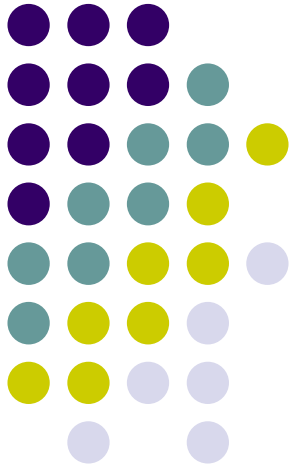
第十六章
Java collection集合物件

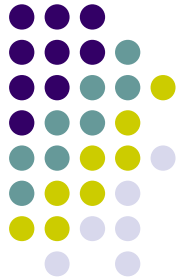


認識collection架構

認識並學習如何建立各種集合物件

學習利用Iterator介面的method走訪元素

利用ListIterator介面的method走訪元素

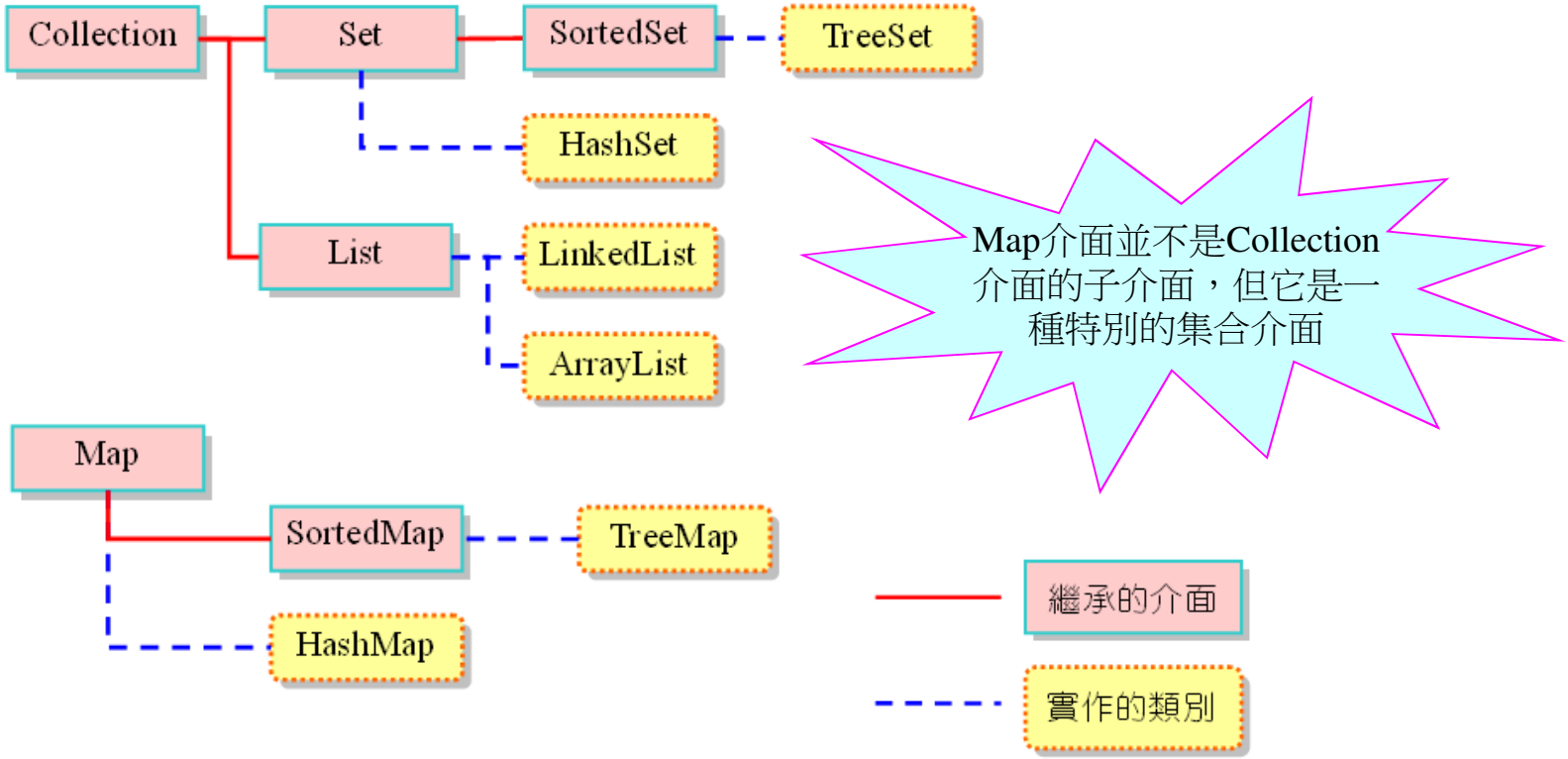


集合物件的概念

- 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 Collections Framework包括三個部分：
  - 介面（Interface）
  - 演算法（Algorithms）
  - 實作（Implementations）

collection介面的繼承關係圖

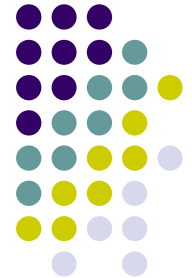
- 下圖是各種collection介面的繼承關係圖



泛型與collection

- 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
  - 泛型型態要使用原始資料型態的wrapper class
- 想要在TreeSet類別的集合物件裡儲存整數int型態的資料，可做出如下的宣告：

```
TreeSet<Integer> tset=new TreeSet<Integer>();
```



Set介面

- Set是集合的意思，在集合中的元素沒有特定的順序，但是元素不能重複出現，下表為Set介面常用的method

表 16.2.1　Set 介面常用的 method

method	主要功能
boolean add(E o)	將物件 o 新增為元素，成功時傳回 true
boolean addAll(Collection<? extends E> c)	將 Collection 的元素新增為此集合的元素，成功時傳回 true
void clear()	從集合中移除所有的元素
boolean contains(Object o)	當集合物件裡包含元素 o 時，傳回 true
boolean containsAll(Collection<?> c)	當集合物件裡包含 Collection 的元素 c 時，傳回 true
boolean isEmpty()	集合物件若沒有任何元素，傳回 true
boolean remove(Object o)	從集合物件中刪除物件 o，成功時傳回 true
boolean removeAll(Collection<?> c)	從集合物件中刪除 Collection 的元素 c，成功時傳回 true
boolean retainAll(Collection<?> c)	從集合物件中保留 Collection 的元素 c，其餘刪除，成功時傳回 true
int size()	傳回集合物件的元素個數
Iterator<E> iterator()	取得集合物件

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.HashSet<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)

- 想宣告一個泛型型態為Integer的HashSet類別之物件hset，可以寫如出下的敘述：

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

- 下面的範例說明如何使用HashSet類別：

```
01  // app16_1, 簡單的 HashSet 範例
02  import java.util.*;
03  public class app16_1
04  {
05     public static void main(String args[])
06     {
07        HashSet<String> hset=new HashSet<String>();
08
09        String str1="Puppy";
10        String str2="Kitten";
11        System.out.println("Hash empty: "+hset.isEmpty());
12        hset.add("Moneky");       // 增加元素
13        hset.add("Bunny");        // 增加元素
14        hset.add(str1);           // 增加元素
15        hset.add(str2);           // 增加元素
16
17        System.out.println("Hash size="+hset.size()); // 顯示元素個數
18        System.out.println("Hash empty: "+hset.isEmpty());
19        System.out.println("HashSet 內容:"+hset); // 顯示集合物件的內容
```

使用HashSet類別 (2/2)

```
20
21          hset.remove(str2);
22          System.out.println("清除 Kitten..., Hash size="+hset.size());
23
24          System.out.println("Hash 中是否有"+str2+"? "+hset.contains(str2));
25          System.out.println("Hash 中是否有 fish? "+hset.contains("fish"));
26          System.out.println("Hash 中是否有 Puppy? "+hset.contains("Puppy"));
27          hset.remove("Bunny");
28          System.out.println("清除 Bunny..., Hash size="+hset.size());
29
30          System.out.println("HashSet 內容:"+hset);
31          hset.clear();
32          System.out.println("清除 Hash 中所有的物件...");
33          System.out.println("Hash empty: "+hset.isEmpty());
34      }
35  }
```

```
/* app16_1 OUTPUT----------------------------

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

SortedSet介面

- SortedSet介面
  - 資料會由小而大排列
  - 為一種排序集合物件（sorted collection）
  - SortedSet介面的method：

表 16.2.3　SortedSet 介面的 method

method	主要功能
E first()	取得集合物件中的第一個元素
SortedSet<E> headSet(E toElm)	取得小於 toElm 的 TreeSet 物件
E last()	取得集合物件中的最後一個元素
SortedSet<E> subSet(E fromElm, E toElm)	取得大於等於 fromElm，且小於 toElm 的 TreeSet 物件
SortedSet<E> tailSet(E fromElm)	取得大於等於 fromElm 的 TreeSet 物件

TreeSet類別 (1/2)

- TreeSet類別常用的建構元：

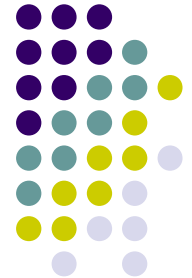
表 16.2.4　java.util.TreeSet<E> 類別的建構元

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

- app16_2是TreeSet的範例：

```
/* app16_2 OUTPUT-------------------------
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]
大於等於 10 的集合=[10, 12, 14, 16, 18, 20]
小於 8 的集合=[2, 4, 6]
----------------------------------------*/
```

```
01  // app16_2, 簡單的 TreeSet 範例
02  import java.util.*;
03  public class app16_2
04  {
05     public static void main(String args[])
06     {
07        TreeSet<Integer> tset=new TreeSet<Integer>();
08
09        for(int i=20;i>=2;i-=2)                    // 增加元素
10           tset.add(i);
```



TreeSet類別 (2/2)

```
11
12          System.out.println("元素個數="+tset.size());
13          System.out.println("集合內容="+tset);     // 顯示集合物件的內容
14
15          System.out.println("第一個元素="+tset.first());
16          System.out.println("最後一個元素="+tset.last());
17          System.out.println("介於 6 和 14 之間的集合="+tset.subSet(6,14));
18          System.out.println("大於等於 10 的集合="+tset.tailSet(10));
19          System.out.println("小於 8 的集合="+tset.headSet(8));
20      }
21  }
```

```
/* app16_2 OUTPUT-------------------------
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]
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

List介面

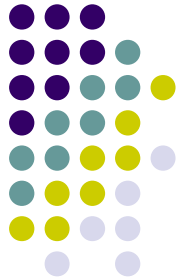
- List介面
  - 屬於有序集合物件（ordered collection）
  - 元素可以重複
  - 元素具有索引值（index）
- List介面和SortedSet介面的比較
  - List會依照索引值來排列元素的位置
  - SortedSet會根據元素本身的大小來排列

List介面的method

- 下表列出List介面裡常用的method：

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，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，並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，並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

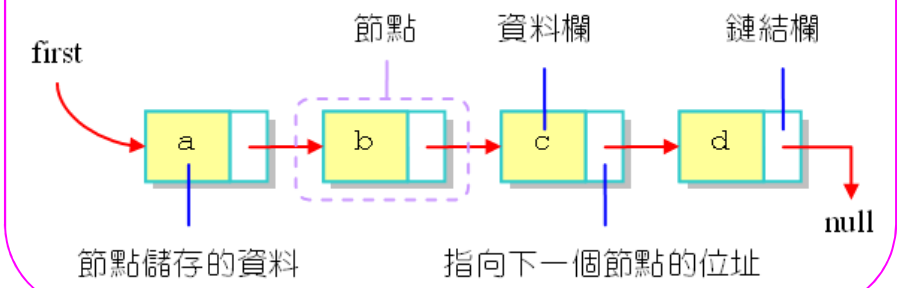


LinkedList類別

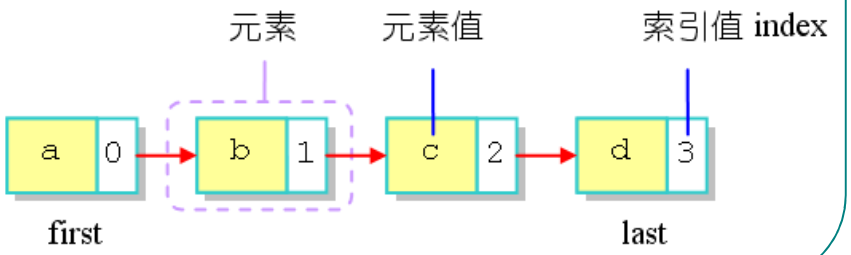
- 鏈結串列（linked list）
  - 節點（node）分為2個欄位
    - 資料欄
      - 資料欄儲存的是所需之資料
    - 鏈結欄
      - 鏈結欄記錄著下一個節點的位址
  - 鏈結串列的最後一個節點會指向null
    - 表示後面已沒有節點

鏈結串列

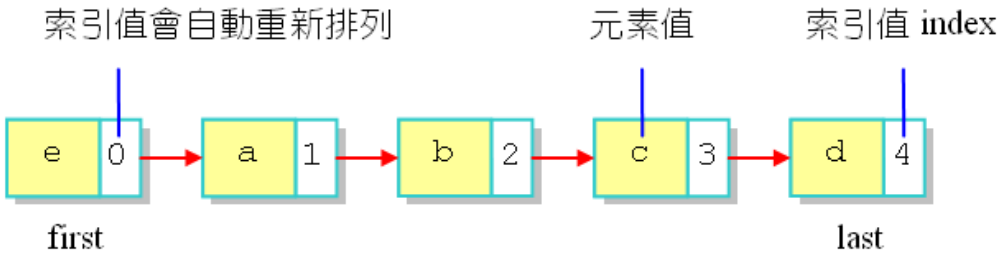
- 下圖為鏈結串列示意圖：



- 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method

- 下表列出LinkedList建構元與常用的method

表 16.3.2 java.util.LinkedList<E> 類別的建構元與 method

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

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

LinkedList的範例

- 範例app16_3是LinkedList的簡單範例

```
01  // app16_3, LinkedList 範例
02  import java.util.*;
03  public class app16_3
04  {
05     public static void main(String args[])
06     {
07        LinkedList<Integer> llist=new LinkedList<Integer>();
08
09        for(int i=10;i<=30;i+=10)          // 增加元素
10           llist.add(i);
11        llist.addFirst(100);
12        llist.addLast(200);
13        llist.addFirst(300);
14
15        System.out.println("元素個數="+llist.size());
16        System.out.print("LinkedList 的元素:");
17        for(int i=0;i<llist.size();i++)  // 顯示集合物件的內容
18           System.out.print(llist.get(i)+" ");
19
20        System.out.print("\n 刪除最後一個元素 ");
21        System.out.println(llist.removeLast()+"...");
22
23        System.out.println("第一個元素="+llist.getFirst());
24        System.out.println("最後一個元素="+llist.getLast());
25        System.out.println("元素值為 200 的索引值="+llist.indexOf(200));
26     }
27  }
```

```
/* app16_3 OUTPUT-------------------------
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
----------------------------------------*/
```

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，預設的元素個數為 10 個
ArrayList(Collection<? extends E> c)	建立一個包含特定的 Collection 物件 c 之 ArrayList 物件
ArrayList(int initialCapacity)	建立一個空的 ArrayList 物件，指定的元素個數為 initialCapacity 個

表 16.3.4　java.util.ArrayList 類別的 method

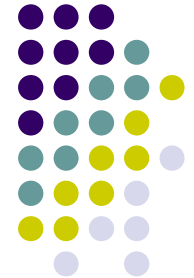
method	主要功能
void ensureCapacity(int minCapacity)	設定 ArrayList 物件的容量，元素的個數至少有 minCapacity 個
void trimToSize()	將 ArrayList 物件的容量剪裁成目前元素的數量

ArrayList類別的範例

```
/* app16_4 OUTPUT----------------------------------
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
-------------------------------------------------*/
```

```
01  // app16_4, ArrayList 範例
02  import java.util.*;
03  public class app16_4
04  {
05     public static void main(String args[])
06     {
07        ArrayList<Integer> alist=new ArrayList<Integer>();
08
09        for(int i=10;i<=50;i+=10)     // 增加元素
10           alist.add(i);
11        alist.add(3,200);
12        alist.add(0,300);
13        alist.add(400);               // 將 400 放在 alist 的最後一個位置
14
15        System.out.println("元素個數="+alist.size());
16        System.out.println("ArrayList 的元素:"+alist);
17        System.out.println("將索引值 1 的元素以 200 取代...");
18        alist.set(1,200);
19        System.out.println("ArrayList 的元素:"+alist);
20        System.out.print("第一個元素值為 200 的索引值=");
21        System.out.println(alist.indexOf(200));
22        System.out.print("最後一個元素值為 200 的索引值=");
23        System.out.println(alist.lastIndexOf(200));
24     }
25  }
```

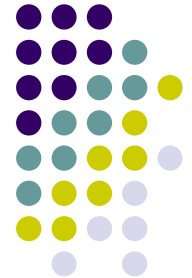
Map介面



- 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

method	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
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，關鍵值與對應值的泛型型態要以逗號分開，如：

```
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

```
/* app16_5 OUTPUT----------------------------------
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
-------------------------------------------------*/
```

```
01  // app16_5, HashMap 範例
02  import java.util.*;
03  public class app16_5
04  {
05     public static void main(String args[])
06     {
07        HashMap<Integer,String> hmap=new HashMap<Integer,String>();
08
09        hmap.put(94001,"Fiona");
10        hmap.put(94003,"Ariel");
11        hmap.put(94002,"Ryan");
12
13        System.out.println("元素個數="+hmap.size());
14        System.out.println("HashMap 的元素:"+hmap);
15        System.out.print("HashMap 中是否有關鍵值 94002? ");
16        System.out.println(hmap.containsKey(94002));
17        System.out.print("HashMap 中是否有對應值 Kevin? ");
18        System.out.println(hmap.containsValue("Kevin"));
19        hmap.remove(94001);
20        System.out.print("清除關鍵值 94001 的資料..., ");
21        System.out.println("元素個數="+hmap.size());
22        System.out.println("HashMap 的元素:"+hmap);
23        System.out.println("關鍵值 94003 的對應值="+hmap.get(94003));
24     }
25  }
```

app16_5是將物件加入HashMap之範例

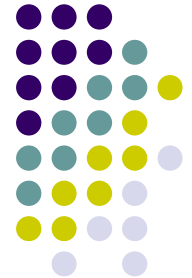
TreeMap類別

- 元素會依關鍵值由小至大排序
- 下表列出SortedMap類別的建構元及method

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

method	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

```
01  // app16_6, TreeMap 範例
02  import java.util.*;
03  public class app16_6
04  {
05     public static void main(String args[])
06     {
07        int k1=94001,k2=94003,key;
08        TreeMap<Integer,String> tmap=new TreeMap<Integer,String>();
09
10        tmap.put(94001,"Fiona");
11        tmap.put(94003,"Ariel");
12        tmap.put(94002,"Ryan");
13        tmap.put(94004,"Jack");
14
15        System.out.println("元素個數="+tmap.size());
16        System.out.println("TreeMap 的元素:"+tmap);
17        key=tmap.firstKey();
18        System.out.println("第一個元素= "+key+", "+tmap.get(key));
19        key=tmap.lastKey();
20        System.out.println("最後一個元素= "+key+", "+tmap.get(key));
21        System.out.print("介於"+k1+"和"+k2+"之間的 TreeMap=");
22        System.out.println(tmap.subMap(k1,k2));
23        System.out.print("大於等於"+k2+"的 TreeMap=");
24        System.out.println(tmap.tailMap(k2));
25     }
26  }
```

```
/* app16_6 OUTPUT-----------------------------------------------

元素個數=4
TreeMap 的元素:{94001=Fiona, 94002=Ryan, 94003=Ariel, 94004=Jack}
第一個元素= 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}
-------------------------------------------------------------*/
```

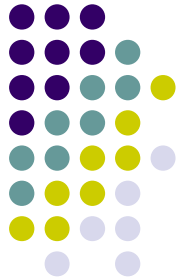
app16_6是將物件加入TreeMap之範例

使用Iterator介面走訪元素

- Iterator與ListIterator介面，皆可用來「走訪」或是刪除集合物件的元素
- 下表列出Iterator介面的method：

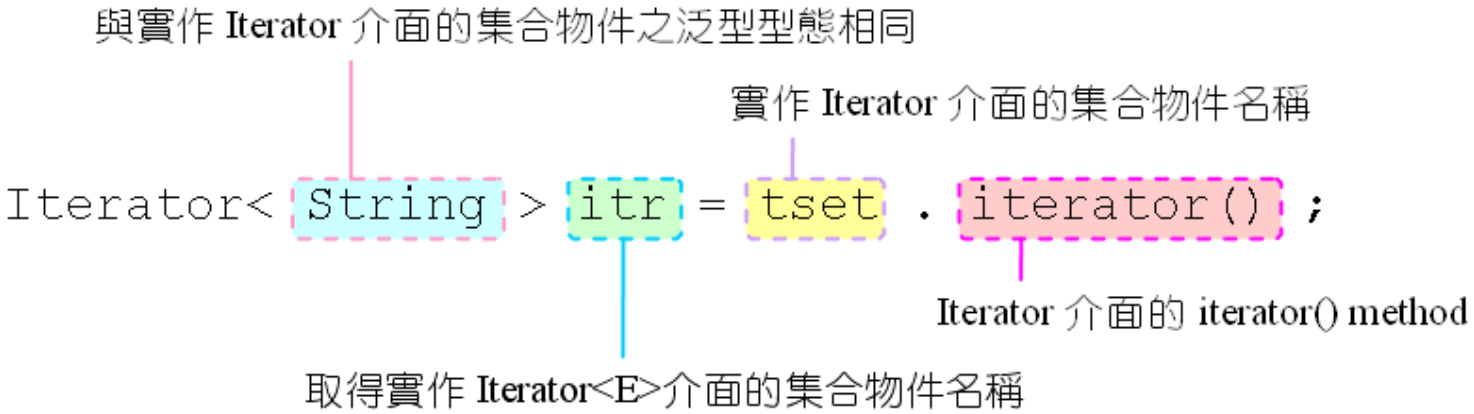
表 16.5.1 Iterator<E> 的 method

method	主要功能
Iterator<E> iterator()	取得實作 Iterator<E>介面的集合物件
boolean hasNext()	集合中若是有下一個元素，即傳回 true
E next()	傳回集合的下一個元素
void remove()	刪除集合中最後一個取得的元素



iterator() method

- 想走訪TreeSet<String>物件tset，程式的敘述及解釋如下圖所示：



走訪TreeSet元素

- 下面的範例是利用Iterator介面的method走訪元素：

```
01  // app16_7, 以 Iterator 走訪 TreeSet 元素
02  import java.util.*;
03  public class app16_7
04  {
05     public static void main(String args[])
06     {
07        TreeSet<String> tset=new TreeSet<String>();
08        String str="";
09        tset.add("Moneky");                  // 增加元素
10        tset.add("Bunny");                   // 增加元素
11        tset.add("Puppy");                   // 增加元素
12        tset.add("Kitten");                  // 增加元素
13
14        Iterator<String> itr=tset.iterator();
15        System.out.print("TreeSet 內容:");
16        while(itr.hasNext())                 // 走訪元素
17        {
18           str=itr.next();
19           System.out.print(str+" ");        // 印出元素內容
20        }
21
22        System.out.println("\n 刪除最後讀取的元素"+str+"...");
23        itr.remove();                        // 刪除最後讀取的元素
24        System.out.println("TreeSet 內容:"+tset);
25     }
26  }
```

```
/* app16_7 OUTPUT--------------------

TreeSet 內容:Bunny Kitten Moneky Puppy
刪除最後讀取的元素 Puppy...
TreeSet 內容:[Bunny, Kitten, Moneky]
---------------------------------------*/
```

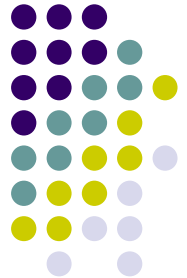
Iterator物件的讀取是單向的且只能讀取一次

使用ListIterator介面走訪元素

- ListIterator物件的走訪可以是雙向的
- 下表列出ListIterator介面的method：

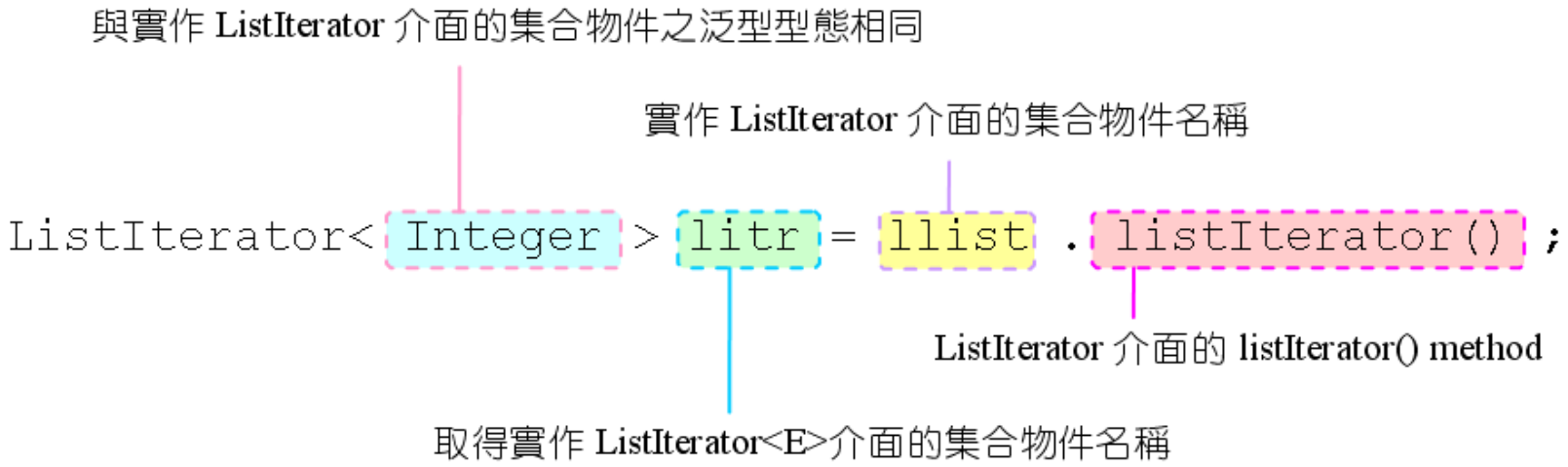
表 16.5.2 ListIterator<E> 的 method

method	主要功能
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
void add(E o)	在下一個元素前加入 o
boolean hasNext()	集合中若是有下一個元素，即傳回 true
boolean hasPrevious()	集合中若是有前一個元素，即傳回 true
E next()	傳回集合的下一個元素
int nextIndex()	傳回集合的下一個元素之索引值
E previous()	傳回集合的前一個元素
int previousIndex()	傳回集合的前一個元素之索引值
void remove()	刪除集合中最後一個取得的元素
void set(E o)	將集合中的最後一個元素以 o 取代



listIterator() method

- 下圖為listIterator() method的使用說明：



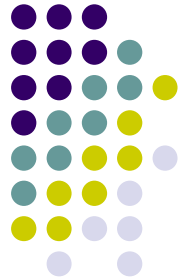
走訪LinkedList元素

```
01  // app16_8, 以 ListIterator 走訪 LinkedList 元素
02  import java.util.*;
03  public class app16_8
04  {
05     public static void main(String args[])
06     {
07        LinkedList<Integer> llist=new LinkedList<Integer>();
08
09        for(int i=10;i<=100;i+=10)                    // 增加元素
10           llist.add(i);
11
12        ListIterator<Integer> litr1=llist.listIterator();
13
14        System.out.print("正向列出 LinkedList 內容:");
15        while(litr1.hasNext())                        // 正向走訪元素
16           System.out.print(litr1.next()+" ");        // 印出元素內容
17        System.out.println();
18
19        ListIterator<Integer> litr2=llist.listIterator(llist.size());
20        System.out.print("反向列出 LinkedList 內容:");
21        while(litr2.hasPrevious())                    // 反向走訪元素
22           System.out.print(litr2.previous()+" "); // 印出元素內容
23        System.out.println();
24     }
25  }
```

app16_8是以ListIterator走訪LinkedList元素的範例

```
/* app16_8 OUTPUT----------------------------------

正向列出 LinkedList 內容:10 20 30 40 50 60 70 80 90 100
反向列出 LinkedList 內容:100 90 80 70 60 50 40 30 20 10
-------------------------------------------------*/
```



-The End-