists**



- **Hash tables**
  - จัดเก็บโดยใช้ key ที่สามารถคำนวณได้จากข้อมูลที่ต้องการ
- **Trees**
  - เช่นข้อมูลแบบ Binary tree

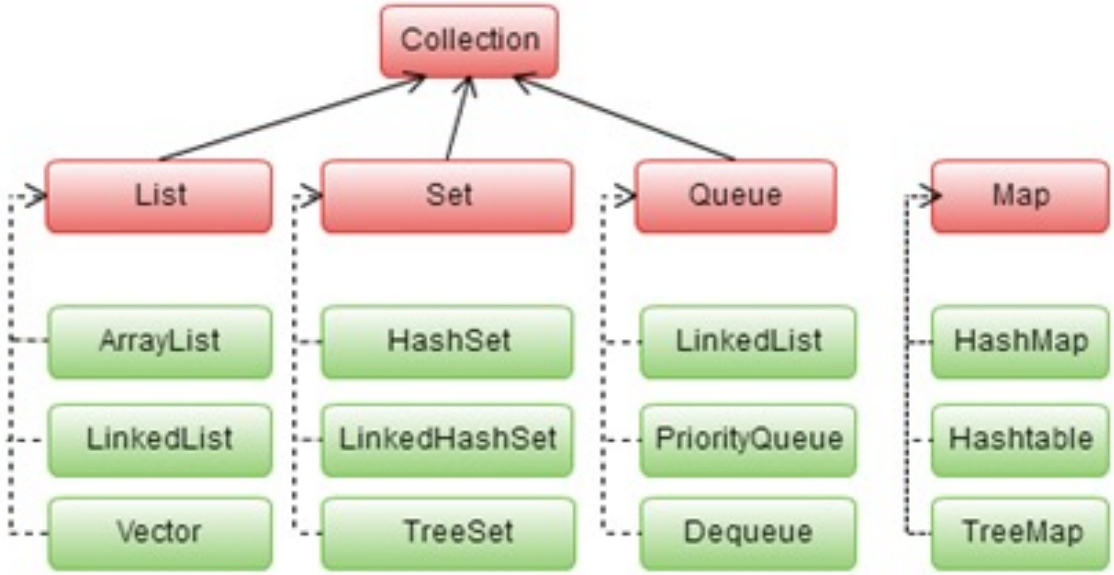


Benefits of a Collection Framework

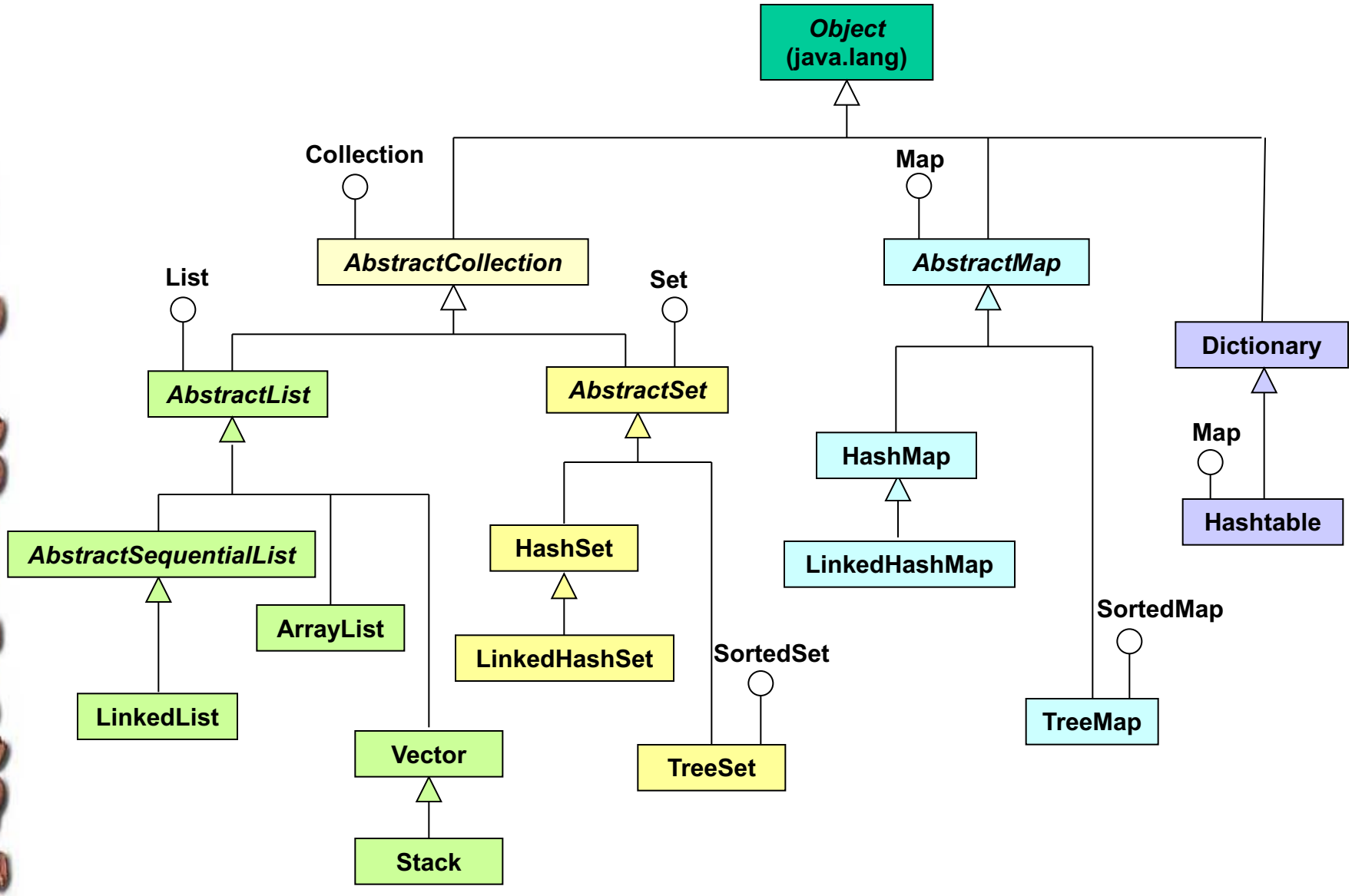
- ลดเวลาในการโปรแกรม
  - เป็นโครงสร้างข้อมูลและอัลกอลิทึมส์ที่มีประสิทธิภาพ
- เพิ่มความเร็วและคุณภาพของการโปรแกรม
- ลดเวลาในการเรียนรู้ APIs ใหม่ ๆ
- เป็น Framework ที่มีลักษณะเป็นหนึ่งเดียวกัน
- ใช้เป็น APIs สำหรับโปรแกรมประยุกต์ต่าง ๆ
- กระตุ้นให้เกิดการนำกลับมาใช้ใหม่ของซอฟต์แวร์
- โครงสร้างข้อมูลและอัลกอลิทึมส์แบบใหม่ ๆ

The Java Collection API

- **Collection**: ประกอบไปด้วย interfaces ต่าง ๆ
  - **Set**: ไม่สามารถเก็บค่าซ้ำกัน และลำดับไม่ใช่สิ่งสำคัญ
  - **List**: เก็บค่าซ้ำกันได้ลำดับเป็นสิ่งสำคัญ
  - **Map**: ถือเป็น “dictionary” ที่มีค่าเกี่ยวข้องกับ *keys* และ values ลำดับเป็นสิ่งที่ไม่สำคัญ



Java Collections



General Purpose Implementations

Interfaces	Implementations				
	Hash table	Resizable array	Tree (sorted)	Linked list	Hash table + Linked list
Set	HashSet		TreeSet (sorted)		LinkedHashSet
List		ArrayList		LinkedList	
Queue					
Map	HashMap		TreeMap (sorted)		LinkedHashMap

Interface: Collection

- Collection ถือเป็น interface หลักที่ถูกนำไปใช้สืบทอดตามลำดับชั้นของ Collection
- สามารถใช้ในการจัดเก็บ
  - ข้อมูลที่ซ้ำหรือไม่ก็ได้
  - ข้อมูลที่เรียงลำดับหรือไม่ก็ได้

```
public interface Collection {
    // Basic Operations
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(Object element);
    boolean remove(Object element);
    Iterator iterator();

    // Bulk Operations
    boolean containsAll(Collection c);
    boolean addAll(Collection c);
    boolean removeAll(Collection c);
    boolean retainAll(Collection c);
    void clear();

    // Array Operations
    Object[] toArray();
    Object[] toArray(Object a[]);
}
```

interface Collection : Methods

สรุปเมธอดที่ใช้ใน interface Collection
boolean add(O e) ใช้สำหรับการเพิ่มสมาชิกลงใน collection โดยเมธอดคืนค่า false ในกรณีที่ไม่สามารถเพิ่มสมาชิกได้
boolean addAll(Collection c) ใช้สำหรับการเพิ่มทุก ๆ สมาชิกจาก collection c ลงใน collection ปัจจุบันที่ถูกระบุ
boolean contains(Object e) เมธอดคืนค่า true ในกรณีที่ collection มีสมาชิกที่ถูกระบุอยู่แล้ว
boolean containsAll(Collection c) เมธอดคืนค่า true ในกรณีที่ collection ประกอบไปด้วยทุก ๆ สมาชิกปรากฏอยู่ภายใน Collection ที่ถูกระบุ
void clear() ใช้สำหรับการลบสมาชิกทั้งหมดออกจาก collection

interface Collection : Methods

สรุปเมธอดที่ใช้ใน interface Collection
boolean isEmpty() คืนค่า true ในกรณีที่ไม่มีสมาชิกอยู่ภายใน collection
Iterate iterator() คืนค่า iterator ที่ยอมให้ผู้ใช้สามารถเห็นค่าสมาชิกภายใน collection
boolean remove(Object o) ลบออปเจคในรูปของสมาชิกที่ถูกระบุออกจาก collection ในกรณีที่มีสมาชิกดังกล่าวอยู่จริง
boolean removeAll(Collection c) ลบทุก ๆ สมาชิกที่อยู่ภายใน Collection เป้าหมายที่เหมือนกับหรือมีอยู่ใน collection ที่ถูกระบุ

interface Collection : Methods

สรุปเมธอดที่ใช้ใน interface Collection
boolean retainAll(Collection c) ลบสมาชิกทั้งหมดภายใน Collection เป้าหมายที่ไม่เหมือนกับหรือมีอยู่ใน collection ที่ถูกระบุ ดังนั้นสมาชิกที่เหลืออยู่ของ collection ทั้งสองจะต้องเท่ากัน
int size() คืนค่าจำนวนสมาชิกที่อยู่ภายใน collection ปัจจุบัน
Object[] toArray() คืนค่าอะเรย์ที่ประกอบไปด้วยสมาชิกทั้งหมดของ collection ปัจจุบัน
Object[] toArray(Object a[]) คืนค่าอะเรย์ที่ประกอบไปด้วยสมาชิกทั้งหมดของ collection ปัจจุบัน รันไทม์ของอาร์เรย์ที่มีการคืนค่าจะต้องเป็นชนิดเดียวกับอาร์เรย์ a

interface List

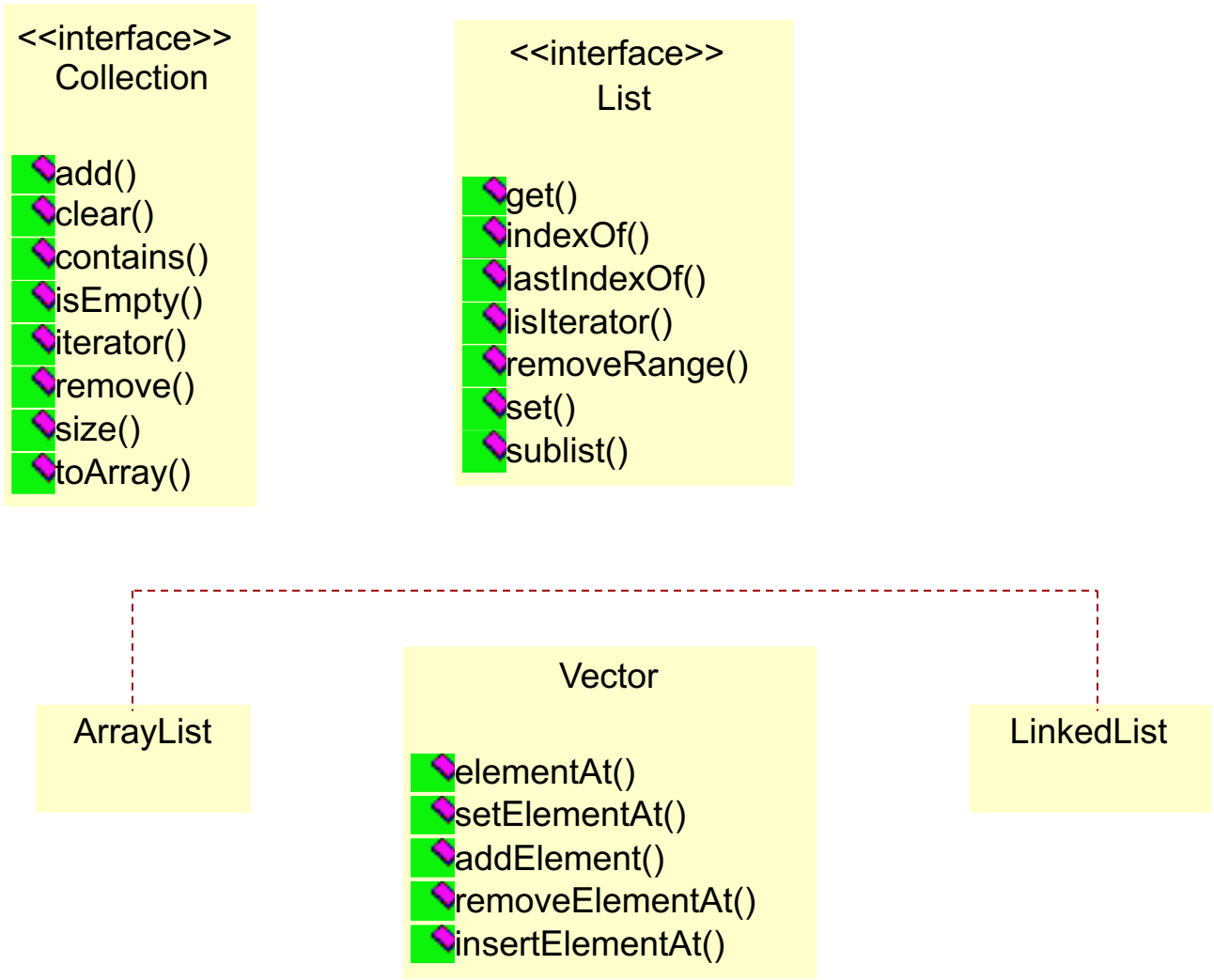
- List — เป็น collection ที่มีสมาชิกภายในเรียงลำดับกัน และสามารถมีค่าซ้ำกันได้
- ผู้ใช้สามารถควบคุมการเข้าถึงข้อมูลสมาชิกภายใน List โดยการระบุตำแหน่ง เช่นเดียวกับการทำงานของ Vector

```
public interface List {
 // Positional Access
    Object get(int index);
    Object set(int index, Object element);
    void add(int index, Object element);
    Object remove(int index);
    abstract boolean addAll(int index,
Collection c);

    // Search
    int indexOf(Object o);
    int lastIndexOf(Object o);

    // Iteration
    ListIterator listIterator();
    ListIterator listIterator(int index);

    // Range-view
    List subList(int from, int to);
}
}
```

Java Lists: ArrayList

- ความแตกต่างระหว่าง arrays และ ArrayLists:
  - Array มีขนาดคงที่ แต่ ArrayList สามารถเพิ่มหรือลดขนาดได้อัตโนมัติ
  - ดังนั้นจึงไม่จำเป็นต้องทราบขนาดของ ArrayList ก่อนการทำงาน
  - Arrays สามารถใช้งานได้กับข้อมูลชนิดพื้นฐานหรือออปเจค แต่ ArrayLists สามารถเก็บค่าได้เฉพาะออปเจค (Java 1.4)
  - การทำงานแบบไม่ synchronized

Class Diagram : ArrayList



Array Lists

Constructor Summary
ArrayList() สร้าง list ว่าง ๆ ที่มีขนาดเริ่มต้นเท่ากับ 10
ArrayList() สร้าง list ที่ประกอบด้วยสมาชิกของ collection ที่ถูกระบุ เพื่อให้สามารถคืนค่าได้โดยใช้ iterator ร่วมกับ collection
ArrayList() สร้าง list ว่างพร้อมกับระบุค่าเริ่มต้น

- วิธีการเรียกใช้ constructors (แบบเดิม):
  - import java.util.ArrayList;
  - ArrayList vec1 = new ArrayList();
  - ArrayList vec2 = new ArrayList(***initialCapacity***);

ArrayLists

สรุปเมธอดที่ใช้ในการเข้าถึงและเพิ่มข้อมูลใน list
size() คืนค่าจำนวนสมาชิกที่อยู่ภายใน list
get(int index) คืนค่าสมาชิกที่ถูกระบุตำแหน่ง index ภายใน list
add(int index, java.lang.Object o) แทรกสมาชิกลงในตำแหน่ง ***index*** ที่ถูกระบุอยู่ภายใน list
add(java.lang.Object o) เพิ่มออปเจคโดยต่อท้ายสมาชิกที่อยู่ภายใน list
set(int index, java.lang.Object o) แทนที่สมาชิกที่ถูกระบุตำแหน่งโดย index ด้วยพารามิเตอร์ในรูปของออปเจคที่ต้องการ

Array Lists

สรุปเมธอดเพิ่มเติมที่ใช้ใน list
isEmpty () คืนค่า true ในกรณีที่ list ไม่มีสมาชิกอยู่ภายใน
clear () ลบทุก ๆ สมาชิกภายใน list.
contains (java.lang.Object o) คืนค่า true ในกรณีที่ list ปัจจุบันประกอบไปด้วยออปเจคที่ถูกระบุ
indexOf (java.lang.Object o) คืนค่าตำแหน่ง index ที่พบครั้งแรกของออปเจคที่ถูกระบุ
trimToSize() Trims the capacity of this ArrayList instance to be the list's current size.

Differences from Java 1.4

- ในจาวา 1.4 หรือต่ำกว่า ArrayLists ไม่สามารถใช้ในการจัดเก็บข้อมูลชนิดพื้นฐานได้
  - ดังนั้นการจัดเก็บข้อมูลของ ArrayList ต้องอยู่ในรูปของออปเจคเท่านั้น
  - **สิ่งสำคัญ:** เมื่อมีการรับค่าออปเจคออกจาก ArrayList ด้วยเมธอดใด ๆ จะต้องมีการแปลงค่าออปเจคนั้น ๆ ให้เป็นชนิดของข้อมูลที่ถูกจัดเก็บเป็นครั้งแรกเสมอ
  - ในกรณีที่จำเป็นต้องจัดเก็บข้อมูลชนิดพื้นฐาน (เช่น int, double) ผู้ใช้ต้องใช้ “wrapper” คลาสร่วมด้วยเสมอ เช่น Integer, Double เช่น

```
myIntList.add(new Integer(7));
```

  - เมื่อต้องการรับค่าจาก ArrayList จำเป็นต้องใช้คลาส wrapper อีกเช่นกัน

```
int x = ((Integer) myIntList.get(i)).intValue();
```

Differences from Java 1.0- 1.5

<table>
<tr><td>Java 1.0
8 packages
212 classes</td><td>Java 1.1
23 packages
504 classes</td><td>Java 1.2
59 packages
1520 classes</td><td>Java 1.3
77 packages
1595 classes</td><td>Java 1.4
103 packages
2175 classes</td><td>Java 1.5
131 packages
2656 classes</td></tr>
<tr><td rowspan="5"></td><td rowspan="3">New Events
Inner class
Object Serialization
Jar Files
International
Reflection
JDBC
RMI</td><td rowspan="2">JFC/Swing
Drag and Drop
Java2D
CORBA</td><td rowspan="2">JNDI
Java Sound
Timer</td><td>Regular Exp
Logging
Assertions
NIO</td><td>javax.activity, javax. management</td></tr>
<tr><td colspan="2">java.nio, javax.imageio, javax.net, javax.print, javax.security, org.w3c</td></tr>
<tr><td colspan="3">javax.naming, javax.sound, javax.transaction</td></tr>
<tr><td colspan="3">javax.accessibility, javax.swing, org.omg</td></tr>
<tr><td colspan="4">java.math, java.rmi, java.security, java.sql, java.text, java.beans</td></tr>
<tr><td colspan="6">java.applet, java.awt, java.io, java.lang, java.net, java.util</td></tr>
</table>

Generics

- ในจาวา 1.4 หรือก่อนหน้านั้น เมื่อมีการรับค่าสมาชิกจาก collection ผู้ใช้จำเป็นต้องทำการ cast ก่อนเสมอ
  - การ Cast ที่ไม่ถูกต้องอาจก่อให้เกิดปัญหาตามมา
  - การ Cast ถือเป็น unsafe - อาจก่อให้เกิดความล้มเหลวในขณะ runtime
  - นอกจากนั้นแล้วยังไม่สามารถเห็นได้ชัดเจนว่าข้อมูลชนิดใดที่ถูกนำเข้าหรือออกจาก collection
- แต่การทำงานแบบ Generics ช่วยให้คอมไพเลอร์ทราบว่าข้อมูลชนิดใดที่ถูกจัดเก็บโดย collection
  - คอมไพเลอร์ทำหน้าที่ casts แทนผู้ใช้
  - ช่วยให้ช่วงเวลาในการคอมไพล์เป็นแบบ type safety
  - ได้มาจากแนวคิดของ C++ templates

Generics

- นอกจากนั้นแล้วในจาวาเวอร์ชัน 1.4 หรือก่อนหน้า ArrayList จะถูกใช้เฉพาะการจัดเก็บออปเจคเท่านั้น ไม่สามารถใช้เก็บข้อมูลแบบพื้นฐานได้
  - แต่ด้วยเหตุผลของความ compatibility ของการทำงานแบบเดิม ยังคงสามารถทำได้ตามปกติ แต่จะมีข้อความแจ้งเตือนผู้ใช้อยู่เสมอ
- ส่วนในจาวา 5 ผู้ใช้สามารถเรียกใช้ ArrayList ได้โดยใช้รูปแบบใหม่ที่เรียกว่า generics syntax
- เป็นวิธีการที่มีการคอมไพล์ซ้ำด้วยชนิดข้อมูลที่ต้องการใช้
- การใช้งานเมื่อเปรียบเทียบกับแบบเดิม:
  - `List words = new ArrayList();` // java 1.4
  - `List <String> words = new ArrayList <String> ();` // java 5

Using an ArrayList

- ในกรณีที่ต้องการใช้รูปแบบ generic ให้ระบุเครื่องหมาย angle brackets ระหว่างชื่อของ ArrayList และชนิดของออปเจคที่ต้องการจัดเก็บ
- การใช้งานเมื่อเปรียบเทียบกับแบบเดิม:
  - ArrayList words = new ArrayList(); // java 1.4
  - ArrayList **<String>** words = new ArrayList **<String>** (); // java 5
- ในกรณีที่ต้องการใช้งานแบบเดิม (จาวา 1.4) แต่ไม่ต้องการข้อความแจ้งเตือน อาจใช้ <?> wildcard แทนได้ดังนี้
  - ตัวอย่างเช่น :

```
ArrayList<?> vec1 = new ArrayList<?>();
```

Accessing with and without generics

- เรียกใช้เมธอด get() ด้วยจาวา 1.4 :

```
ArrayList myList = new ArrayList();
myList.add("Some string");
String s = (String) myList.get(0);
```

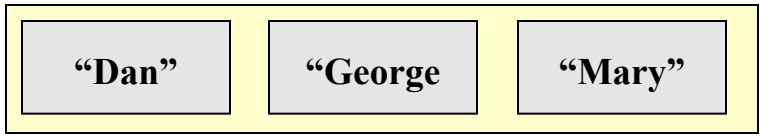
- เรียกใช้เมธอด get() ด้วยจาวา 5 :

```
ArrayList<String> myList = new ArrayList<String>();
myList.add("Some string");
String s = myList.get(0);
```

- ไม่จำเป็นต้องใช้ Cast อีกต่อไปเมื่อใช้งานร่วมกับ generic
- คลาสและ interfaces ที่สามารถใช้ generic ได้แก่ : Vector, ArrayList, LinkedList, Hashtable, HashMap, Stack, Queue, PriorityQueue, Dictionary, TreeMap , TreeSet

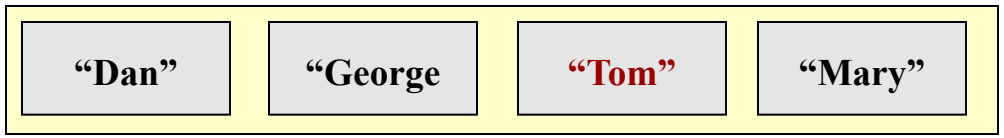
An example:

```
import java.util.ArrayList;

public class ArrayListTest
{
   public ArrayListTest
   {
      System.out.println("ArrayListTest");
      ArrayList aList = new ArrayList();
      aList.add("Dan");
      aList.add("George");
      aList.add("Mary");

      System.out.println("aList contains:");
      for(int x = 0; x < aList.size(); x++)
      {
         System.out.println(aList.get(x));
      }
   }
}
```



An example:

```
import java.util.ArrayList;

public class ArrayListTest
{
   public ArrayListTest
   {
      System.out.println("ArrayListTest");
      ArrayList<String> aList = new ArrayList <String>();
      aList.add("Dan");
      aList.add("George");
      aList.add("Mary");
      aList.add(2, “Tom”);
      System.out.println("aList contains:");
      for(int x = 0; x < aList.size(); x++)
      {
         System.out.println(aList.get(x));
      }
   }
}
```



The "For Each" Loop

- ในจาวาเวอร์ชัน 5.0 ได้มีการเพิ่มการทำงานของ for loop ชนิดใหม่ที่เรียกว่า *for-each* loop
  - การทำงานของ loop จะประมวลผลหนึ่งครั้งสำหรับแต่ละค่าของสมาชิกที่อยู่ภายใน collection
- รูปแบบของคำสั่ง

  for(***type varname*** : ***collection***) {...}

- type : ชนิดข้อมูลใน collection Varname : เป็นตัวแปรที่ใช้แสดงผล
- Collection : ระบุชื่อของ collection ที่ใช้

```
double[] array = {2.5, 5.2, 7.9, 4.3, 2.0};

for(double d: array )
{
    System.out.println(d);
}
```

For each loop (Collections)

- ตัวอย่างการใช้ for each สำหรับ ArrayList ที่จัดเก็บค่า Integer ออปเจค

```
import java.util.*;
public class ArrayList_ForEach1
{
public static void main(String []args)
{
    ArrayList<Integer> list = new ArrayList<Integer>();
            list.add(7);
            list.add(15);
            list.add(-67);

    for (Integer number: list){
            System.out.println(number);
  }
  }
}
```

Output : 7
15
-67

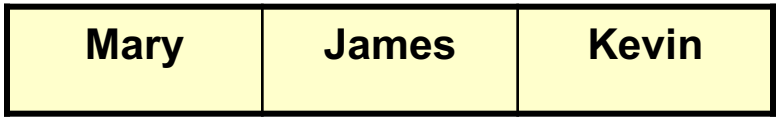
Using enhanced FOR loop:


```
import java.util.ArrayList;

public class ArrayList_ForEach2
{
   public static void main (String [] arg)
   {
      System.out.println("ArrayListTest");
      ArrayList<String> aList = new ArrayList <String>();
      aList.add(new String("Dan"));
      aList.add("George");
      aList.add("Mary");

      System.out.println("aList contains:");

         for(String e:aList)
         {
            System.out.println(e);
         }
      }
}
```

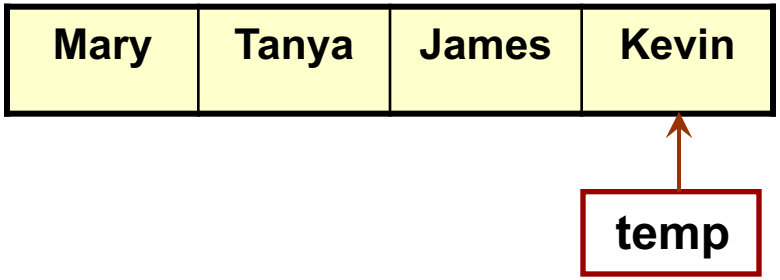
```
ArrayListTest
aList contains:
Dan
George
Mary
```

What happens ?

```
ArrayList<String> students = new ArrayList<String>();
students.add("Mary");
students.add("James");
students.add("Kevin");
```




```
students.add(1, "Tanya");

String temp = students.get(3);
System.out.println(temp);
```




```
students.remove(2);
```


```
students.set(1, "John");
System.out.println(students.size());
```

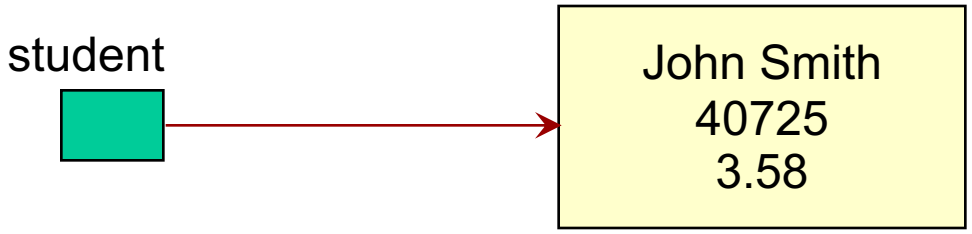
Vector with Generic

```
import java.util.*;
public class VectorDemo {
  public static void main(String[] args) {
   Vector<String> v = new Vector<String>();
    v.add("1");
    v.add("2");
    v.add("3");
    ArrayList<String> arrayList = new ArrayList<String>();
    arrayList.add("4");
    arrayList.add("5");

//append all elements of ArrayList to Vector
    v.addAll(arrayList);

   System.out.println("After appending all elements of ArrayList, Vector contains..");
    for(String temp:v)
         System.out.print("    " +temp);
   }
}
```

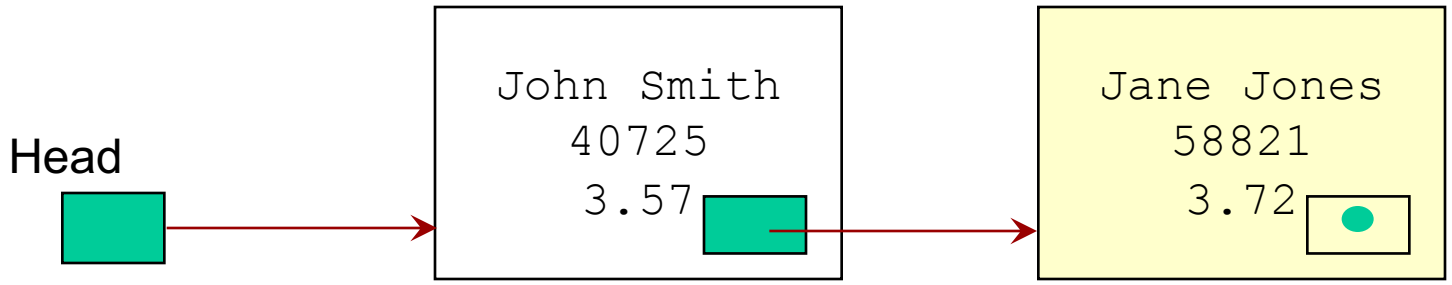
```
After appending all elements of ArrayList, Vector contains..
  1   2   3   4   5
```

Object References

- จาวาใช้ reference สำหรับการอ้างถึงแอดเดรสที่แท้จริงของออปเจค เช่น
  - Student student = new Student(“John Smith”, “40725”, 3.85);

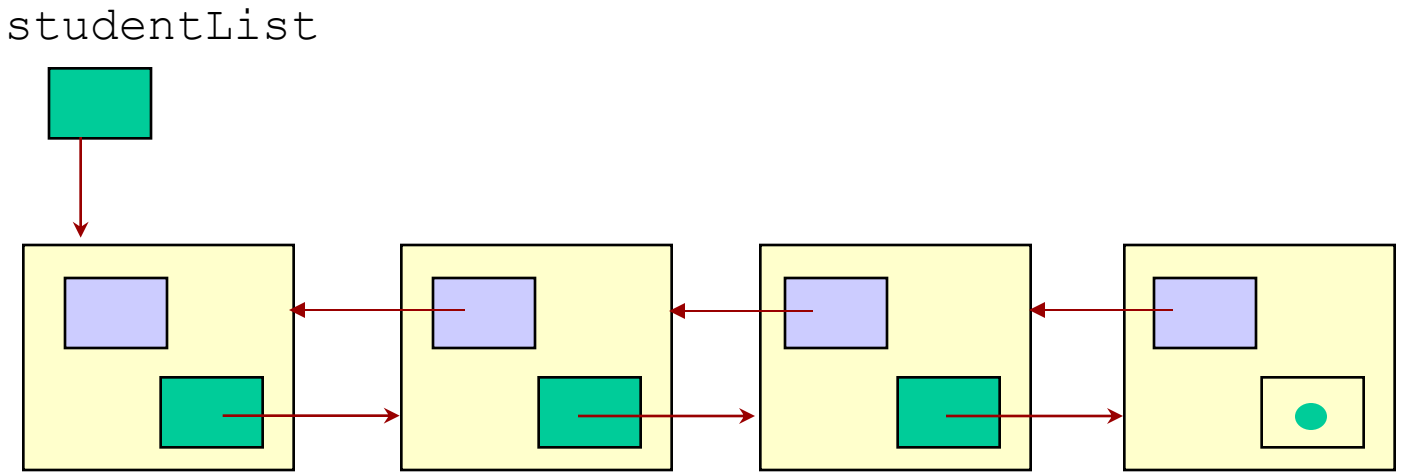


- References สามารถนำมาสร้าง *links* เพื่อเชื่อมกันระหว่างออปเจคได้ เช่น คลาส Student ประกอบไปด้วย reference ไปยังออปเจคอื่น ๆ ของ Student



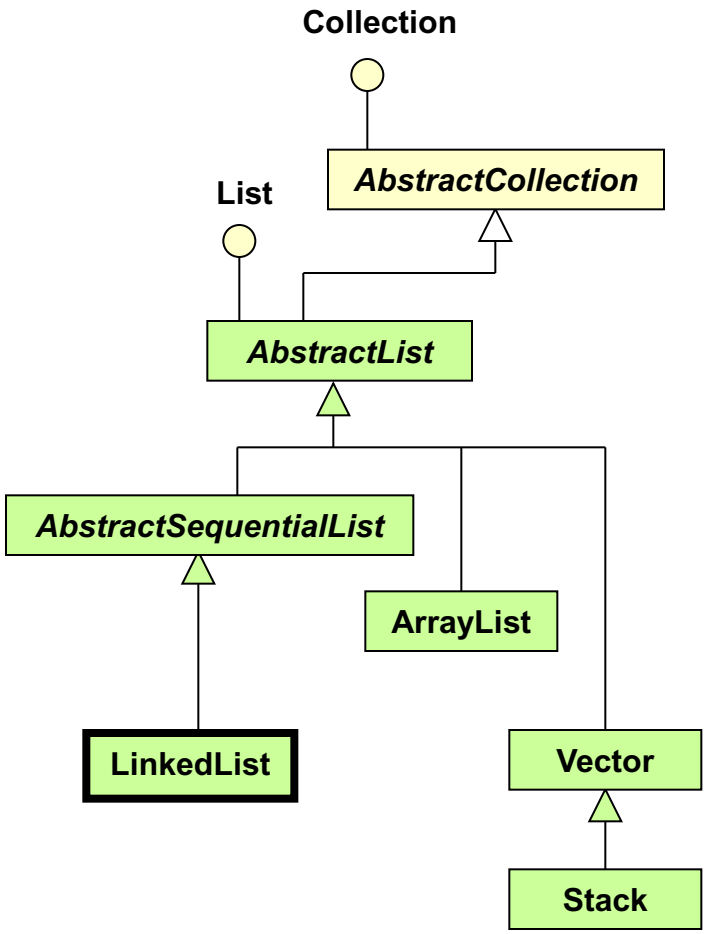
References as Links

- แม้ว่าการใช้ References ในการสร้าง Single Linked List เป็นสิ่งที่สามารถทำได้ แต่อย่างไรก็ตามการทำงานยังมีประสิทธิภาพที่ไม่สูงนัก
- ดังนั้นจึงได้มีการพัฒนา Double Linked List ขึ้นมาใหม่ เพื่อแก้ไขปัญหาดังกล่าว



Java Lists

- LinkedList : ถูกพัฒนาขึ้นในรูปของ double linked list
  - มีผลทำให้การแทรกข้อมูล และการลบข้อมูลในช่วงกลาง ๆ ของ list ทำได้อย่างรวดเร็ว
- LinkedList ได้จัดเตรียมเมธอดประเภท get, remove และ insert สมาชิกที่จุดเริ่มต้นและสิ้นสุดของ list
- LinkedList ไม่สนับสนุนการทำงานแบบ synchronized



Interface: List

- List ใช้กับการจัดเก็บข้อมูลที่มีการเรียงลำดับกัน
- Lists สามารถประกอบไปด้วยข้อมูลที่ซ้ำกันได้
- ผู้ใช้สามารถเข้าถึงข้อมูลของ List ได้โดยใช้ค่าตำแหน่ง index

```
public interface List extends Collection {
   // Positional Access
   Object get(int index);
   Object set(int index, Object element);
   void add(int index, Object element);
   Object remove(int index);
   abstract boolean addAll(int index, Collection c);

   // Search
   int indexOf(Object o);
   int lastIndexOf(Object o);

   // Iteration
   ListIterator listIterator();
   ListIterator listIterator(int index);

   // Range-view
   List subList(int from, int to);
}
```

Linked List Method

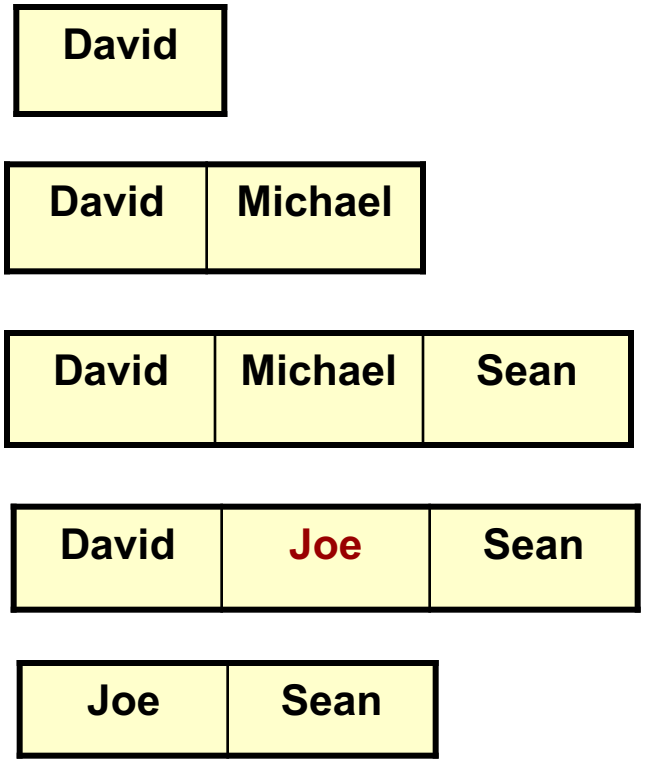
สรุปเมธอดที่ใช้ในการเข้าถึงและเพิ่มข้อมูลใน list
LinkedList() คอนสรัคเตอร์สำหรับ Linked list ว่าง ๆ
LinkedList() คอนสรัคเตอร์ที่ประกอบไปด้วยสมาชิกภายใน collection ทุถูกระบุ เพื่อให้สามารถคืนค่าได้โดยใช้ collection's iterator
addAll(Collection c) ทุก ๆ สมาชิกทั้งหมดที่อยู่ภายใน collection ถูกระบุจะถูกนำไปต่อท้าย list เพื่อให้สามารถคืนค่าได้โดยใช้ collection's iterator
addAll(int index, Collection c) แทรกทุก ๆ สมาชิกลงใน collection ที่ถูกระบุ โดยเริ่มต้นจากตำแหน่งที่ถูกระบุ
addFirst(Object o) แทรกสมาชิกที่ต้องการลงในตำแหน่งแรกของ list
addLast(Object o) เพิ่มสมาชิกที่ต้องการต่อท้าย list

Linked List Method

สรุปเมธอดเพิ่มเติมที่ใช้ใน list	
getFirst()	คืนค่าสมาชิกลำดับแรกที่อยู่ภายใน list
getLast()	คืนค่าสมาชิกลำดับสุดท้ายที่อยู่ภายใน list
removeFirst()	ลบและคืนค่าสมาชิกลำดับแรกออกจาก list
removeLast()	ลบและคืนค่าสมาชิกลำดับสุดท้ายออกจาก list
remove(int index)	ลบค่าสมาชิกตามตำแหน่งที่ถูกระบุภายใน list
remove(Object o)	คืนค่าสมาชิกที่ถูกพบเป็นครั้งแรกภายใน list

Linked List Example

```
import java.util.*;

class MyLinkedList {

    public static void main(String[] args) {

      LinkedList listIT = new LinkedList();

      listIT.add("David");

      listIT.add("Michael");

      listIT.add("Sean");

      listIT.set(1, “Joe");

      listIT.remove(0);

      System.out.println(listIT);
 }
}
```



Illustration/example

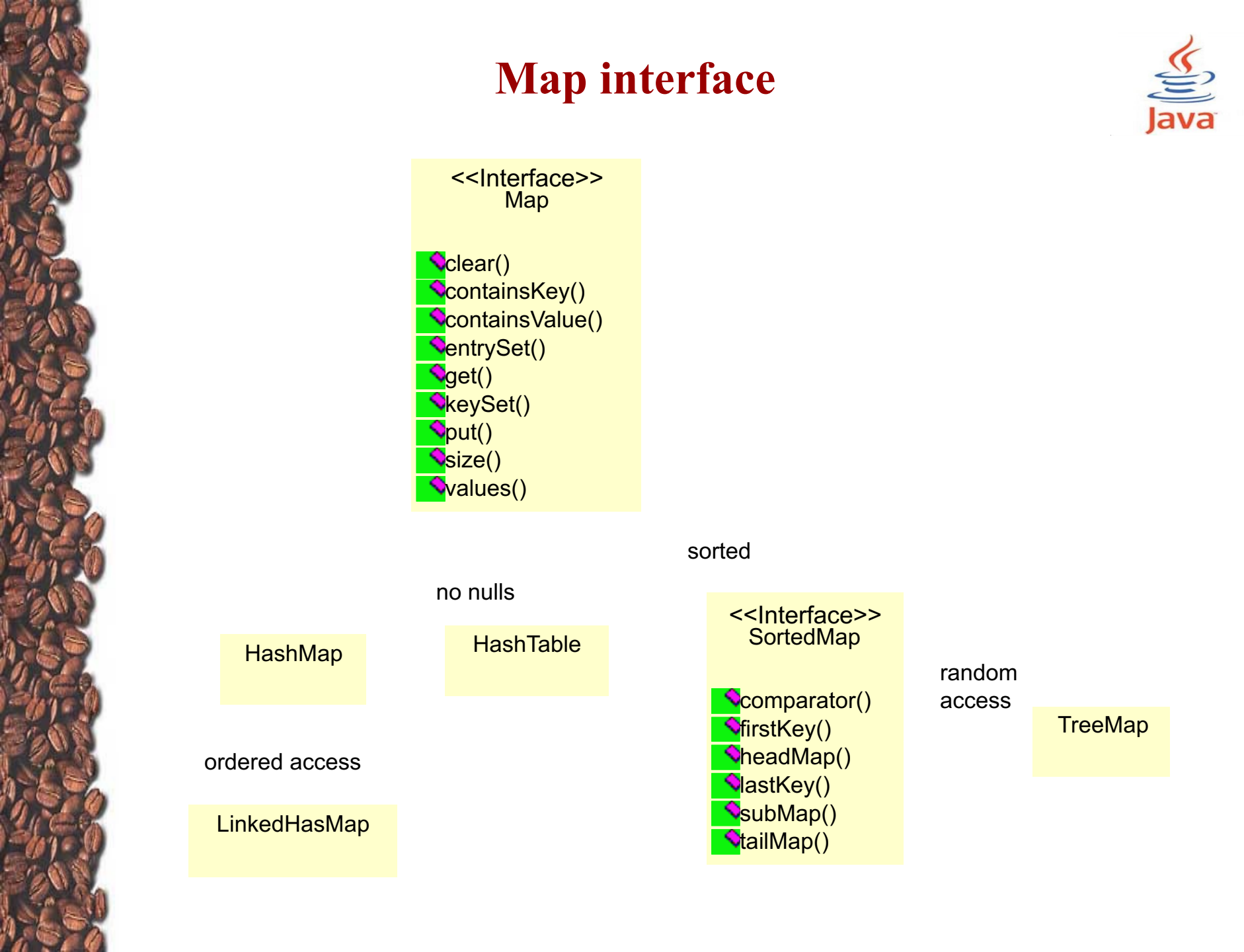
Operation	List's contents	Return value
1. Initialiaze(List)	<empty>	-
2. Add(0,A,List)	A	-
3. Add(0,B,List)	B A	-
4. Add(1,C,List)	B C A	-
5. Set(1,N,List)	B N A	-
6. Add(1,D,List)	B D N A	-
7. Remove(A,List)	B D N	A
8. Set(3,I,List)	B D N I	-
9. Remove(D,List)	B N I	D
10. Remove(N,List)	B I	N

Interface: Map

- map บางครั้งอาจถูกอ้างถึงในรูปของ **dictionary**
- Map เป็นออปเจคที่ใช้สำหรับการแปลง keys ให้เป็น values
- Maps เก็บค่า key ที่ไม่ซ้ำกันแต่ values ซ้ำกันได้
- Hashtables เป็นตัวอย่างการใช้งานที่สืบทอดมาจาก Maps

```
public interface Map {
    // Basic Operations
    Object put(Object key, Object value);
    Object get(Object key);
    Object remove(Object key);
    boolean containsKey(Object key);
    boolean containsValue(Object value);
    int size();
    boolean isEmpty();

    void putAll(Map t);
    void clear();

    public Set keySet();
    public Collection values();
    public Set entrySet();

    // Interface for entrySet elements
    public interface Entry {
        Object getKey();
        Object getValue();
        Object setValue(Object value);
    }
}
```

Map interface



Map interface Method

- เมธอดดังต่อไปนี้ กำหนดให้ k เป็นออปเจค และใช้สำหรับนำเสนอ key ส่วน v เป็นออปเจคที่ใช้นำเสนอ value และ m ใช้นำเสนอ Map

สรุปเมธอดที่ใช้ใน Map interface
clear() ลบทุก ๆ สมาชิกออกจาก map
containKey(k) คืนค่า true ในกรณีที่ map ประกอบด้วย mapping สำหรับ key k
containsValue(v) คืนค่า true ในกรณีที่ map มีการ maps keys ตั้งแต่หนึ่งหรือมากกว่าไปยังค่า v
entrySet() คืนค่า Set view ของการ mappings ที่ถูกจัดเก็บไว้ใน map

Map interface Method

สรุปเมธอดที่ใช้ใน Map interface	
get(k)	คืนค่า true ในกรณีที่มีการmaps ค่า key k
isEmpty()	คืนค่า true ในกรณีที่ map ไม่มีการแปลงค่า key-value
keySet()	คืนค่า Set view ของคีย์ที่อยู่ภายใน map
put(k,v)	เพิ่มค่า v พร้อมกับคีย์ k ภายใน map
putAll(m)	สำเนาค่าจาก map m ไปยัง map ที่เรียกใช้
remove(k)	ลบค่า mapping สำหรับคีย์ k จาก map ปัจจุบัน
values()	คืนค่า Collection view ของค่าที่อยู่ภายใน map

Interface: Map

- ประกอบไปด้วย 3 มุมมอง
  - **มุมมองแบบ Entry set** ข้อมูลทั้งหมดที่เข้าสู่ map เช่น
    - Set entrySet()
  - **มุมมองแบบ key set** กลุ่มของ keys ทั้งหมดภายใน map ที่ถูกกำหนดค่าเนื่องจาก keys มีลักษณะ unique เช่น
    - Set keySet()
  - **มุมมองจาก value collection** เป็นกลุ่มของทุก ๆ values ภายใน map มีลักษณะเป็น collection เนื่องจาก keys ที่แตกต่างกันสามารถ map ลงในค่าเดียวกันได้ เช่น
    - Collection values()

Map Example

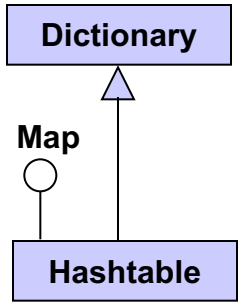

```
import java.util.*;
public class MapDemo {
        public static void main(String args[]) {
                Map<String, String> m = new HashMap<String, String>();
                m.put("1", "C#");
                m.put("2", "Java");
                m.put("3", "Php");

                 System.out.println(" Size of Map = " m.size());
                 System.out.println("The keys of this Map :  " +m.keySet());
                 System.out.println("The values of this Map :   "+ m.entrySet());
        }
}
```

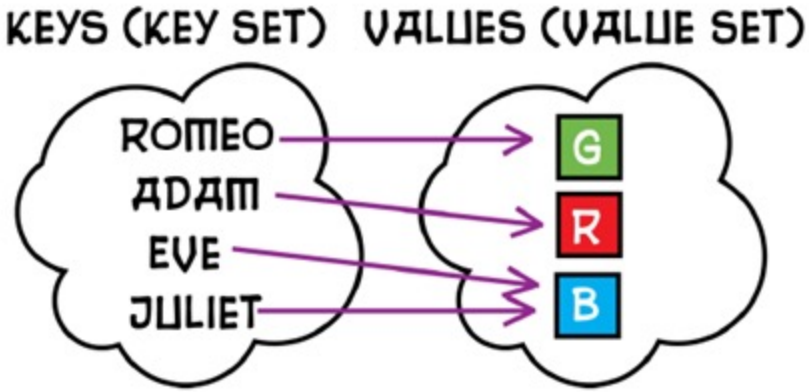
output :

```
Size of Map = 3
The keys of this Map : [3, 2, 1]
The values of this Map : [3=Php, 2=Java, 1=C#]
```

The class Hashtable

- Hashtable ใช้สำหรับเก็บข้อมูลเป็นลักษณะคู่ลำดับ <key,value>
- ในคลาส Hashtable จะใช้ Hash code ของ key ทำหน้าที่เป็นดัชนีในการค้นหาค่าของ value
- คลาส Hashtable สามารถใช้งานแบบ Multi-Thread ได้ (Thread-Safe)





Hashing Concept

- เป็นวิธีการจัดเก็บและค้นหาข้อมูลอีกแบบหนึ่งที่ประกอบไปด้วย
  - ฟังก์ชันสำหรับการ hash (h)
  - อะเรย์ที่มีขนาดเท่ากับ N
- Collision เกิดขึ้นในกรณีที่ 2 keys ถูก map ลงในอะเรย์ในตำแหน่งเดียวกัน
- ฟังก์ชัน hash (h) แปลงค่า keys ที่มีชนิดข้อมูลที่กำหนดไว้ให้อยู่ในรูปของค่า integer ตั้งแต่ {0, N-1}
- ตัวอย่างเช่น:

  $$h(x) = x \bmod N$$

- mod : ฟังก์ชันที่ใช้หาค่าเศษของ Key Value

Hashing Function

- กำหนดให้ ความจุในการจัดเก็บข้อมูลที่ได้ ไม่เกิน 20 เรคคอร์ด
- จากข้อกำหนดเลือก N สำหรับการ mode = 19
- **3211 2134 1532 1123 1143 1135 2432 7213 8812 5321**

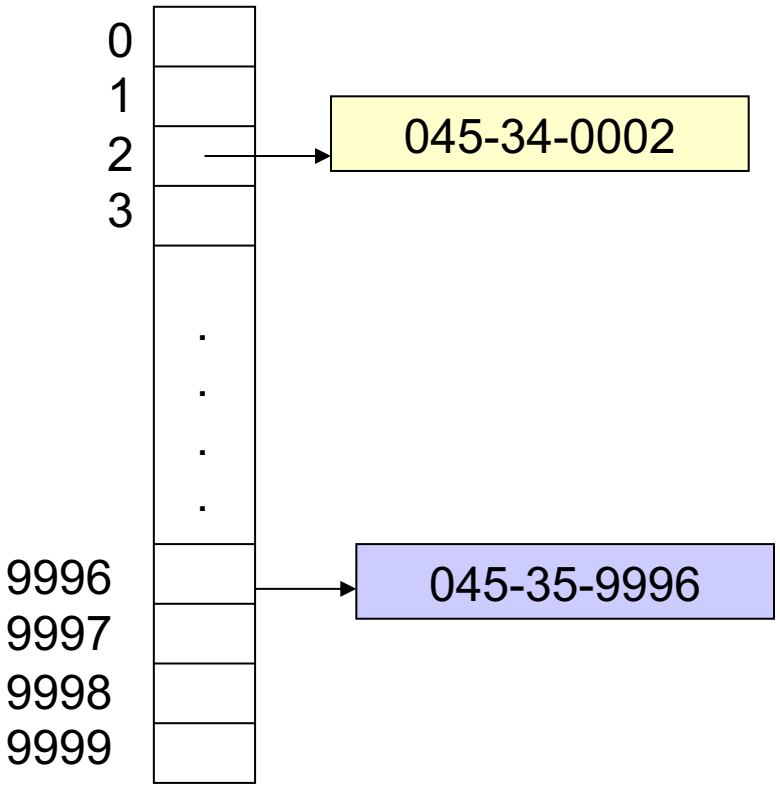
```
3211 : 0
2134 : 6
1532 : 12
1123 : 2
1143 : 3
1135 : 14
2432 : 0
7213 : 12
8812 : 15
5321 : 1
```

h(x) = x mod N

Key:	Value
0	3211
1	5321
2	1123
3	1143
4	
5	
6	2134
7	
8	
9	
10	
11	
12	1532
13	
14	1135
15	8812
16	
17	
18	
19	

Hash Table: Example

- กำหนดให้ Item(name, ssn) เมื่อ ssn เป็นตัวเลข 9 หลัก
- Hash table N = 10000.
- Hash function: เฉพาะตัวเลข 4 หลักสุดท้ายของ x
- การจัดเก็บค่า HashTable จะมีลักษณะดังต่อไปนี้



Example use of a Hashtable

- การเพิ่มค่าเข้าสู่ Hashtable

  table.put(*key, value*);

- การรับค่าจาก Hashtable

  value = table.get(*key*);

Key	Value
ABC	abc
XYZ	xyz
MNO	mno

```
import java.util.*;

public class HashtableUser {
    public static void main(String[] args) {
        Hashtable table = new Hashtable();
        table.put("ABC", "abc");
        table.put("XYZ", "xyz");
        table.put("MNO", "mnos");
        System.out.println("two -> " + table.get("XYZ"));
        System.out.println("deux -> " + table.get("xyz"));
    }
}
```

```
two -> xyz
deux -> null
```

Hashtable (1.5)


```
import java.util.*;
class HashtableDemoGen {
  public static void main(String args[]) {
    Hashtable<String,String> hashtable = new Hashtable<String,String>();
    hashtable.put("apple", "red");
    hashtable.put("strawberry", "red");
    hashtable.put("lime", "green");
    hashtable.put("banana", "yellow");
    hashtable.put("orange", "orange");

    for (Enumeration<String> e = hashtable.keys() ; e.hasMoreElements() ;) {
      String k = e.nextElement();
      String v = hashtable.get(k);
      System.out.println("key = " + k +  "; value = " + v);
     }

    System.out.print("\nThe color of an apple is: ");
    String v = hashtable.get("apple");
    System.out.println(v);
  }
}
```

```
key = banana; value = yellow
key = apple; value = red
key = orange; value = orange
key = lime; value = green
key = strawberry; value = red

The color of an apple is: red
```