Java 課程

第11章：物件容器

大綱

- Collection類
- Map類

簡介List介面

- List介面是java.util.Collection介面的子介面
- Collection介面是java.lang.Iterable子介面

```
package java.lang;
import java.util.Iterator;
public interface Iterable<T> {
    Iterator<T> iterator();
}
```

- 在Java SE的API中找不到任何實作Iterator的類別
  - Iterator會根據實際的容器資料結構來迭代元素
  - 而容器的資料結構實作方式對外界是隱藏的

簡介List介面

- Collection介面繼承了Iterator介面

```
package java.util;
public interface Collection<E> extends Iterable<E> {
    int size();
    boolean isEmpty();
    boolean contains(Object o);
    Iterator<E> iterator();
    <T> T[] toArray(T[] a);
    boolean add(E o);
    boolean remove(Object o);
    boolean containsAll(Collection<?> c);
    boolean addAll(Collection<? extends E> c);
    boolean removeAll(Collection<?> c);
    boolean retainAll(Collection<?> c);
    void clear();
    boolean equals(Object o);
    int hashCode();
}
```

簡介List介面

- 每個加入List中的元素是循序加入的，並可指定索引來存取元素

```
package java.util;
public interface List<E> extends Collection<E> {
    ....
    boolean addAll(int index, Collection<? extends E> c);
    E get(int index);
    E set(int index, E element);
    void add(int index, E element);
    E remove(int index);
    int indexOf(Object o);
    int lastIndexOf(Object o);
    List<E> subList(int fromIndex, int toIndex);
    ....
}
```

簡介List介面

- List可以使用陣列（Array）或是鏈結串列（Linked List）來實作這個特性
- 對於循序加入與存取，使用ArrayList的效率比較好
- 對於經常變動元素排列順序的需求，使用LinkedList會比較好

ArrayList

- 使用陣列結構實作List資料結構
- 可以使用索引來快速指定物件的位置
- 於快速的隨機取得物件來說，使用ArrayList可以得到較好的效能
- 若要從中間作移除或插入物件的動作，會需要搬動後段的陣列元素以重新調整索引順序，所以速度上就會慢的多

ArrayList

```
Scanner scanner = new Scanner(System.in);

List<String> list = new ArrayList<String>();

System.out.println("輸入名稱(使用quit結束)");
while(true) {
    System.out.print("# ");
    String input = scanner.next();
    if(input.equals("quit"))
        break;
    list.add(input);
}

System.out.print("顯示輸入: ");
for(int i = 0; i < list.size(); i++)
    System.out.print(list.get(i) + " ");
System.out.println();
```

ArrayList

- 如果您的目的是要循序取出容器中所有的物件，則您可以使用Iterator

```
Iterator iterator = list.iterator();
while(iterator.hasNext()) { // 還有下一個元素嗎？
    // 使用 next() 取得下一個元素
    System.out.print(iterator.next() + " ");
}
```

- Iterator的實例是在ArrayList中根據陣列的結構而實作的，但您不用理會實作細節

ArrayList

- 使用「增強的for迴圈」（Enhanced for loop）來直接遍訪List的所有元素

```
// 使用foreach來遍訪List中的元素
for(String s : list) {
    System.out.print(s + " ");
}
```

LinkedList

- 如果經常從容器中作移除或插入物件的動作，使用LinkedList會獲得較好的效能
- LinkedList使用鏈結串列（Linked list）實作了List介面
- addFirst()、addLast()、getFirst()、getLast()、removeFirst( )、removeLast()等

```
private LinkedList<String> linkedList;
public StringStack() {
    linkedList = new LinkedList<String>();
}
public void push(String name) {
    // 將元素加入串列前端
    linkedList.addFirst(name);
}
public String top() {
    // 取得串列第一個元素
    return linkedList.getFirst();
}
public String pop() {
    // 移出第一個元素
    return linkedList.removeFirst();
}
public boolean isEmpty() {
    // 串列是否為空
    return linkedList.isEmpty();
}
```

LinkedList

```
private LinkedList<String> linkedList;

public StringQueue() {
    linkedList = new LinkedList<String>();
}

public void put(String name) {
    linkedList.addFirst(name);
}

public String get() {
    return linkedList.removeLast();
}

public boolean isEmpty() {
    return linkedList.isEmpty();
}
```

HashSet

- 實作了java.util.Set介面，Set介面繼承了Collection介面
- List容器中的物件允許重複，但Set容器中的物件都是唯一的
- Set容器有自己的一套排序規則
- HashSet容器中的物件是否相同時，會先比較hashCode()方法傳回的值是否相同，如果相同，則再使用equals()方法比較，如果兩者都相同，則視為相同的物件

HashSet

```
Set<String> set = new HashSet<String>();

set.add("caterpillar");
set.add("momor");
set.add("bush");
// 故意加入重複的物件
set.add("caterpillar");

// 使用 Iterator 顯示物件
Iterator iterator = set.iterator();
while(iterator.hasNext()) {
    System.out.print(iterator.next() + " ");
}
System.out.println();
```

HashSet

```
Set<String> set = new LinkedHashSet<String>();

set.add("caterpillar");
set.add("momor");
set.add("bush");

// 使用 enhanced for loop 顯示物件
for(String name : set) {
    System.out.print(name + " ");
}
System.out.println();
```

TreeSet

- TreeSet實作Set介面與java.util.SortedSet介面
- TreeSet是Java SE中唯一實作SortedSet介面的類別
- 自動依字典順序進行排列的動作

TreeSet

```
Set<String> set = new TreeSet<String>();

set.add("justin");
set.add("caterpillar");
set.add("momor");

// 使用 enhanced for loop 顯示物件
for(String name : set) {
    System.out.print(name + " ");
}
System.out.println();
```

TreeSet

- 自訂一個實作Comparator介面的類別

```
public class CustomComparator<T> implements Comparator<T> {
    public int compare(T o1, T o2) {
        if (((T) o1).equals(o2))
            return 0;
        return ((Comparable<T>) o1).compareTo((T) o2) * -1;
    }
}
```

```
Comparator<String> comparator =
                new CustomComparator<String>();
Set<String> set =
                new TreeSet<String>(comparator);
```

HashMap

- Map的特性即「鍵-值」（Key-Value）匹配
- java.util.HashMap實作了Map介面，
- HashMap在內部實作使用雜湊（Hash），很快的時間內可以尋得「鍵-值」匹配

HashMap

```
Map<String, String> map =
                    new HashMap<String, String>();
String key1 = "caterpillar";
String key2 = "justin";
map.put(key1, "caterpillar 的訊息");
map.put(key2, "justin 的訊息");

System.out.println(map.get(key1));
System.out.println(map.get(key2));
```

- 可以使用values()方法返回一個實作Collection的物件，當中包括所有的「值」物件

```
Map<String, String> map =
          new HashMap<String, String>();

map.put("justin", "justin 的訊息");
map.put("momor", "momor 的訊息");
map.put("caterpillar", "caterpillar 的訊息");

Collection collection = map.values();
Iterator iterator = collection.iterator();
while(iterator.hasNext()) {
    System.out.println(iterator.next());
}
System.out.println();
```

HashMap

```
Map<String, String> map =
            new LinkedHashMap<String, String>();

map.put("justin", "justin 的訊息");
map.put("momor", "momor 的訊息");
map.put("caterpillar", "caterpillar 的訊息");

for(String value : map.values()) {
    System.out.println(value);
}
```

TreeMap

- java.util.TreeMap實作Map介面與java.util.SortedMap介面
- SortedMap提供相關的方法讓您有序的取出對應位置的物件，像是 firstKey()、lastKey()等方法
- TreeMap是Java SE中唯一實作SortedMap介面的類別

TreeMap

```
Map<String, String> map =
        new TreeMap<String, String>();
map.put("justin", "justin 的訊息");
map.put("momor", "momor 的訊息");
map.put("caterpillar", "caterpillar 的訊息");

for(String value : map.values()) {
    System.out.println(value);
}
```

TreeMap

- 如果對物件有一套排列順序，要定義一個實作java.util.Comparator介面的物件

```
CustomComparator<String> comparator =
        new CustomComparator<String>();
Map<String, String> map =
        new TreeMap<String, String>(comparator);
```