進階資料結構 List, Set, Map

簡介

資料結構主要是在研究如何把「資料」(Data)存放到電腦中(記憶體中或是檔案中)、如何處理與操作這些資料的一門科學。

在資料分析 大數據分析 資料科學 機器學習 資料探勘領域，因為需要處理大量資料，資料的存放非常重要，適當的資料結構讓資料的分析更方便。

如果日後有機會接觸到 Python, R, Spark, Hadoop 等資料科學領域，你會成天與資料為伍，會有機會用到 List, Set, Map。

以前學過的陣列(Array)是一種基本的資料結構，但是它不夠好用，因為它不能動態改變存放空間的大小。

Java 提供進階的資料結構，可以動態改變大小，並提供多種不同方式操作資料的方法，包括：

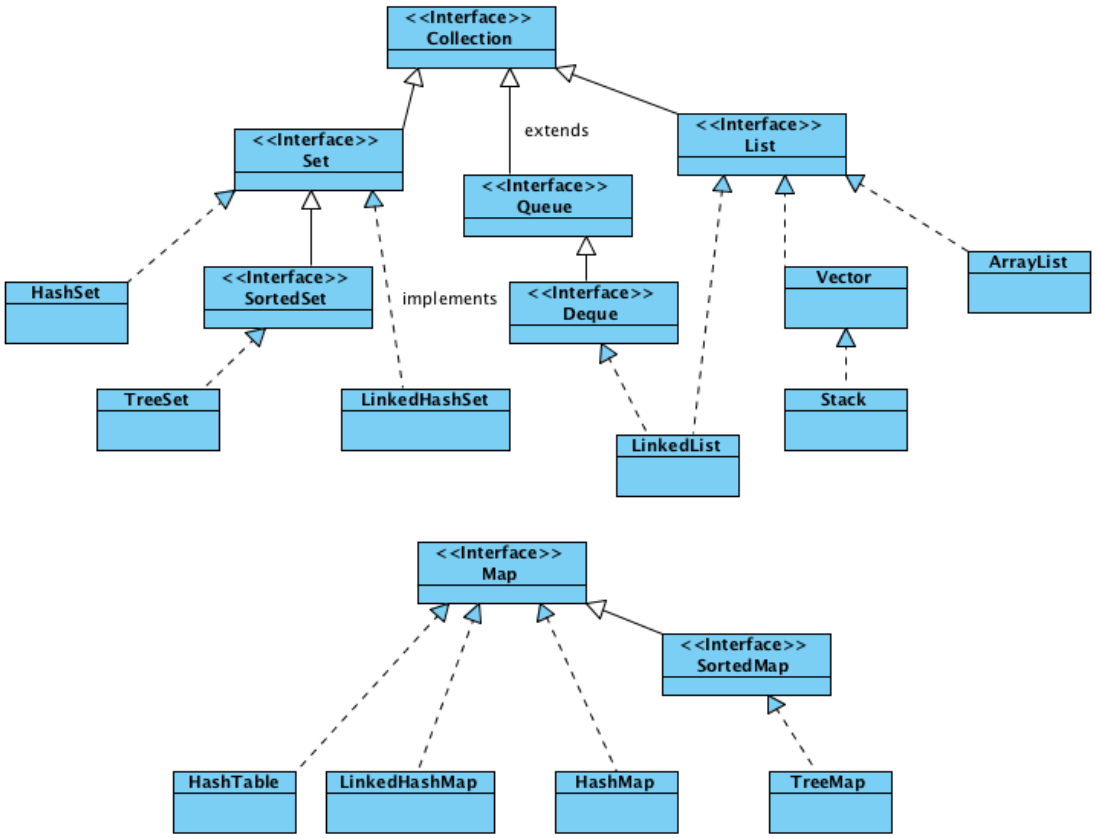
- 串列(List)、
- 堆疊(Stack)、
- 佇列(Queue)、
- 集合(Set)、
- 映射(Map)

除了 Map 外，其餘都是衍生自 Collection 介面。各介面也都衍生許多類別，這些類別可以讓程式設計師因應不同情況去使用。這裡只會介紹其中的幾種類別與其應用實例。

List 介面 (串列、串列) 存放資料可重複
-ArrayList 陣列串列 類別
-LinkedList 鏈結串列 類別

Set 介面 (集合) 存放不重複資料
-HashSet 雜湊集合類別 沒有排序
-TreeSet 樹集合類別 有排序 繼承自 SortedSet 類別

Map 介面 (映射) 存放(索引值 數值) 像字典一樣 可以透過索引查出其數值
-HashMap 類別 沒有排序
-TreeMap 類別 有排序 繼承自 SortedMap 類別



http://ordinarygeek.me/2009/12/02/an-overview-and-comparison-of-java-util-collections/

Collections 類別提供很多方便使用的靜態方法去操作資料，如 sort, search, reverse() 等方法。就如同先前學過的 Arrays 類別一樣，可以對陣列進行排序、搜尋等操作。

ArrayList 類別

宣告一個 ArrayList 方式：

ArrayList list = new ArrayList();

可以放入任何型態的資料(以物件方式存放)。

如果你只要存放字串，宣告方式如下：(Java 泛型 generic 的宣告寫法)

ArrayList<String> list = new ArrayList();

或是這樣宣告也可以，不過，似乎沒有必要

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

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


宣告時也可以給予一個參數(有建構子的版本)
String[] colors = {"red", "green", "blue","white"};
ArrayList list = new ArrayList(       Arrays.asList(colors)       );


//排序
Collections.sort(list);
Collections.sort(list, Collections.reverseOrder());
System.out.println(Collections.max(list));
```

```
package list;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class ArrayListDemo {

	public static void main(String[] args) {

		ArrayList list = new ArrayList();
		list.add(5);
		list.add("A");
		list.add("red");//在最後面位置 加入元素
		list.add("black");
		print(list);
		list.add(1, "green"); //在索引位置 加入元素

		print(list);
		list.remove("red"); //移除元素
		print(list);
```

```
        list.remove(2);//在索引位置 移除元素
        print(list);
        list.remove(new Integer(5));// 移除整數物件元素 remove(5)是指索引位置
        print(list);

        String[] colors = {"red", "green", "blue", "white"};
        ArrayList<String> list2 = new ArrayList(Arrays.asList(colors));
        print(list2);
        print(list2);
        Collections.sort(list2); //排序
        print(list2);



        //置放特定的物件
        //只能放字串物件
        ArrayList<String> list4 = new ArrayList();
        list4.add("5");
        list4.add("A");

        //只能放整數物件
        ArrayList<Integer> list5 = new ArrayList();
        list5.add(5);
        list5.add(50);
        list5.add(8);
        print(list5);
        Collections.sort(list5);
        print(list5);

    }

    public static void print(List list) {
        for (int i = 0; i < list.size(); i++) {
            System.out.print(list.get(i) + " ");
        }
        System.out.println();
    }
```

```
}
```

練習:

```
ArrayList<String> list = new ArrayList();
list.add("A");
list.add("B");
list.add("C");
list.add("D");
list.add("E");
list.add("F");
list.add("G");
list.add("H");

欲刪除以下位置的元素，程式該如何寫?
int erase[] = {1,3,5};
```

LinkedList 類別

*LinkedList 提供較多的增刪方法

ArrayList, LinkedList, 兩者操作方法不同。

```
1   package list;
2
3   import java.util.ArrayList;
4   import java.util.Arrays;
5   import java.util.LinkedList;
6   import java.util.List;
7
8   public class LinkedListTest {
9
10      public static void main(String[] args) {
11          String[] colors = {"red", "green", "blue", "white"};
```

```
12              LinkedList list = new LinkedList(Arrays.asList(colors));
13              //List list2 = Arrays.asList(colors);
14              list.addFirst("cyan");
15              list.addLast("purple");
16
17              for (int i = 0; i < list.size(); i++) {
18                    System.out.println(list.get(i));
19              }
20        }
21  }
```

*也可以用 List 介面當作資料型態

List list = new ArrayList();

List <String> list = new ArrayList();

為甚麼要用 List 介面當資料型態?

ArrayList 實作了 List 介面，因此可以用 List 介面當資料型態。

這有點抽象，想要搞清楚這個觀念，你必須要把以前教過的 OO 觀念，繼承、介面、多型等這些章節的內容再拿出來寫一次。

以前學過用父類別當資料型態，目的是想要讓程式處理資料比較方便一些，例如，讓方法的撰寫更方便、處理資料更方便。例如，Pet 的子類別有 Dog, Cat，讓一個陣列可以存放各種子類別的物件，並用一個迴圈操作這些物件(例如:貓叫、狗叫)：

Pet[] pets = new Pet[3];

Pets[0] = new Dog(“黑狗”);

Pets[1] = new Dog(“白狗”);

Pets[2] = new Cat(“花貓”);

這裡用介面當資料型態，目的與用途也是類似的。(可以稱為“父”介面，與父類別不同，你可以暫時把他們想成是類似的東西!)

Iterator 用法

```
1   package list;
```

```
2
3   import java.util.ArrayList;
4   import java.util.Collections;
5   import java.util.Iterator;
6
7   public class IteratorDemo {
8
9       public static void main(String[] args) {
10          ArrayList al = new ArrayList();
11          al.add(20);
12          al.add(10);
13
14          //方法 1
15          for (int i = 0; i < al.size(); i++) {
16              System.out.println(al.get(i));
17          }
18
19          //方法 2
20          Iterator it = al.iterator();
21
22          while (it.hasNext()) {
23              int s = (int) it.next();
24              System.out.println(s);
25          }
26
27          //方法 3
28          for (Iterator it2 = al.iterator(); it2.hasNext();) {
29              int s = (int) it2.next();
30              System.out.println(s);
31          }
32
        }
    }
```

*小畫家畫的點可以用 ArrayList 存放，就可以無限制畫很多點。

```
1   package paint;
2
3   import java.awt.BorderLayout;
4   import java.awt.Color;
5   import java.awt.Dimension;
6   import java.awt.Graphics;
7   import java.awt.Point;
8   import java.awt.event.MouseEvent;
9   import java.awt.event.MouseMotionListener;
10  import java.util.ArrayList;
11  import java.util.Iterator;
12  import javax.swing.JFrame;
13  import javax.swing.JPanel;
14
15  public class DrawPanel_01 extends JPanel implements MouseMotionListener{
16      private ArrayList<Point> points = new ArrayList();
17     private Color penColor = Color.RED;
18      private int penSize = 4;
19      public DrawPanel_01()
20      {
21          this.setPreferredSize(new Dimension(500,600));
22          this.addMouseMotionListener(this);
23      }
24
25      @Override
26      public void paintComponent(Graphics g)
27      {
28          super.paintComponent(g);
29          g.setColor(penColor);
30
31
32          for (int i=0; i< points.size(); i++)
            {
                g.fillOval(points.get(i).x,    points.get(i).y, penSize, penSize);
            }

            /*寫法二
```

```
            Point p;
            Iterator<Point> it = points.iterator();
            while (it.hasNext())
            {
                p = it.next();
                g.fillOval(p.x, p.y, penSize, penSize);
            }*/
        }
        @Override
        public void mouseDragged(MouseEvent e) {
            points.add(    e.getPoint() );
            repaint();
        }

        @Override
        public void mouseMoved(MouseEvent e) {
        }

        public static void main(String[] args) {
            DrawPanel_01 panel = new DrawPanel_01();
            JFrame app = new JFrame("小畫家");
            app.add(panel, BorderLayout.CENTER);
            app.setSize(400, 500);
            app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            app.setVisible(true);
        }

}
```

HashMap、TreeMap 類別

使用 TreeMap，計算字頻率非常方便。

宣告方式任意一種皆可：

HashMap<String, Integer> freq = new HashMap();

TreeMap<String, Integer> freq = new TreeMap();

Map<String, Integer> freq = new HashMap();

Map<String, Integer> freq = new TreeMap();
(Map 是介面，介面也可以當作資料型別去使用)
//這樣宣告也可以，不過右邊的<>有點多餘
Map<String, Integer> freq = new TreeMap<>();

放入一組資料:
freq.put(“java”,1);

取出資料:
依據“java”當作 key 索引，取出對應之值
freq.getKey(“java”)

寫法一：

```
1  package wordfreqmap;
2  import java.util.HashMap;
3  import java.util.Map;
4  import java.util.Set;
5  import java.util.TreeMap;
6  import java.util.TreeSet;
7  public class WordFreqMap_ver0 {
8      public static void main(String[] args) {
9
10         String str = "I love java you love java everyone love java";
11         String input[] = str.split(" ");
12
13         TreeMap<String, Integer> freq = new TreeMap();
14         //HashMap 比較快，TreeMap 有排序
15         //HashMap<String, Integer> freq = new HashMap();
16         for (int i = 0; i < input.length; i++) {
17             String word = input[i].toLowerCase();
18             if (freq.containsKey(word)) {
19                 int count = freq.get(word);
20                 count++;
21                 freq.put(word, count);
22             } else {
23                 freq.put(word, 1);
```

```
24                  }
25              }
26              //列印結果
27              for (String key : freq.keySet()) {
28                  System.out.printf("%s %d\n", key, freq.get(key));
29              }
30      }
31  }
32
```

如果使用 HashMap，雖沒有排序，也可以以下方式將結果排序後列印出來

```
Set keys = freq.keySet(); //把所有字拿出來
TreeSet<String> sortedKeys = new TreeSet(keys); //TreeSet
for (String key: sortedKeys)
System.out.printf("%s %d\n", key, freq.get(key));
```

寫法二：

```
1   package wordfreqmap;
2
3   import java.util.Map;
4   import java.util.TreeMap;
5
6   public class WordFreqTreeMap {
7       public static void main(String[] args) {
8
9           String str = "I love java you love java everyone love java";
10          String input[]    = str.split(" ");
11          Map<String, Integer> freq = new TreeMap();
12          for (int i=0; i < input.length; i++)
13              {
14                  String word = input[i].toLowerCase();
15                  System.out.println(word);
16                  Integer count = freq.get(word);
17                  if (count == null) //用 freq.containsKey(word)比較容易理解
18                      count = 1;
```

```
19                          else
20                              count++;
21                          freq.put(word, count);
22                      }
23                      for (String key: freq.keySet())
24                          System.out.printf("%s %d\n", key, freq.get(key));
25              }
26  }
```

HashSet、TreeSet 類別

計算交集聯集差集

```
1   package set_demo;
2
3   import java.util.Arrays;
4   import java.util.Collection;
5   import java.util.HashSet;
6   import java.util.List;
7   import java.util.Set;
8   import java.util.SortedSet;
9   import java.util.TreeSet;
10
11  public class SetTest0 {
12
13      public static void main(String[] args) {
14          String[] colors1 = {"red", "green", "blue", "grey"};
15          Set<String> set1 = new HashSet(Arrays.asList(colors1));
16
17
18          String[] colors2 = {"red", "green", "blue", "white"};
19          Set<String> set2 = new HashSet(Arrays.asList(colors2));
20
21          Set result = new HashSet();
22          result.clear();
23          result.addAll(set1);
24          result.retainAll(set2);
```

```
25              System.out.println("交集：" + result);
26
27              result.clear();
28              result.addAll(set1);
29              result.removeAll(set2);
30              System.out.println("差集：" + result);
31
32              result.clear();
                result.addAll(set1);
                result.addAll(set2);
                System.out.println("聯集：" + result);
        }
}
```

這個物件可以使用的方法：

add( )

Adds an object to the collection.

clear( )

Removes all objects from the collection.

contains( )

Returns true if a specified object is an element within the collection.

isEmpty( )

Returns true if the collection has no elements.

iterator( )

Returns an Iterator object for the collection, which may be used to retrieve an object.

remove( )

Removes a specified object from the collection.

size( )

Returns the number of elements in the collection.

- 參考官方線上教學

https://docs.oracle.com/javase/tutorial/collections/interfaces/collection.html

- GitHub 有一本非常棒的入門書：Java SE 6 技術手冊

作者：良葛格 (林信良)

https://github.com/JustinSDK/JavaSE6Tutorial

特色：淺顯易懂 觀念清晰 讀完之後觀念通暢--知道怎麼做 也 知道為什麼