第13章集合

資訊科技系
林偉川

集合

- 了解Collection具體類別所使用的基本資料結構
- 熟悉Collection Framework介面的繼承架構
- 熟悉各個Collection具體類別的特性及應用
- 熟悉Generic
- 了解Arrays與Collections類別的常用方法
- 了解覆蓋equals()方法的規則
- 了解equals()和hashCode()之間的關係
- 了解Comparable介面的功能

2

Collection中所使用的基本資料結構

- 陣列(array)
- 鏈結串列(linked list)
- 樹(tree)
- 雜湊表(hash table)

3

13-1-1 陣列

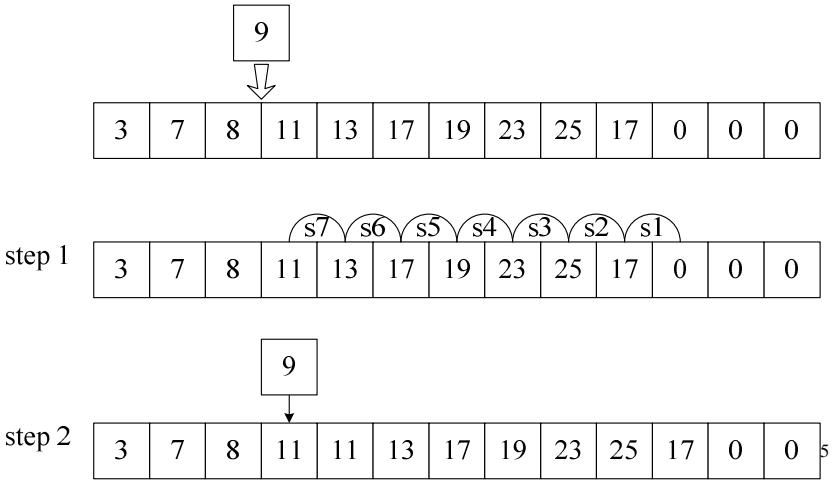
陣列的重要特性：

- 存取元素的速度快。
- 要在某位置插入或移除元素值時，並不方便。
- 陣列長度為固定，欲改變長度必須重新建立另一個陣列。

4

13-1-1 陣列

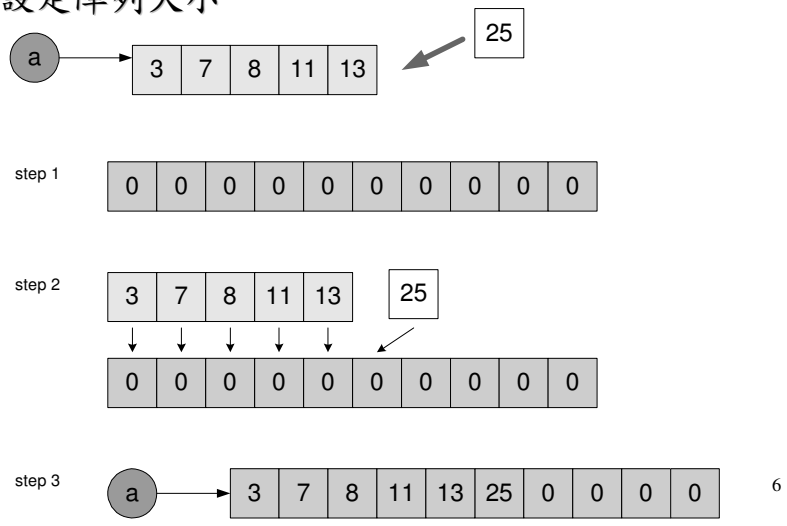
在陣列中插入一個元素值



5

13-1-1 陣列

重新設定陣列大小



6

13-1-2 鏈結串列

- 可以鏈結的類別

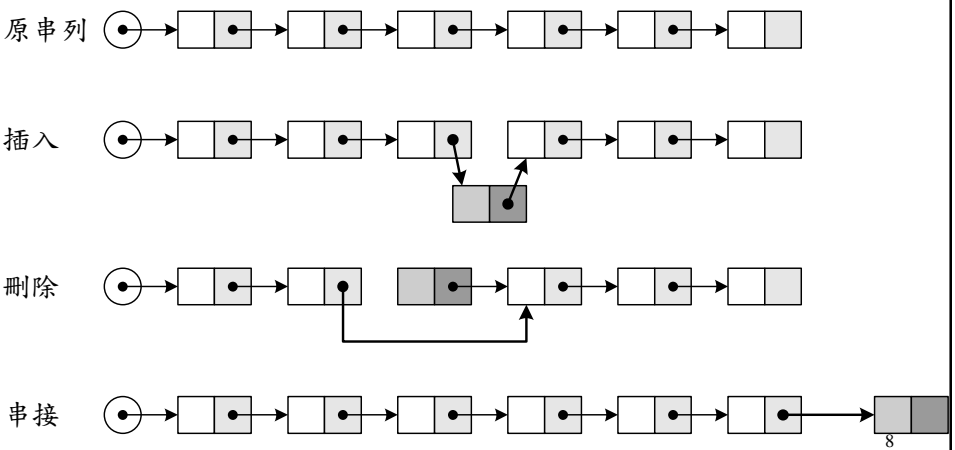
```
class LinkedObj {
        Object data;          //節點資料
        LinkedObj next;               //下一個節點
}
```

- 鏈結串列的重要特性：
  - 插入或刪除節點很方便。
  - 變更鏈結串列的長度也很方便。
  - 存取節點的速度較慢。

7

13-1-2 鏈結串列

- 鏈結串列的插入、刪除和串接節點



8

13-1-3 樹

- 實體可以做為二元樹的節點的類別

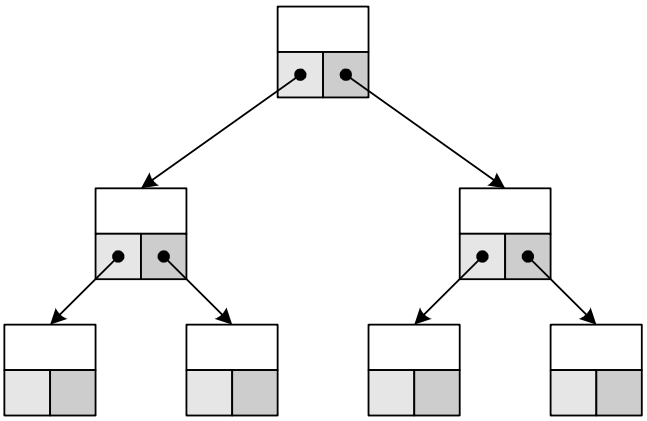
```
class TreeNode {
        Object data;
        TreeNode left;
        TreeNode Mid;
        TreeNode right;
}
```

  - 樹的優點是有排序的功能，缺點和鏈結串列一樣，節點資料的存取比較慢。

9

13-1-3 樹

- 二元樹



10

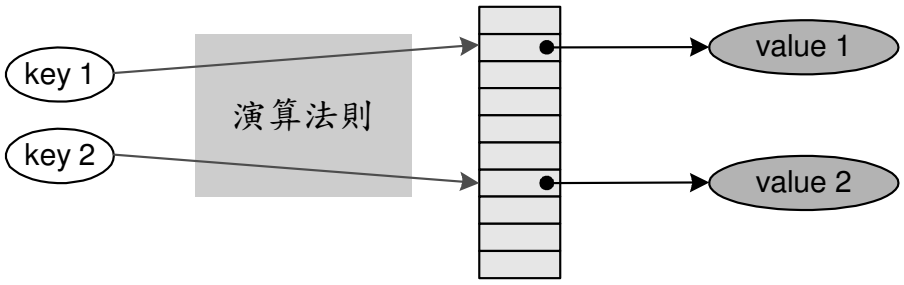
13-1-4 雜湊表

- 雜湊表(hash table)的資料是以鍵值對(key value paired)的形式存在。
- 加入一個元素時必須同時給予一個key和一個value，透過key就可以取得value。
- 雜湊表的特點：
  - 存取資料的速度快。
  - 較浪費記憶體空間。

11

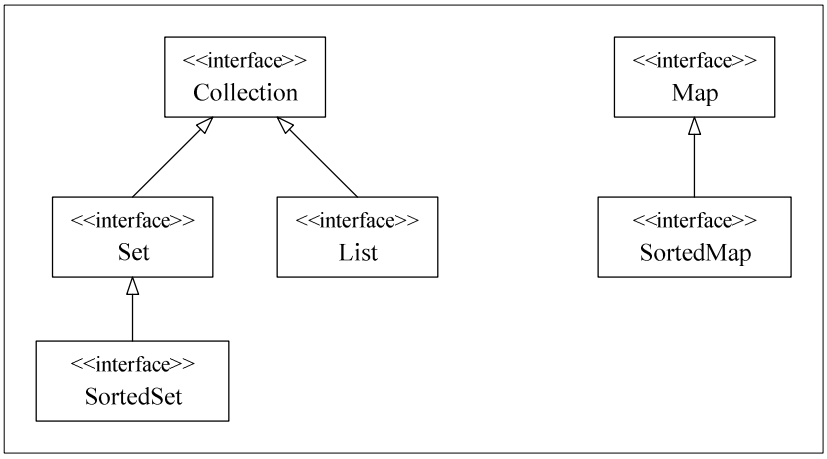
13-1-4 雜湊表

- 雜湊表的實作方式



12

Collection Framework內的介面架構



13

13-2-1 Collection介面

- Collection介面定義的集合，其元素可以為無順序(non-ordered)和可重複(repetition allowed)。
- Collection兩個延伸介面List和Set，分別保有Collection的不同特性。
- Set的特點為，其元素不可重複。SortedSet介面繼承Set介面，且宣告了排序的方法。
- List介面繼承Collection介面，然而它是有順序的，而且是可以重複的。

14

13-2-2 Map介面

- Map介面和Collection介面沒有繼承的關係。
- Map介面的元素是「鍵值對」(key-value pairs)。
- Map中的key和value皆為參照變數，而且key不能重複，一個key對應到一個value。
- Map中的元素是沒有順序的。
- SortedMap介面為Map的子介面，其為可排序的集合。

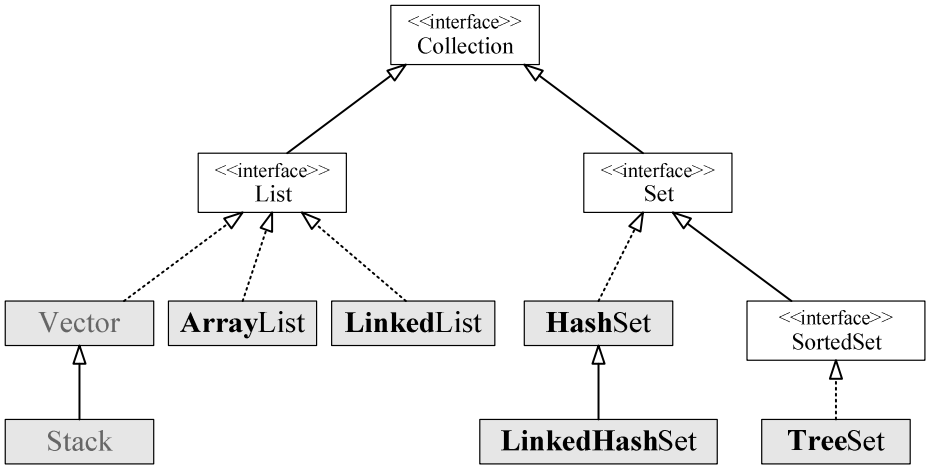
15

請注意Collection具體類別的相關特性：

- 類別實作的介面。
- 類別使用的資料結構。
- 元素可否重複。
- 無序或有序(加入的先後順序，或有排序功能)。
- 是否為執行緒同步(Thread safe class)。
- 有哪些常見的應用。

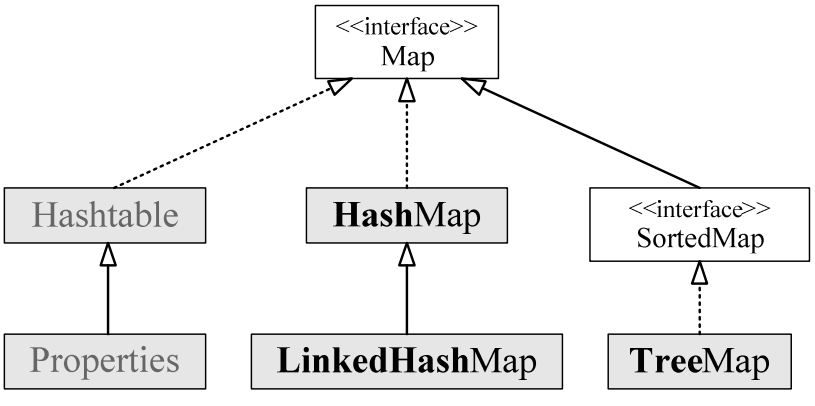
16

實作Collection延伸介面的具體類別



17

實作Map介面的具體類別



18

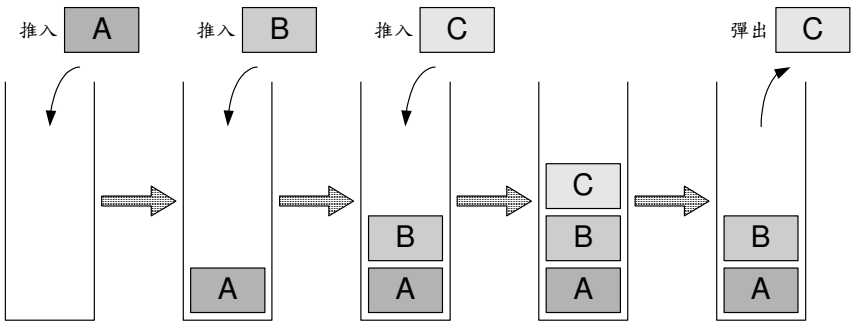
13-3-1 實作List介面的具體類別

- ArrayList使用array資料結構實作List。保有陣列的特性，存取元素的效率佳，但不利於插入或移除元素，且重定陣列長度效率差。
- LinkedList使用linked list實作List。保有linked list的特性，存取元素的效率較差，但利於插入元素、移除元素和改變長度。
- Vector和ArrayList很像，都是以array實作List。兩者最大的不同是在於：Vector為「執行緒同步類別」，而ArrayList則否。

19

13-3-1 實作List介面的具體類別

- **Stack**為Vector的延伸類別，所以同樣為有序、執行緒同步類別。Stack類別是堆疊的抽象資料型別(Abstract Data Type)。



20

13-3-2 泛型(Generic)

- Generic提供了限定Collection物件的元素型別之功能，也去除感覺累贅的型別轉換敘述。
- Generic使用尖括號(和小於、大於使用的符號相同)，尖括號內放的是元素限定的型別名稱。
- 整個尖括號放在Collection型別名稱之後，包含使用建構子時。

21

13-3-2 泛型(Generic)

- Generic的例子

```
LinkedList <Integer> list;
list = new LinkedList <Integer>();
```

```
LinkedList <Integer> list = new LinkedList <Integer>();
```

22

- java.util.Collection介面的原始碼定義：

```
package java.util;
public interface Collection<E> extends Iterable<E> {
   int size();
   boolean isEmpty();
   boolean contains(Object o);
   Iterator<E> iterator();
   Object[] toArray();
   <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();
}
```

13-3-3 實作Set介面的具體類別

- HashSet使用hash table實作Set。元素沒有順序，而且元素不可重複存在HashSet物件內。其key和value是使用相同的物件，也就是被加入的物件，存取效率佳。
- LinkedHashSet除了使用hash table，還使用linked list實作Set，使之成為有序集合。元素的順序是依照加入的順序，而且元素不可重複存在。
- TreeSet使用tree實作SortedSet，所以元素在加入集合時，就會和既有的元素比較，以放在適當的位置，不同型別不能互相比較，加入不同型別產生例外

24

13-3-4 實作Map介面的具體類別

- HashMap使用hash table實作Map，此集合中key-value pairs順序和放入的順序無關，而且key無法重複，Map沒有add()方法，而改用put()和get()方法。
- LinkedHashMap使用hash table和linked list實作Map，此集合中key-value pairs順序就是放入的順序，而key同樣無法重複。
- TreeMap使用tree實作Map，此集合中key-value pairs順序是依照key在集合中的排序而定，而key同樣無法重複。

25

13-3-4 實作Map介面的具體類別

- HashMap、LinkedHashMap、TreeMap類別沒有iterator()方法可以取得Iterator物件，但可透過keySet()取得包含key的Set型別物件，再利用Set物件取得Iterator物件，或使用value()取得Collection型別物件，再利用Collection物件取得Iterator物件
- Hashtable和HashMap的功能差不多，不同處在於Hashtable為執行緒同步，HashMap則不是。
- Properties為Hashtable的延伸類別。Properties的key和value應該為String型別，而且使用setProperty()和getProperty()取代put()和get()。

26

13-3-5 具體類別的整體比較

具體類別	實作介面	資料結構	元素重複性	元素順序	執行緒同步
ArrayList	List	array	可	依加入順序	否
LinkedList	List	linked list	可	依加入順序	否
Vector	List	array	可	依加入順序	是*
Stack	List	array	可	依加入順序	是*
HashSet	Set	hash table	不可	無關加入順序	否
LinkedHashSet	Set	hash table linked list	不可	依加入順序	否
TreeSet	SortedSet	tree	不可	排序	否
HashMap	Map	hash table	key不可	無關加入順序	否
LinkedHashMap	Map	hash table linked list	key不可	依加入順序	否
TreeMap	SortedMap	tree	key不可	依key排序	否
Hashtable	Map	hash table	key不可	無關加入順序	是*
Properties	Map	hash table	key不可	無關加入順序	是*

27

13-4-1 Arrays

Arrays的靜態方法	說明
List **asList**(Object[] a)	將Object陣列a轉換成List物件，然後回傳。
int **binarySearch**(int[] a, int key)	在已排序的a陣列中以二次搜尋法，找出key的索引值。若找不到key則回傳 –1。此方法有許多因應不同型別陣列的多載方法。
boolean **equals**(int[] a, int[] a2)	比較兩陣列a和a2，若兩陣列中的元素值皆相等則回傳true。此方法有許多因應不同型別陣列的多載方法。
void **fill**(int[] a, int val)	將a陣列中的所有的元素值設定為val。此方法有許多因應不同型別陣列的多載方法。
void **sort**(int[] a)	對陣列a排序。此方法有許多因應不同參數的多載方法。

28

13-4-2 Collections

- Collections中的靜態方法大都針對List，因為List不像Set和Map有可排序的類別。
- Collections另外提供將「非執行緒同步」集合包裹成「執行緒同步」集合的方法，這些方法在多執行緒程式裡會顯得相當方便。

29

Collections的常用靜態方法	說明
int **binarySearch**(List list, Object key)	以二次搜尋法，找出key的索引值。
void **copy**(List dest, List src)	將src序列的所元素複製給dest序列。
boolean **replaceAll**(List list, Object oldVal, Object newVal)	將list序列中的oldVal換成newVal物件。若oldVal不包含在list內則回傳false。
void **reverse**(List list)	將list序列的順序顛倒。
void **shuffle**(List list)	亂數重排list序列中元素的順序。
void **sort**(List list)	對list序列排序。
List **synchronizedList**(List list)	取得一執行緒同步的集合。
Map **synchronizedMap**(Map m)	
Set **synchronizedSet**(Set s)	
SortedMap **synchronizedSortedMap**(SortedMap m)	
SortedSet **synchronizedSortedSet**(SortedSet s)	

30

13-5-1 是否相等

- Object.equals()的定義如下：

```
public boolean equals(Object obj){
        return (this == obj);
}
```

- 當兩個物件obj1和obj2使用equals()比較後認定為相等時，他們由hashCode()方法所取得的雜湊碼也必須相同。所以當兩物件相等時，以下兩個運算式的結果都必須為true。

```
obj1.equals(obj2)== obj2.equals(obj1)== true
obj1.hashCode()== obj2.hashCode()
```

31

13-5-1 是否相等

- hashCode()方法在使用hash table的集合中顯得相當重要，因為hash table是以key.hashCode()取得的hash code value(雜湊值)。

- 當你覆蓋equals()方法時，也必須覆蓋hashCode()方法，讓兩者的行為一致。

32

13-5-1 是否相等

- Integer類別中定義的equals()

```
public boolean equals(Object obj){
  if(obj instanceof Integer){
    return value == ((Integer)obj).intValue();
  }
  return false;
}
```

- Integer類別中定義的hashCode()

```
public int hashCode(){
        return value;
}
```

33

- String類別的equals()方法

```
public boolean equals(Object anObject){
    if (this == anObject){
        return true;                                    //參照相同
    }
    if (anObject instanceof String){
        String anotherString = (String)anObject;
        int n = count;                                  //字串長度
        if (n == anotherString.count){
            char v1[] = value;                          //字元陣列
            char v2[] = anotherString.value;
            int i = offset;                     //第一字元的位置
            int j = anotherString.offset;
            while (n-- != 0){
                if (v1[i++] != v2[j++])
                    return false;               //有任何字元不同
            }
            return true;                        //所有字元都相同
        }
    }
    return false;                               //不是String型別
}
```

13-5-1 是否相等

- String類別的hashCode()方法

```
public int hashCode(){
      int h = hash;                              //取得字串的雜湊碼
      if(h == 0){                          //若雜湊碼為0表示尚未計算
            int off = offset;              //第一個字元的位置
            char val[] = value;            //取得字元陣列
            int len = count;               //字串的長度
            for (int i = 0; i < len; i++){
                  //計算雜湊碼的規則
                  h = 31*h + val[off++];
            }
            hash = h;
      }
      return h;
}
```

s[0]*31^(n-1)+ s[1]*31^(n-2)+ ... + s[n-1]

13-5-1 是否相等

- 自訂類別的equals()和hashCode()
  - 定義equals()時，可以一一比對各個屬性是否相等。屬性大多為現成的類別或基本資料型別，因此，就可以使用現成的類別的equals()方法。
  - 定義hashCode()時，只要將屬性的雜湊碼相互做位元互斥(XOR，^)運算即可。

13-5-2 較大或較小

- TreeSet和TreeMap都有自動排序的功能，既然要能排序必定有比較大小的規則。
- 使用tree實作的collection，其元素必須是相同型別的物件，而且必須實作**Comparable**介面。
- **String類別和基本型別的包裝器**(Boolean除外)實作了Comparable介面。

37

13-5-2 較大或較小

- Comparable介面只宣告一個抽象方法compareTo()

```
int compareTo(Object o);
```

- 實作compareTo()方法必須遵守
  - 不同型別的物件不能比較
  - 若物件本身比傳入的物件小時，則回傳負值。
  - 若物件本身比傳入的物件大時，則回傳正值。
  - 若不大也不小時，回傳0。

38