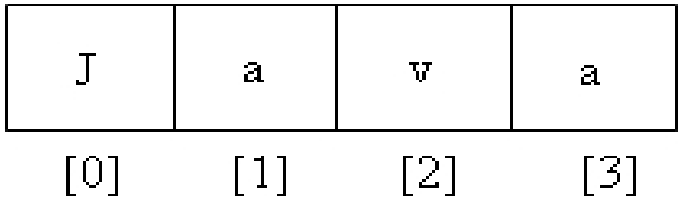
第五章　字串的使用

本章內容

- 5-1　字串的使用
- 5-2　使用　StringBuffer 類別
- 5-3　使用　StringBuilder 類別－ J2SE 5.0
- 5-4　使用　StringTokenizer 類別

5-1 字串的使用

- 字串是一種使用非常頻繁的資料型態。字串可以被視為一連串的字元所組合而成的資料型態。屬於「String」類別，該類別是位在「java.lang」套件下 。
- 由於字串內部是由字元陣列所組合而成的，因此，字串中的各個字元的排列方式也是和字元陣列相同。例如：字串「Java」的排列方式是:



5-1-1 初始及定義 String 物件

- 我們可以使用字串常數、字元陣列、位元陣列、StringBuffer 物件來建構字串物件。
- 在 J2SE 5.0 之後，我們還可以使用新增的 StringBuilder 物件來建構字串物件。
- 最簡單的建構字串物件的方式是直接傳入由「" "」包括的一組字串當作字串物件的初始值，或是僅使用「" "」來產生一個只含有空字串的字串物件，例如：

```
String str1 = new String(" 這是一組字串 ");
String str2 = new String("");      // 建構一個空字串
```

5-1-1 初始及定義 String 物件

- **傳入一個 char 陣列**
  - String 物件的建構子也可以接受其他型態的資料，例如：傳入 char 陣列的宣告方式為：

```
String 字串名稱 = new String(char[]
  value);
```

  - 例如：

```
char[] value = new char[] {' 學 ', ' 習 ', 'J',
  'A', 'V', 'A'};

String str1 = new String(value);
```

5-1-1 初始及定義 String 物件

- **傳入一個 char 陣列，並指定元素的內容**
  - 使用 char 陣列來初始 String 物件的內容時，不一定要將 char 陣列中的所有元素都當做是 String 物件的初始內容，例如：傳入 char 陣列，並指定元素值範圍的宣告方式為：

```
String 字串名稱 = new String(char[] value, int offset, int
    count);
```

  - 「offset」用來指定 char 陣列的開始位置的索引值，而「count」則是指定需要取用的元素個數。以下的範例程式示範如何使用 char 陣列來將 String 物件的內容初始為「JAVA」：

```
char[] value = new char[] {'學', '習', 'J', 'A', 'V', 'A'};
String str1 = new String(value, 2, 4);
```

5-1-1 初始及定義 String 物件

- 傳入一個 **byte** 陣列
  - String 物件的建構子也可以接受 byte[] 型態的資料，宣告方式為：

```
String 字串名稱 = new String(byte[] value);
```

  - 以下的範例程式示範如何使用 byte 陣列來初始化陣列：

```
byte[] value = “學習 JAVA”.getBytes();
String str1 = new String(value);
```

5-1-1 初始及定義 String 物件

- **傳入一個 StringBuffer 物件**
  - 必要時，我們也可以使用「StringBuffer」物件來當做是 String 物件的初始內容，例如：傳入「StringBuffer」物件宣告方式為：

    String 字串名稱 = new String(StringBuffer buffer);

  - 以下的範例程式示範如何使用 StringBuffer 物件來將 String 物件的內容初始為「JAVA」：

```
StringBuffer buffer = new StringBuffer("JAVA");
String str3 = new String(buffer);
```

5-1-1 初始及定義 String 物件

- **傳入一個 StringBuilder 物件**
  - J2SE5.0 所提供的新的 StringBuilder 類別也可以用來當做是 String 物件的初始內容，例如：傳入「StringBuffer」物件宣告方式為：

    String 字串名稱 = new String(StringBuilder buffer);

  - 以下的範例程式示範如何使用 StringBuilder 物件來將 String 物件的內容初始為「JAVA」：

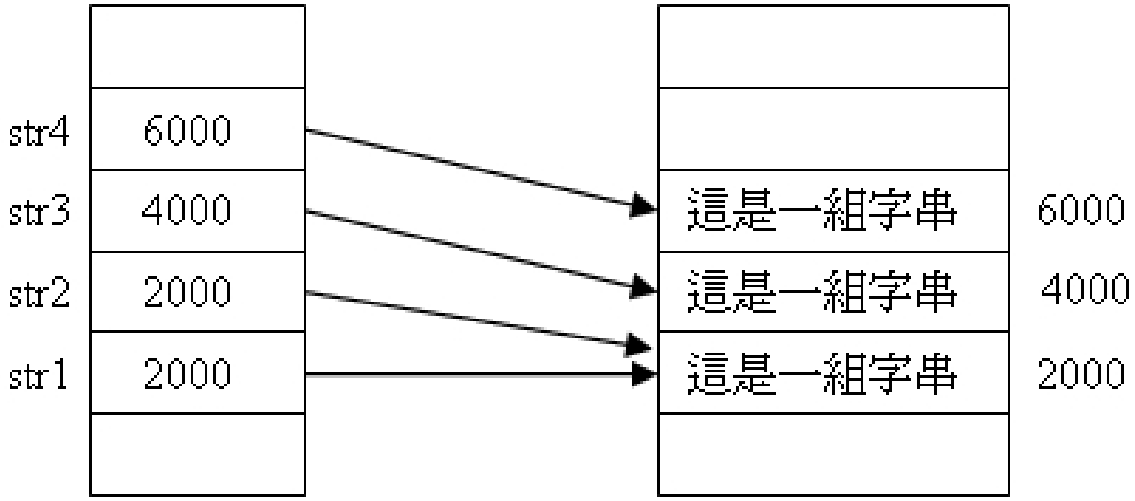
    StringBuffer buffer = new StringBuilder ("JAVA");

    String str3 = new String(buffer);

5-1-1 初始及定義 String 物件

- 字串建構的範例

```
String str1 = " 這是一組字串 ";
String str2 = " 這是一組字串 ";
String str3 = new String(" 這是一組字串 ");
String str4 = new String(" 這是一組字串 ");
Stri
```



5-1-2 String 物件的內容是不能更改的

- 「String」物件一旦建立後，其內容就無法再「**更改**(immutable)」了。
- 即使您將該 String 變數重新指定新的內容， Java 的運作方式是新建立一個 String 物件，再將新的 String 物件指定給 String 變數。
- 因為 String 物件的儲存是存放在 Hash Table 中

5-1-3 String 物件的常用方法

方法	功用
char charAt(int index)	取行字串中 index 位置的字元
int compareTo (String anotherString)	依字母順序比較兩個字串是否完全相同
int compareToIgnoreCase (String anotherString)	依字母順序比較兩個字串是否相同，比較時忽略字母的大小寫
String concate(String str)	將傳入的 str 字串連在呼叫該方法的字串後方
static String copyValueOf (char[] data)	將字元陣列 data 的內容複製到呼叫該方法的字串中
boolean equals (Object anObject)	比較字串物件和 anObject 物件是否相同
byte[] getBytes()	將字串以位元陣列的型態傳回
int hashCode()	傳回字串的 hashCode
int indexOf(int ch)	傳回 ch 字元在字串中的位置
int length()	取得字串的長度

5-1-3 String 物件的常用方法…

方法	功用
String replace (char oldChar, char newChar)	將字串中的 oldChar 字元轉換成 newChar 字元
String substring (int beginIndex)	從字串中第 beginIndex 位置開始，取出子字串
String substring (int beginIndex, int endIndex)	取出字串中從 beginIndex 位置到 endIndex 位置的子字串
char[] toCharArray()	將 String 物件的內容轉換為 char 陣列並傳回
String toString()	傳回 String 物件的本身的內容
String toLowerCase()	將 String 物件中的英文字轉為小寫並傳回
String toUpperCase()	將 String 物件中的英文字轉為大寫並傳回
String trim()	將字串兩端的空格移除
static String valueOf (Object obj)	obj 可以是其他的物件或是基本資料型態的變數，將 obj 轉換為字串

5-1-4 字串屬性的取得及轉型

- 我們可以使用「length」來取得字串的長度。
- 必要時，可以使用「toCharArray」方法將字轉換為字元陣列。
- 利用「getBytes」方法可以將字串轉換為位元陣列。
- 如果要進行字元大小寫的轉換，我們可以使用「toLowerCase」或是「toUpperCase」等方法。

5-1-5 字串內容的取得及搜尋

- 利用「charAt」方法可以從字串中取出某個索引值位置的字元。
- 「indexOf」方法可以用來在字串中搜尋特定的子字串。
- 「indexOf()」和「lastIndexOf()」方法尚有其他的同名異式的函式，但皆是用於字串的搜尋。

5-1-6 字串內容的處理

- 除了使用「+」運算子之外，我們可以使用「concate」方法來進行字串連接的動作。
- 「replace」方法可以用來將字串中的某個字元取代成另一個字元。
- 「substring」方法可以用來取出子字串。
- 利用「split」方法，我們可以依照某個規則來分割字串。

5-2 使用 StringBuffer 類別

- 如果您要處理一個需要時常變更內容的字串時，直接利用 String 物件會比較不適合。
- 在處理一個需要時常變更內容的字串時，一般的做法是會利用 StringBuffer 類別來處理會變更內容的字串物件。
- 處理完成後，再將它轉換為 String 物件。
- 因為，在處理固定不變的字串物件時， String 類別的效能是優於 StringBuffer 類別。

5-2 使用 StringBuffer 類別…

- StringBuffer 提供了三個建構子:

建構子	作用
StringBuffer()	建構出不含資料的 StringBuffer 物件，預設情況下， Java 會將該物件初始化成可以儲存 16 個字元的物件
StringBuffer(int length)	自行定義儲存字元的長度，不含資料
StringBuffer(String str)	以傳入的 str 參數建構出 StringBuffer 物件

- StringBuffer 物件是可以變更長度及內容的，使用 StringBuffer 時如果需要更大的容量時， JVM 會自動配置更多的儲存空間。

5-2 使用 StringBuffer 類別…

- 定義出 StringBuffer 物件的方式有：

```
// 定義一個可以儲存 16 字元的 StringBuffer 物件
StringBuffer str1 = new StringBuffer();
// 定義一個可以儲存 30 字元的 StringBuffer 物件
StringBuffer str2 = new StringBuffer(30);
// 定義一個 StringBuffer 物件，內容為「 Hi 」
StringBuffer str3 = new StringBuffer("Hi");
```

- 更改 StringBuffer 物件的內容並不會產生一個新的物件，這點和 String 物件不同。

5-2-1 StringBuffer 的方法

- StringBuffer 類別提供的方法和 String 類別中的方法的使用大致相同，但 StringBuffer 類別中提供額外的字串的操作方法有：

方法	功用
StringBuffer append (Object obj)	將 obj 物件增加到 StringBuffer 物件的緩衝區後
int capacity()	取得 StringBuffer 緩衝區的大小
StringBuffer delete (int start, int end)	移除緩衝區中由 start 到 end 位置的字串
StringBuffer deleteAt (int index)	移除緩衝區中 index 位置的字元
StringBuffer insert (int offset, Object obj)	將 obj 物件加入 StringBuffer 物件中的 offset 位置之後
StringBuffer reverse()	將緩衝區中的字串反向排列

5-2-2 使用 StringBuffer 來操作字串

- 字串的操作不外乎是新增、刪除、修改、查詢等作業方式，我們用以下的範例來實際說明如何使用 StringBuffer 類別中的方法：

```
// 建構 StringBuffer 物件
StringBuffer sb = new StringBuffer("This is a book");
// 使用 append 新增字串緩衝區的內容
sb.append(", or a magazine");
// 使用 insert 新增字串到緩衝區中的特定位置
sb.insert(0, "I guess ");
// 使用 delete 刪除字串緩衝區的內容
sb.delete(0, 8);
// 使用 replace 取代字串緩衝區的內容
// 將「That」更改為「This」
sb.replace(0, 4, "This");
// 使用 reverse 將字串緩衝區的內容反轉
sb.reverse();
```

5-2-3 少用 "+" 運算子處理 String 物件

- 「＋」運算子可以用來將兩個數值相加。但如果「＋」運算子前後的任何運算元是 String 物件，則 Compiler 會將另一個運算元的型別轉換為「String」型別，「＋」運算子再將前後兩個運算元做「相連」的處理
- 如果深究「＋」運算子使用於字串相連的運算時，運算式實際的處理過程是使用 StringBuffer 物件來處理字串的相連動作。
- 所以，應該直接使用 StringBuffer 物件來處理字串的相連動作。

5-3 使用 StringBuilder 類別－J2SE 5.0

- 新版的 Java 建議您使用 StringBuilder 類別來取代 StringBuffer 類別以便得到更好的程式執行效能。
- 舊有程式只要修改宣告式即可：
- 例如：將下列的程式碼：

```
StringBuffer sb = new StringBuffer();
```

- 修改成：

```
StringBuilder sb = new StringBuilder();
```

5-3 使用 StringBuilder 類別－J2SE 5.0..

- 使用 StringBuilder 的執行效率是高於 StringBuffer，其主要的原因是在於 StringBuilder 物件是「不受保護」的。
- 「不受保護」並不是指使用 StringBuilder 物件時，資料容易損毀，而是指 StringBuilder 不適合使用於多執行緒 (Thread) 的環境之下。
- 但是，StringBuilder 物件的使用就沒有這樣的保護機制了。正因為如此，StringBuilder 物件的執行效能會優於 StringBuffer 物件。

5-4 使用 StringTokenizer 類別

- 「StringTokenizer」類別可以用來將字串資料的內容分解。
- 「StringTokenizer」類別位於「java.util」套件之下，該類別的建構子有:

建構子	功用
StringTokenizer(String str)	利用傳入的 str 字串建構 StringTokenizer 物件，預設情況下，會以空白符號分解字串
StringTokenizer (String str, String delim)	利用 delim 的符號分解傳入的 str 字串並建構 StringTokenizer 物件，
StringTokenizer (String str, String delim, boolean b)	同上一個建構子，但如果將 b 設定為 true，則 delim 也會被視為 token 而傳回

5-4 使用 StringTokenizer 類別…

- StringTokenizer 物件的建構方式可以是:

```
String s1 = "Today is a sunny day";
String s2 = "Today, is, a, sunny, day";
String s3 = "Today\\is\\a\\sunny\\day";
StringTokenizer st1= new StringTokenizer(s1);
StringTokenizer st2= new StringTokenizer(s2, ",");
StringTokenizer st3= new StringTokenizer(s3, "\\",
   true);
```

5-4 使用 StringTokenizer 類別…

- StringTokenizer 類別提供的方法有:

方法	功用
int countTokens()	傳回 StringTokenizer 物件中剩下的 Tokens 個數
boolean hasMoreTokens()	判斷 StringTokenizer 物件中是否還有下一個 Tokens
String nextToken()	取得 Tokens
String nextToken(String delim)	用新的 delim 符號取得 Tokens