第十六章
Java collection集合物件

認識collection架構

認識並學習如何建立各種集合物件

學習利用Iterator介面的函數走訪元素

利用ListIterator介面的函數走訪元素

1

為何需要Collection(集合物件) ?

- 如何儲存多個同性質的資料，可用array(陣列)，也可以使用collection(集合)。
  - List: 與陣列很像
  - Set: 不按特定的方式排序，且沒有重復物件
  - Map: 每一個元素都包含key and value
- 在撰寫程式時, **常需要處理一群同性質資料**的集合, 為了方便程式處理這樣的集合, Java API 特別在 java.util 套件中提供了一組介面和類別, 讓我們可建立這類資料集合的物件, 這個物件就稱之為 **Collection (集合物件)**。

2

List 放值

```
import java.util.ArrayList;

public class list {
  public static void main(String args[]) {
   ArrayList list = new ArrayList();
    list.add(5);
    list.add(4);
    list.add("alex");
    list.add(true);
   System.out.println(list);
}
}
```

```
[5, 4, alex, true]
```

3

課堂練習

ex16_1.java

建立一個儲存學生物件的集合,儲存3個學生姓名並列印出此集合。

```
/* output----------------------------
 [大雄, 靜香, 胖虎]
 ----------------------------------*/
```

4

List 取值

```
import java.util.ArrayList;

public class list {
public static void main(String args[]) {
 ArrayList list = new ArrayList();
   list.add(5);
   list.add(4);
   list.add("alex");
   list.add(true);
 System.out.println(list);
   int n1=list.get(0);
   String s=list.get(2);
 System.out.println(n1+" "+s);
 }
}
```

```
[5, 4, alex, true]
5 alex
```

5

課堂練習

- **ex16_2.java**

List 取全部的值

將 Max, Alex, 5, true, 8 ，共5個值放入list，而後以 for迴圈取出list中所有的值。

```
/*output----------------------------------
        Max  Alex  5  true  8
 ----------------------------------------*/
```

6

List 的限定存放資料類型 (Generics)

```
import java.util.ArrayList;

public class list {
public static void main(String args[]) {
ArrayList<Integer> list = new ArrayList<Integer> ();
  list.add(5);
  list.add(4);
  list.add(1);
  list.add(2);
 System.out.println(list);
  int n1=list.get(0);
  int n2=list.get(1);
 System.out.println(n1+" "+n2);
}
}
```

☺同一個集合，放不同型態的資料，很容易造成取用錯誤，因此在定義類別之後，加入Generics<>(泛型)來限定放入資料的型態。

```
[5, 4, 1, 2]
5 4
```

7

List的特性

- 有順序性
- 有索引值
- 允許重複資料

```
import java.util.ArrayList;

public class list {
public static void main(String args[]) {
ArrayList<Integer> list = new ArrayList<Integer> ();
  list.add(5);
  list.add(4);
  list.add(1);
  list.add(2);
  list.add(1);
 System.out.println(list);
 }
}
```

```
[5, 4, 1, 2, 1]
```

8

Set

- 希望資料不要有重複放的時候
  - HashSet: 沒有index值，沒有次序的概念，資料不重複(例如:學號，座號)

```
import java.util.HashSet;
public class studentlist {
    public static void main(String[] args) {
        HashSet <Integer> hset = new HashSet<Integer>();
        hset.add(3);
        hset.add(1);
        hset.add(2);
        hset.add(1);
       System.out.println(hset);
    }
}
```

[1, 2, 3]

10

Map

- 有值對應Key

```
import java.util.HashMap;
public class studentlist {
    public static void main(String[] args) {
    HashMap<String, String> hmap= new HashMap();
        hmap.put("3","大雄");
        hmap.put("1","宜靜");
        hmap.put("2","胖虎");
        hmap.put("1","小花");
       System.out.println(hmap);
    }
}
```

{1=小花, 2=胖虎, 3=大雄}

11

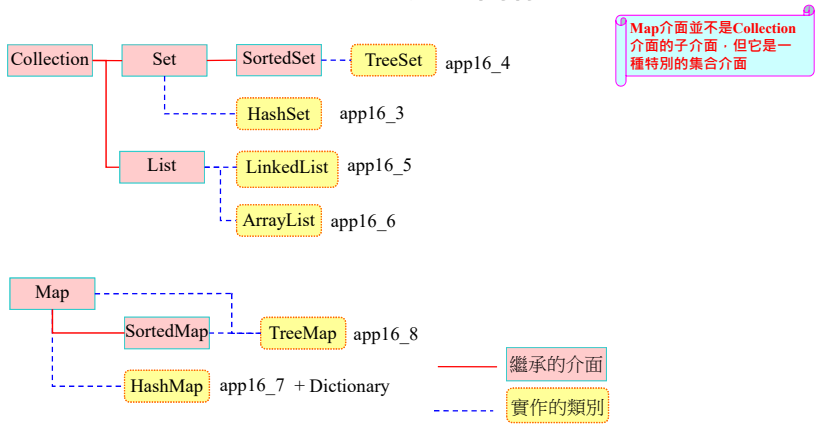
集合物件的概念

- 集合物件
  - 一群相關聯的資料，集合在一起組成一個物件
  - 集合物件裡的資料，稱為元素(elements)
- Java Collection Framework包括三個部分：
  - 介面(Interface): collection是一個介面
  - 演算法(Algorithms): collection類別提供許多排序搜尋等演算法
  - 實作(Implementations): 透過implements來實作

12

Collection介面的繼承關係圖

- 下圖是各種collection介面的**繼承關係圖**



13

不同儲存資料型態的主要差異

- TreeSet只儲存一個物件，TreeSet中不能有重複物件
- TreeMap儲存兩個物件Key和Value(僅僅key物件有序)，而TreeMap中的value可以存在重複
- HashSet以物件作為元素， HashSet不可重複物件
- HashMap以(key-value)的一組物件作為元素,HashMap可重複物件

14

16.1 認識集合物件

簡單的範例

<>泛型的寫法

Collection介面的實作類別放置在java.util類別中

集合內物件元素的型態為String

```
//app16_1, 簡單的範例
import java.util.*;
public class app16_1
{
 public static void main(String args[])
 {
   HashSet<String> hset=new HashSet<String>();

   hset.add("Monkey"); //增加元素
   hset.add("Bunny");  //增加元素
   hset.add("Monkey"); //增加元素

   System.out.println("HashSet內容:"+hset); //顯示集合物件的內容
  }
}
```

HashSet是利用**雜湊法**存取元素的一種集合，集合內的元素不能重複存在

```
/* output----------------------------------------
  HashSet內容:[Monkey, Bunny]
 ----------------------------------------*/
```

15

泛型 generic (1/3)

- 泛型型態(generic)
  - 在編譯時期即會檢查集合物件的型態
  - 小於及大於符號（<、>）所含括起來的型態，就是**泛型型態**
  - 利用一個通用的型態來代表所有可能的型態
  - 泛型型態要使用原始資料型態的wrapper class
  - 泛型是將程式碼簡潔化的一個重要技術

Wrapper class (12.6.3 p12-39)

基於效率考量，原始資料型態(primitive) 如:**boolean, byte, char, short, int, long, float, double**等均不被看成是物件。但Java還是提供一些特殊的類別，讓原始資料型態在使用上有如物件一般，這些特殊的類別，我們稱為wrapper class。

表12.6.5 原始資料型態與其wrapper class

原始資料型態	包裹類別(wrapper class)
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

- wrapper class提供的變數均屬「類別變數」
- wrapper class所提供的函數均是「類別函數」

wrapper class提供的轉換函數

表12.6.6 各種類別常用的轉換函數

類別	method	主要功能
Byte	static byte parseByte(String s)	將字串s轉換成byte型態的值
Byte	static String toString(byte b)	將byte型態的數值b轉換成字串
Character	static String toString(char c)	將字元c轉換成字串
Short	static short parseShort(String s)	將字串s轉換成短整數
Short	static String toString(short s)	將短整數s轉換成字串
Integer	static int parseInt(String s)	將字串s轉換成整數
Integer	static String toString(int i)	將整數i轉換成字串
Long	static long parseLong(String s)	將字串s轉換成長整數
Long	static String toString(Long i)	將長整數i轉換成字串
Float	static float parseFloat(String s)	將字串s轉換成浮點數
Float	static String toString(float f)	將浮點數f轉換成字串
Double	static double parseDouble(String s)	將字串s轉換成倍精度浮點數
Double	static String toString(double d)	將倍精度浮點數d轉換成字串

18

wrapper class的範例

- 下面的程式碼是Integer類別使用的範例：

```
//app12_11,Integer class method的應用
public class app12_11
{
 public static void main(String args[])
 {
   String str;
   int inum;

   inum=Integer.parseInt("654")+3; // 將字串轉成整數後，再加3

   System.out.println(inum);

   str=Integer.toString(inum)+"3"; // 將 "3" 附加在字串後面

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

```
657
6573
```

9

泛型 generic (2/3)

- 假設類別CMember內含一個資料成員id，因此需要一個setId() 函數來設定id的值。
  - **如果id的型態允許是整數或字串**，則多載的技術可能派不上用場。多載只能設定函數可接收不同型態的引數
  - **泛型技術是利用一個通用的型態來代表所有可能的型態**
- 類別CMember內的資料成員id想設計成通用的型態，可利用下面的語法來宣告：

```
class CMember<T>      //定義泛型類別CMember, T為通用型態
{ // CMember類別的內容      }
```

- 如果id的型態希望是整數，可利用下面的語法來建立：

```
CMember<Integer> obj= new CMember<Integer>();
```

必須是wrapper class裡的資料型態

泛型 Generic(3/3)

```
//app16_2, 簡單的泛型範例
public class app16_2
{
public static void main(String args[])
{
   CMember<Integer> obj1=new CMember<Integer>();
   CMember<String>  obj2=new CMember<String>();
   obj1.setId(6);
   obj2.setId("Lily");
   obj1.show();
   obj2.show();
}
}
class CMember<T> //定義泛型類別CMember，T為通用的型態
{
private T id;  //宣告id的型態為T
public void setId(T value)
 {
  id=value;        //將id成員設為傳入的引數
 }
public void show()
 {
  System.out.println("id="+id);
 }
}
```

使用泛型類別的好處
只要物件明確指出通用型態是哪一種型態，物件成員即可以在指定型態下正確的執行。

```
id=6
id=Lily
```

通用型態的說明

```
public class app16_2
{
    public static void main (String args[])
    {
        Cmember < Integer > obj1=new Cmember < Integer > ();
        ……..
    }
}
```

```
Class CMebmer <T>
{
    private T id;
    public void setId(T value)
    {
        id=value;
    }
    ……..
}
```

```
public Cmember
{
    private Integer id;
    public void setId(Integer value)
    {
        id=value;
    }
    ……
}
```

String

String

16-1課堂練習1

ex16_1_1.java

試修改app16_2，加入一個CMember<T> 建構元，可用來設定id成員的初值，並撰寫程式碼測試之。

```
/* output----------
    id=6
    id=Lily
    id=0.35
------------------*/
```

集合Set的特性(1/2)

- 依照集合的特性(不同的集合具備不同的特性)，為資料選擇適合儲存的集合物件。
  - **自動排序性**：自動加入集合的元素做遞增或遞減的排序。
  - **重複性**：集合中的元素是否允許存在相同的物件。
  - **次序性**：元素是否會依照加入集合時的順序依次排列。
  - **使用關鍵值**：利用關鍵值存放元素，一個關鍵值(key)對照一個對應值(value)，因此關鍵值的內容必須是唯一存在。

24

集合Set的特性(2/2)

- 下表列出常用的集合類別（或介面）的各種特性

表16.1.1 集合的特性與級合類別/介面的關係

集合類別/介面	排序性	不可重複性	次序性	使用關鍵值
TreeSet	♥	♥		
SortedSet	♥	♥		
HashSet		♥		
LinkedHashSet		♥	♥	
ArrayList			♥	
LinkedList			♥	
TreeMap	♥			♥
SortedMap	♥			♥
HashMap				♥
Hashtable				♥
LinkedHashMap			♥	♥

25

集合Set介面

- Set是集合之意
- 在集合中的元素沒有特定的順序，且不能重複

表16.3.1 List介面常用的method

Method	主要功能
void add(int index, E element)	在index位置加入element元素，List的索引值從0開始
boolean addAll (int index,Collection<? extends E> c)	在index位置加入Collection的所有元素，成功時傳回true
E get(int index)	從集合中取得並傳回索引值為index的元素
int indexOf(Object o)	搜尋集合中是否有與o相同的元素，傳回第一個搜尋到的索引值，找不到則傳回-1
Iterator iterator()	取得集合物件
int lastIndexOf(Object o)	搜尋集合中是否有與o相同的元素，傳回最後一個搜尋到的索引值，找不到則傳回-1
ListIterator<E> listIterator()	取得實作ListIterator<E>介面的集合物件，即listIterator(0)，第一個元素的索引值為0
ListIterator<E> listIterator(int index)	取得實作ListIterator<E>介面的集合物件，第一個元素的索引值為index
E remove(int index)	從集合物件中刪除index位置的元素
E set(int index, E element)	將集合中index位置的元素置換成element
List<E> subList(int fromIndex, int toIndex)	傳回索引值fromIndex(含) 到toIndex(不含)位置的子集合

認識HashSet類別

- HashSet類別
  - 實作Set介面的類別
  - 利用雜湊表（hash table）演算法改進執行的效率
  - 儲存元素時，元素排列的順序和原先加入時的順序可能不同
  - HashSet物件裡的元素都是唯一存在的
- 下表列出常用的HashSet建構元

表 16.2.2 java.util.HashS<E> 類別的建構元

建構元	主要功能
HashSet()	建立一個全新空的HashSet物件，預設的元素個數為16個
HashSet(Collection<? Extends E> c)	建立一個新的且包含特定的 Collection物件c之HashSet物件

使用HashSet類別 (1/2)

- 宣告元素型態為Integer的HashSet類別之物件hset：

```
HashSet<Integer> hset=new HashSet<Integer>();
```

- 下面的範例說明如何使用HashSet類別：


```
import java.util.*;
public class app16_3
{
   public static void main(String args[])
   {
      HashSet<String> hset=new HashSet<String>();

      String str1="Puppy";
      String str2="Kitty";
      System.out.println("Hash empty: "+hset.isEmpty());
      hset.add("Monkey");  //增加元素
      hset.add("Bunny");   //增加元素
      hset.add(str1);   //增加元素
      hset.add(str2);   //增加元素

      System.out.println("Hash size="+hset.size()); // 顯示元素個數
      System.out.println("Hash empty: "+hset.isEmpty());//集合內沒有東西, 則傳回true
      System.out.println("HashSet內容:"+hset); // 顯示集合物件的內容

      hset.remove(str2);
      System.out.println("清除Kitty..., Hash size="+hset.size());

      System.out.println("Hash中是否有"+str2+"? "+hset.contains(str2));
      System.out.println("Hash中是否有fish? "+hset.contains("fish"));
      System.out.println("Hash中是否有Puppy? "+hset.contains("Puppy"));
      hset.remove("Bunny");
      System.out.println("清除Bunny..., Hash size="+hset.size());

      System.out.println("HashSet內容:"+hset);
      hset.clear();
      System.out.println("清除Hash中所有的物件...");
      System.out.println("Hash empty: "+hset.isEmpty());
   }}
```

元素加入HashSet物件的順序和輸出順序不同

```
Hash empty: true
Hash size=4
Hash empty: false
HashSet內容:
[Monkey, Kitty, Puppy, Bunny]
清除Kitty..., Hash size=3
Hash中是否有Kitty? false
Hash中是否有fish? false
Hash中是否有Puppy? true
清除Bunny..., Hash size=2
HashSet內容:[Monkey, Puppy]
清除Hash中所有的物件...
Hash empty: true
```

TreeSet類別與SortedSet介面

- TreeSet實作SortedSet介面
  - 資料會由小而大排列，為排序集合物件（sorted collection）
  - 元素均不能重複出現
  - TreeSet所提供的建構元，以及實作SortedSet介面的函數如下：

表16.2.3 java.util.TreeSet<T> 類別的建構元

建構元	主要功能
TreeSet()	建立一個全新、空的TreeSet物件
TreeSet(Collection<? extends E> c)	建立一個新的、且包含特定的Collection物件c之TreeSet物件

函數	主要功能
T first()	取得集合物件中的第一個元素
SortedSet<T> headSet(T to Elm)	取得小於toElm的TreeSet物件
T last()	取得集合物件中的最後一個元素
SortedSet<T> subSet(T fromElm, T toElm)	從fromElm這個元素開始取出，取到toElm之前的元素。toElm不會被抓取到新的子集合中
SortedSet<T> tailSet(T fromElm)	取得大於等於fromElm的元素

TreeSet的範例

- app16_4是TreeSet的範例：

```
import java.util.*;
public class app16_4
{
   public static void main(String args[])
   {
    TreeSet<Integer> tset=new TreeSet<Integer>();

      for(int i=20;i>=2;i-=2) //增加元素
         tset.add(i);

      System.out.println("元素個數="+tset.size());
      System.out.println("集合內容="+tset); //顯示集合物件的內容

      System.out.println("第一個元素="+tset.first());

      System.out.println("最後一個元素="+tset.last());

      System.out.println("介於6和14之間的集合="+tset.subSet(6,14));

      System.out.println("大於等於10的集合="+tset.tailSet(10));

      System.out.println("小於8的集合="+tset.headSet(8));
   }
}
```

```
元素個數=10
集合內容=[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
第一個元素=2
最後一個元素=20
介於6和14之間的集合=[6, 8, 10, 12] 6=<x<14
大於等於10的集合=[10, 12, 14, 16, 18, 20]
小於8的集合=[2, 4, 6]
```

不管元素加入TreeSet的次序如何，其儲存的順序會自動由小到大排列

16-2課堂練習1

ex16_2_1.java

請依下面的題意依序完成程式的需求。

(a) 試著將整數**65**、**29**、**18**、**34**以**HashSet**的泛型型態儲存。

(b) 試著將整數**97**、**62**、**53**以**TreeSet**的泛型型態儲存。

(c) 將**HashSet**的元素加入到**TreeSet**集合裡，並將**TreeSet**所有的元素印出。

(d) 計算 **(c)** 完成的 **TreeSet** 集合裡的第一個元素與最後一個元素之平均值。

Note:將HashSet的元素加入到TreeSet集合語法：tset.addAll(hset)

```
/* output----------------------------------------------------------------
 HashSet的內容:[34, 65, 18, 29]
 TreeSet的內容:[53, 62, 97]
 加入新元素後，TreeSet的內容:[18, 29, 34, 53, 62, 65, 97]
 18 與 97 的平均值為 57.5
--------------------------------------------------------------------------*/
```


16-2課堂練習2

ex16_2_2.java

請依下面的題意依序完成程式的需求。

(a) 試建立**TreeSet**型態的物件**tset**，內含字串型態的元素，其內容如下所示:

Everything has an end.
Good to begin well, better to end well.
You cannot tell a book by its cover.
A good book is a light to the soul.

(b) 將**tset**中的第一個元素印出。

(c) 將**tset**中的最後一個元素印出。

```
/* output-----------------------------------------
第一個元素: A good book is a light to the soul.
最後一個元素:You cannot tell a book by its cover.
-------------------------------------------------*/
```

List介面

- List介面
  - 屬於有序集合物件（ordered collection）
  - 元素可以重複
  - 元素具有索引值（index）
- List介面和SortedSet（TreeSet）介面的比較
  - List會依照索引值來排列元素的位置
  - SortedSet會根據元素本身的大小來排列

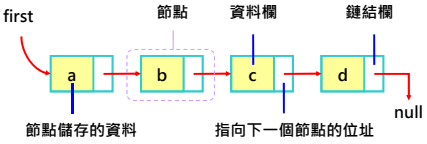
List介面常用的函數

表16.3.1 List介面常用的method

函數	主要功能
void add(int index, T element)	在index位置加入element元素，List的索引值從0開始
Boolean addAll(int index, Collection<? extends T> c)	在index位置加入Collection的所有元素，成功時傳回true
T get(int index)	從集合中取得並傳回索引值為index的元素
int indexOf(Object o)	搜尋集合中是否有與o相同的元素，傳回第一個搜尋到的索引值，找不到則傳回-1
Iterator iterator()	取得集合物件
int lastIndexOf(Object o)	搜尋集合中是否有與o相同的元素，傳回最後一個搜尋到的索引值，找不到則傳回-1
ListIterator<T> listIterator()	取得實作ListIterator<T>介面的集合物件，即listIterator(0)，第一個元素的索引值為0
ListIterator<T> listIterator(int index)	取得實作ListIterator<T>介面的集合物件，第一個元素的索引值為index
T remove(int index)	從集合物件中刪除index位置的元素
T set(int index, E element)	將集合中index位置的元素置換成element
List<T> subList(int fromIndex, int toIndex)	傳回索引值fromIndex(含) 到toIndex(不含)位置的子集合

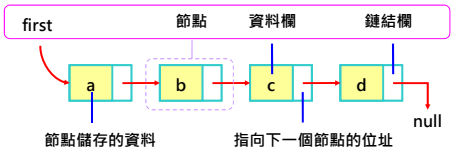
LinkedList類別

- 鏈結串列（linked list）
  - 節點（node）分為2個欄位
    - 資料欄
      - 資料欄儲存的是所需之**資料**
    - 鏈結欄
      - 鏈結欄記錄著下一個**節點的位址**
  - 鏈結串列的最後一個節點會指向**null**
    - 表示後面已沒有節點

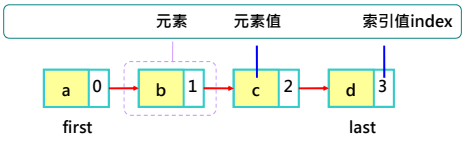


鏈結串列

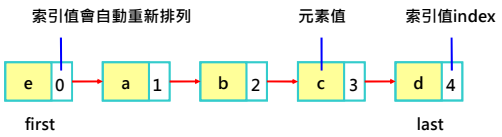
- 下圖為鏈結串列示意圖：



- 在Java中，LinkedList物件的表示方式：



- 在List中增加或刪除元素時，索引值會自動重新排序
- 下圖是將e加到LinkedList物件起始處之後的情形：



LinkedList的建構元與函數

- 下表列出LinkedList建構元與常用的函數

表16.3.2 java.util.LinkedList<T> 類別的建構元與函數

建構元	主要功能
LinkedList()	建立一個空的LinkedList物件
LinkedList(Collection<? extends T> c)	建立一個包含特定的Collection物件c之LinkedList物件

函數	主要功能
void addFirst(T o)	將元素o加入LinkedList物件的起始處
void addLast (T o)	將元素o加入LinkedList物件的結尾處
T getFirst()	取得LinkedList物件中的第一個元素
T getLast()	取得LinkedList物件中的最後一個元素
T removeFirst()	刪除並傳回LinkedList物件中的第一個元素
T removeLast()	刪除並傳回LinkedList物件中的最後一個元素

LinkedList的範例

```
import java.util.*;
public class app16_5
{
   public static void main(String args[])
   {
      LinkedList<Integer> llist=new LinkedList<Integer>();

      for(int i=10;i<=30;i+=10)  // 增加元素
         llist.add(i);
      llist.addFirst(100);
      llist.addLast(200);
      llist.addFirst(300);

      System.out.println("元素個數="+llist.size());
      System.out.print("LinkedList的元素:");
      for(int i=0;i<llist.size();i++)  // 顯示集合物件的內容
         System.out.print(llist.get(i)+" ");

      System.out.print("\n刪除最後一個元素 ");
      System.out.println(llist.removeLast()+"...");

      System.out.println("第一個元素="+llist.getFirst());
      System.out.println("最後一個元素="+llist.getLast());
      System.out.println("元素值為200的索引值="+llist.indexOf(200));
   }
}
```

❤TreeSet是由小到大排資料，
❤LinkedList是由加入先後順序排資料

加入元素

```
元素個數=6
LinkedList的元素:300 100 10 20 30 200
刪除最後一個元素 200...
第一個元素=300
最後一個元素=30
元素值為200的索引值=-1
```

印出-1, 表示集合中沒有這個元素值

列出元素且刪除

16-3課堂練習1

ex16_3_1.java

請依下面的題意依序完成程式的需求。

(a) 請取出**10**個介於**0~100**的整數亂數，將它們加入一個**LinkedList**型態的物件 **llist** 中，然後印出 **llist** 物件中所有的元素。

(b) 印出**llist**物件中的第一個及最後一個元素。

(c) 計算並印出第一個及最後一個元素的乘積。

Note: 0~100的整數亂數 ((int)(Math.random()*100)

```
/* output----------------------------------------------
 LinkedList 的內容:[51, 83, 34, 42, 40, 99, 18, 53, 32, 86]
第一個元素=51
最後一個元素=86
51*86=4386
-------------------------------------------------------*/
```

40

16.3 實作List介面

ArrayList類別

- 元素加入ArrayList物件時，是用索引值（index）儲存
- ArrayList類別的建構元：

表16.3.3　java.util.ArrayList<E> 類別的建構元

建構元	主要功能
ArrayList()	建立一個空的ArrayList物件
ArrayList(Collection<? extends E> c)	建立一個包含特定的Collection物件c之ArrayList物件

函數	主要功能
void trimToSize()	將ArrayList物件的容量剪裁成目前元素的數量

Array：不特定型別，固定長度的陣列，長度需事先宣告。
ArrayList：不特定型別，不固定長度的陣列。

41

ArrayList類別的範例

```
import java.util.*;
public class app16_6
{
   public static void main(String args[])
   {
      ArrayList<Integer> alist=new ArrayList<Integer>();

      for(int i=10;i<=50;i+=10) // 增加元素
         alist.add(i);
      alist.add(3,200);
      alist.add(0,300);
      alist.add(400);// 將400放在alist的最後一個位置

      System.out.println("元素個數="+alist.size());
      System.out.println("ArrayList的元素:"+alist);
      System.out.println("將索引值1的元素以200取代...");
      alist.set(1,200);
      System.out.println("ArrayList的元素:"+alist);
      System.out.print("第一個元素值為200的索引值=");
      System.out.println(alist.indexOf(200));
      System.out.print("最後一個元素值為200的索引值=");
      System.out.println(alist.lastIndexOf(200));
   }
}
```

```
元素個數=8
ArrayList的元素:[300, 10, 20, 30, 200, 40, 50, 400]
將索引值1的元素以200取代...
ArrayList的元素:[300, 200, 20, 30, 200, 40, 50, 400]
第一個元素值為200的索引值=1
最後一個元素值為200的索引值=4
```

比陣列還有彈性之處在於可以add()無限增加，若要查詢總數，用alist.size()即可知道共有幾個元素

Array 與 ArrayList 的分別

- **建置方式不同**
  - **Array**
    是一個有固定大小的Array。
    建立新的Array時，需設定一個大小。
    建立後不能再更改大小。
    int arr[] = new int[10]
  - **ArrayList**
    是一個浮動大小的Array。
    建立新的Array時，不需要為它設定大小。
    建立後可以隨意更改它的大小。
    以List interface的實作。
    ArrayList arrL = new ArrayList();
- **新增元素及存取元素的方式不同**
  - Array透過[]的方式新增元素而ArrayList就透過add()。
  - Array透過[]的方式存取元素而ArrayList就透過get()。
- **資料結構(Data type)的類別不同**
  - Array 可以包含primitive data types 和object entities。
    ArrayList 只可以包含object entries 但不支持primitive data types。

Array 與 ArrayList 的範例(比較差異)

```
import java.util.*;
class ArrayTest
{
    public static void main(String args[])
    {
        int[] arr = new int[2];
        arr[0] = 1; // 建立元素
        System.out.println(arr[0]); // 存取元素
       ArrayList<Integer> arrL = new ArrayList<Integer>(2);
        arrL.add(1);
        arrL.add(2);
       System.out.println(arrL.get(0));
    }
}
```

```
1
1
```

```
import java.util.*;
class ArrayTest
{
    public static void main(String args[])
    {
       // 允許primitive data types
        int[] array = new int[3];

        // 允許 object entities
        Test[] array1 = new Test[3];

        // 不允許primitive data types
        // 當運行以下code時，會出現error
        ArrayList<char> arrL = new ArrayList<char>();

        // 允許 object entities
        ArrayList<Integer> arrL1 = new ArrayList<>();
        ArrayList<String> arrL2 = new ArrayList<>();
        ArrayList<Object> arrL3 = new ArrayList<>();
    } }
```

44

16-3課堂練習2

ex16_3_2.java

請將下列字串以 **ArrayList** 物件 **alist** 儲存，並利用迴圈印出 **alist** 中所有的元素內容。

Think before you act.
Full of courtesy, full of craft.
Best is cheapest.
Look before you leap.
Take time when time comes.

```
/* output----------------------------------
  ArrayList 的內容:
 Think before you act.
 Full of courtesy, full of craft.
 Best is cheapest.
 Look before you leap.
 Take time when time comes.
----------------------------------------*/
```

45

16-3回家作業1

hw16_3_1.java

請依下面的題意依序完成程式的需求。

(a) 將字串 **"apple"** 與 **"guava"**加入**LinkedList**型態的物件**llist**中，然後印出**llist** 物件中所有的元素。

(b) 將字串 **"tomato"**、**"apple"**、**"papaya"** 與 **"grape"**，加入 **ArrayList** 型態的物件 **alist** 中，並印出 **alist** 裡的所有元素。

(c) 將**llist**裡的元素加入**alist**中，並將**alist**所有元素印出。

(d) 請印出**(c)**所建立的**alist**中，第一個及最後一個出現**apple**的索引值。

```
/* output----------------------------------------------------
   LinkedList的內容:[apple, guava]
   ArrayList的內容:[tomato, apple, papaya, grape]
   加入新元素後，ArrayList 的內容:[tomato, apple, papaya, grape, apple, guava]
   第一個 apple 的索引值=1
   最後一個 apple 的索引值=4
 ----------------------------------------------------------------*/
```


Map介面

16.4 實作Map介面

- Map介面
  - 以關鍵值（key）儲存
  - 關鍵值對應到的資料，即對應值（value）

表16.4.1 Map<K,V>介面常用的method

函數	主要功能
void clear()	從集合中移除所有的元素
boolean containsKey(Object key)	當集合物件裡包含關鍵值key，即傳回true
boolean containsValue(Object value)	當集合物件裡包含對應值value，即傳回true
V get(Object key)	傳回集合物件中關鍵值key的對應值
boolean isEmpty()	集合物件若沒有任何元素，傳回true
V put(K key, V value)	將關鍵值key新增至集合物件中，若key值相同，則將對應值value取代舊有的資料
VputAll(Map<?ExtendsK,?extends V >t)	將整個Map物件t複製到集合中
Set<K> keySet()	將關鍵值轉換成實作Set介面的物件
V remove(Object key)	從集合物件中刪除關鍵值key的元素，成功時傳回被刪除的value值，否則傳回null
int size()	傳回集合物件的元素個數
Collection<V> values()	將對應值轉換成實作Collection介面的物件

HashMap類別

- HashMap類別儲存的元素分為
  - 關鍵值key
  - 對應值value
- 宣告HashMap類別物件時，關鍵值與對應值以逗號分開：
  - `HashMap<Integer,String> hmap=new HashMap<Integer,String>();`
- 下表列出HashMap類別的建構元

表16.4.2 java.util.HashMap<K,V> 類別的建構元

建構元	主要功能
HashMap()	建立一個空的HashMap物件，預設的元素個數為16個
HashMap(int initialCapacity)	建立一個空的HashMap物件，指定的元素個數為initialCapacity個
HashMap(Map<? extends K,? extends V> m)	建立一個包含特定的Map物件m之HashMap物件

HashMap類別的範例

```
元素個數=3
HashMap的元素:{94001=Fiona, 94003=Ariel, 94002=Ryan}
HashMap中是否有關鍵值94002? true
HashMap中是否有對應值Kevin? false
清除關鍵值94001的資料..., 元素個數=2
HashMap的元素:{94003=Ariel, 94002=Ryan}
關鍵值94003的對應值=Ariel
```

```
//app16_7,將物件加入HashMap之範例
import java.util.*;
public class app16_7
{
public static void main(String args[])
{
   HashMap<Integer,String> hmap=new HashMap<Integer,String>();

   hmap.put(94001,"Fiona");
   hmap.put(94003,"Ariel");
   hmap.put(94002,"Ryan");

   System.out.println("元素個數="+hmap.size());
   System.out.println("HashMap的元素:"+hmap);
   System.out.print("HashMap中是否有關鍵值94002? ");
   System.out.println(hmap.containsKey(94002));
   System.out.print("HashMap中是否有對應值Kevin? ");
   System.out.println(hmap.containsValue("Kevin"));
   hmap.remove(94001);
   System.out.print("清除關鍵值94001的資料..., ");
   System.out.println("元素個數="+hmap.size());
   System.out.println("HashMap的元素:"+hmap);
   System.out.println("關鍵值94003的對應值="+hmap.get(94003));
  }
}
```

16-4課堂練習1

ex16_4_1.java

下表是某班學生的英文成績表，請依下面的題意依序完成程式的需求。

姓名　英文成績

Ryan　95

Fiona　83

Jack　89

Kevin　76

Ariel　92

```
/* output----------------------------------------------------------------------
HashMap的內容:{Ryan=95, Kevin=76, Fiona=83, Jack=89, Ariel=92}
Ariel 與 Fiona 的英文成績平均值=87.5
Kevin 與 Jack 的英文成績相差 13 分
----------------------------------------------------------------------------*/
```

(a) 請將姓名當成關鍵值(字串型態)，英文成績做為對應值(整數)，儲存成 **HashMap** 的泛型型態後，印出 **HashMap** 集合裡的所有元素。

(b) 請計算**Ariel**與**Fiona**的英文成績平均值。

(c) 請計算**Kevin**與**Jack**的英文成績相差多少。

50

HashMap(範例說明)

Hash Map 雜湊，就是一對鍵值(key-value)，一個鍵搭配著一個值的對應方式。例如：身份證字號可以對應人名，身份證字號就是雜湊的鍵(key)，而利用這個鍵所得到的值(value)就是姓名。而且鍵是這個雜湊中唯一的值，也就是一個雜湊中，不能有重複的鍵，以確保可以找到唯一的人。

- HashMap是一個可以直接由key對應value的動態資料結構，我們要做的事情是
- 1.從檔案讀取字典資料。
- 2.把資料放進HashMap中。
- 3.提供字典查詢的功能。

- 建立HashMap物件 HashMap<KeyDataType, ValueDataType>
- 從檔案讀取資料 → data.txt
- 提供字典查詢的功能，輸入0結束程式。

```
apple,蘋果
house,房子
bear,熊
cat,貓
bunny,兔子
hamburger,漢堡
```

51

```java
package exx;
import java.io.*;
import java.util.HashMap;
import java.util.Scanner;
public class Dictionary {
public static void main(String args[]) throws Exception{
Scanner sc=new Scanner(System.in);

HashMap<String, String> dic=new HashMap<String, String>();//key對應value
FileReader in=new FileReader("c:\\Java\\data.txt");
BufferedReader reader=new BufferedReader(in);
String line,english,chinese;
int cutIndex;
while((line=reader.readLine())!=null)
 {
      //System.out.println(line);
      cutIndex=line.indexOf(",");
      english=line.substring(0, cutIndex);
      chinese=line.substring(cutIndex+1, line.length());
      dic.put(english, chinese);
  }
```

```
apple,蘋果
house,房子
bear,熊
cat,貓
bunny,兔子
hamburger,漢堡
```

```java
while(true) {
  System.out.print("請輸入(0結束)：");
  String keyword=sc.nextLine();
   if(keyword.equals("0"))
     {
      System.out.println("程式結束");
      break;
     }
   else
     {//keyword若對應到HashMap的key,則傳回value
        String result=dic.get(keyword);
         if(result==null)
            {
              System.out.println("查不到資料");
            }
         else
            {
              System.out.println(result);
            }
     }
  }
 }
}
```

```
請輸入(0結束)：蘋果
apple
請輸入(0結束)：橘子
查不到資料
請輸入(0結束)：0
程式結束
```

16-4課堂練習2

- ex16_4_2.java
- 請自己建立一個可以允許建立中英文資料的字典檔案，建完之後允許查詢中英文生字。

```
請輸入建檔資料(如：水,water)(0結束建檔)：apple,蘋果
請輸入建檔資料(如：水,water)(0結束建檔)：orange,橘子
請輸入建檔資料(如：水,water)(0結束建檔)：kiwi,奇異果
請輸入建檔資料(如：水,water)(0結束建檔)：0
建立字典結束
請輸入查詢資料(0結束)：apple
蘋果
請輸入查詢資料(0結束)：橘子
orange
請輸入查詢資料(0結束)：0
程式結束
```

53

TreeMap類別

- TreeMap類別是實作SortedMap介面的類別
- 元素依關鍵值由小至大排序
- 下表列出TreeMap類別的建構元及函數

表16.4.3 java.util.TreeMap<K,V>類別的建構元與method

建構元	主要功能
TreeMap()	建立一個空的TreeMap物件，依關鍵值由小至大排序
TreeMap(Map<? extends K,? extends V> m)	建立一個包含特定的Map物件m之TreeMap物件
TreeMap(SortedMap<K,? extends V> m)	建立一個包含特定的實作SortedMap介面物件m之TreeMap物件

函數	主要功能
K firstKey()	傳回集合中第一個關鍵值，即最小關鍵值
K lastKey()	傳回集合中最後一個關鍵值，即最大關鍵值
SortedMap<E> subMap(K fromKey, K toKey)	取得大於等於fromKey，且小於toKey的TreeMap物件
SortedMap<E> tailMap(K fromKey)	取得大於等於fromKey的TreeMap物件

TreeMap類別的範例


```
//app16_8, TreeMap範例
import java.util.*;
public class app16_8
{
public static void main(String args[])
{
   int k1=94001,k2=94003,key;
   TreeMap<Integer,String> tmap=new TreeMap<Integer,String>();

   tmap.put(94001,"Fiona");
   tmap.put(94003,"Ariel");
   tmap.put(94002,"Ryan");
   tmap.put(94004,"Jack");

   System.out.println("元素個數="+tmap.size());
   System.out.println("TreeMap的元素:"+tmap);
   key=tmap.firstKey();
   System.out.println("第0個元素= "+key+", "+tmap.get(key));
   key=tmap.lastKey();
   System.out.println("最後一個元素= "+key+", "+tmap.get(key));
   System.out.print("介於"+k1+"和"+k2+"之間的TreeMap=");
   System.out.println(tmap.subMap(k1,k2));
   System.out.print("大於等於"+k2+"的TreeMap=");
   System.out.println(tmap.tailMap(k2));
}
}
```

app16_8是將物件加入TreeMap之範例

```
元素個數=4
TreeMap的元素:{94001=Fiona, 94002=Ryan, 94003=Ariel, 94004=Jack}
第0個元素= 94001, Fiona
最後一個元素= 94004, Jack
介於94001和94003之間的TreeMap={94001=Fiona, 94002=Ryan}
大於等於94003的TreeMap={94003=Ariel, 94004=Jack}
```

>=94001 and <94003

16-4 課堂作業1

ex16_4_1.java

請依下面的題意依序完成程式的需求。

(a)請取出**5**個小於**100**的整數亂數，以**HashMap**型態的物件**hmap**儲存，關鍵 值為 **0~4**。

(b)請利用**values()**函數，將**(a)**中**hmap**的對應值，轉換成**TreeSet**物件**tset**。

(c)請利用**keySet()**函數，將**(a)**中**hmap**的關鍵值，轉換成**HashSet**物件**hset**。

(d)印出**hmap**、**tset**及**hset**的所有元素。

```
/* output-------------------------------------
 HashMap的內容: {0=0, 1=22, 2=30, 3=90, 4=78}
 TreeSet的內容: [0, 22, 30, 78, 90]
HashSet的內容: [0, 1, 2, 3, 4]
 ---------------------------------------------*/
```

56

16-4 回家作業1

hw16_4_1.java

請依下面的題意依序完成程式的需求。

(a)試著將整數**36**、**15**以**HashSet**的泛型型態儲存。

(b)試著將整數**52**、**23**、**32**、**69**、**10**、**7**、**36**、**15**以**TreeSet**的泛型型態儲存。

(c)將**TreeSet**集合中所有的元素印出。

(d)若是**TreeSet**集合中包含有**32**的元素，則將該元素刪除，刪除後請將集合的內容 重新印出。若是找不到值為 **32** 的元素，則顯示字串 **"TreeSet** 中沒有 **32** 的元素**"**。

(e)請確認**TreeSet**集合物件中是否包含**(a)**所建立的**HashSet**之所有元素。

```
/* output-----------------------------------------------------
 TreeSet內容:[7, 10, 15, 23, 32, 36, 52, 69]
 TreeSet 的元素 32 已被刪除...
 刪除後的TreeSet內容:[7, 10, 15, 23, 36, 52, 69]
 TreeSet中是否包含[15, 36]的元素? true
-----------------------------------------------------------------*/
```

57

使用for-each迴圈(1/2)

- for-each迴圈的格式如下：

預設建構元的格式

```
for(元素型態 迴圈控制變數 : 集合或陣列名稱)
// 迴圈主體;
```

- 迴圈控制變數的型態要與集合或陣列裡的元素型態相同

```
LinkedList <Double> llst=nw LinkedList<Double>(); //元素的型態為Double
 ....                                  型態要一致
for(double data:llst)  //迴圈控制變數的型態為double
 .....
```

資料型態 變數名稱: 陣列名稱

58

使用for-each迴圈(2/2)

- 利用for及for-each迴圈走訪元素的範例

表16.5.1　陣列與集合使用 for 迴圈及 for-each 迴圈的例

傳統 for 迴圈	使用 for-each loop

int arr[ ] = {5, 3, 8 };
int sum=0;
for(int i=0;i<arr.length;i++)
　sum+=arr[i];
	int arr[ ] = {5, 3, 8 };
	int sum=0;
	for(int i：arr)
	　sum += i;

- 使用for-each迴圈時的注意事項
  - 只能從頭開始訪問每個元素，不能從集合或陣列的尾端像前走訪。
  - 只能取出集合或陣列裡的元素而不能置換它。
  - for-each迴圈裡面的變數是區域變數。
  - 只適用於Java 5.0以後的版本。

59

for-each的範例

- 利用for-each迴圈走訪TreeSet物件裡的元素

```
//app16_9, 以for-each loop走訪TreeSet元素
import java.util.*;
public class app16_9
{
public static void main(String args[])
{
   TreeSet<String> tset=new TreeSet<String>();
   tset.add("Monkey");  // 增加元素
   tset.add("Bunny");
   tset.add("Puppy");
   tset.add("Kitty");
   System.out.print("TreeSet內容:");

   for(String i:tset) // 走訪
      System.out.print(i+" ");
  }
}
```

```
TreeSet內容:Bunny Kitty Monkey Puppy
```


16-5 課堂練習

ex16_5_1.java

請取出**10**個小於**100**的整數亂數，以**TreeSet**的泛型型態儲存，並利用**for-each**迴圈走訪集合，將集合元素印出，並計算所有元素的平均值。

0~100的整數亂數 ((int)(Math.random()*100)

/* output----------

TreeSet集合內容:6 16 20 42 54 71 80 82 83 91

平均值=54.5

----------*/