國立屏東大學 資訊工程學系 物件導向程式設計

10. Strings

- String類別
- StringBuffer類別
- StringBuilder類別
- StringTokenizer類別

字串(Strings)是程式設計時常用使用到的一種資料型態，可惜在許多程式語言裡並不支援字串型態，您必須以字元陣列或字元指標的方式來使用字串Java語言雖然也不提供字串型態，但它提供了一個類別幫助我們使用字串，這個類別就是 – java.lang.String關於類別的說明請參考本書第11章，再此您可以先暫時將String視為資料型態，等到瞭解類別的概念後，再行探討。

10.1 String類別

String類別包裹在java.lang套件中，所以在使用前不需要以import載入。

10.1.1 宣告與初始值設定

以下是宣告的範例：

```
String str = "abc";
```

您也可以先宣告，再給定其值，例如：

```
String str;

str = "abc";
```

事實上String str是將str宣告為一個reference它應該要指向一個在記憶體中的「字元陣列」，而在記憶體內的「字元陣列」一旦產生，其值是唯讀的、是不可修改的(immutable)如果需要修改，必須使用String類別的method來進行。

> 不是說Java語言的字串是immutable的，那為什麼下面這段程式碼是正確的？

```
String str = "ABC";
```

```
str = "xyz";
```

> 看清楚「事實上String str是將str宣告為一個reference它應該要指向一個在記憶體中的「字元陣列」，而在記憶體內的「字元陣列」一旦產生，其值是唯讀的、是不可修改的(immutable)。如果需要修改，必須使用String類別的method來進行。

> 看懂了嗎？
>
> String str = “ABC”會在記憶體內產生一個字元陣列，其值為“ABC”並且讓str這個reference指向它所在的位置。此時“ABC”字串是不可以再修改的，除非你使用String類別提供的method來操作。
>
> str = “xyz”會在記憶體內產生另一個字元陣列，其值為“xyz”所以記憶體內現在有兩個字元字串了，然後str=“xyz”會讓str這個reference指向“xyz”字串所在的位置。還是一樣，這兩個字串都是唯讀的，除非你使用String類別提供的method來操作。
>
> 那.... “ABC”字串不就放在記憶體內，沒人管它？
>
> Java語言使用動態記憶體管理garbage collection方法，會自己監測記憶體的使用狀況，這種沒有任何reference指向它的記憶體空間，是會被自動回收的，請不用擔心。

熟悉C語言的程式設計師已經習慣了兩種字串的表達方法char *與char[]也就是字串指標與字串陣列。由於Java語言沒有指標，所以當然不支援char *字串。另一方面char[]在java語言中還是字元陣列，當然也可以做為字串的一種表達方式。不過Java語言預設會以String類別來做為字串，所以建議儘可能使用String類別。

請參考以下的例子：

```
char[] str1 = {'a', 'b', 'c' };
String str2 = "def";

System.out.println(str1);
System.out.println(str1 + ":" + str2);
System.out.printf("%s:%s\n", str1, str2);
```

其結果為:

```
abc
[C@4ac5c32e:def
[C@4ac5c32e:def
```

注意到了嗎System.out.println(str1)時，可以將str1這個char[]視為字串並正確它輸出。而在接下來的兩行中str1就被輸出成[C@4ac5c32e這裡所印出的其實是str1的Object identifier(物件識別碼)，而不是“abc”

<note tip>

Object identifier(物件識別碼)

在Java語言中，萬事萬物都被視為物件，而每個物件都會有一個獨一無二的Objectidentifier(物件識別碼)。

</note>

如果您還是習慣使用char[]字串，那請以下列方法生成String類別的字串：

```
char[] charArrayString = {'a', 'b', 'c' };
String str2 = new String(charArrayString);
```

10.1.2 字串操作

String類別提供許多操作字串的method，請自行參閱文件。我們列舉一些常用的字串操作method如下：

int length()	傳回字串的長度，也就是字串中有幾個字元
char charAt(int index)	取出字串中index位置的字元
int indexOf(int ch)	在字串中尋找ch字元第一次出現的位置，若找不到則傳回-1
int indexOf(int ch, int fromIndex)	由第fromIndex位置開始，在字串中尋找ch字元第一次出現的位置，若找不到則傳回-1
int indexOf(String str)	在字串中尋找str字串出現的位置，若找不到則傳回-1
int indexOf(String str, int fromIndex)	由第fromIndex位置開始，在字串中尋找str字串第一次出現的位置，若找不到則傳回-1
String concate(String str)	將傳入的str字串連接在後面
static String copyVauleOf(char[] data)	將data複製到字串裡
int compareTo(String anotherString)	自第一個字元起，依序與anotherString比較，遇到第一個不同的字元時，則傳回兩者的整數值之差；若所有字元皆相等，則傳回0
int cpmpareTolgnoreCase(String anotherString)	同上，但大小寫視為相同字元
boolean equals(Object anObject)	若anObject為字串物件且內容與相等，則傳回true，否則就傳回false
boolean equalsIgnoreCase(Object anObject)	同上，但大小寫視為相同字元
boolean isEmpty()	如果字串為空字串則傳回true，否則傳回false
boolean startsWith(String prefix)	若字串是以prefix開頭，則傳回true，其它情形則傳回false
boolean startsWith(String prefix, int toffset)	同上，但從toffset位置處開始判斷
boolean endsWith(String suffix)	若字串是以suffix結尾，則傳回true，其它情形則傳回false
byte[] getBytes()	將字串以位元陣列的型式傳回
String replace(char oldChar,char newChar)	將字串中出現oldChar字元，替換為newChar字元
String substring(int beginIndex)	傳回由beginIndex位置開始到字串結尾的子字串

String substring(int beginIndex, int endIndex)	傳回由beginInedx位置開始到endInedx位置的子字串
char[] toCharArray()	將字串以字元陣列的型式傳回
String toString()	傳回字串本身
String toLowerCase()	將字串中的英文字母全部轉為小寫後傳回
String toUpperCase()	將字串中的英文字母全部轉為大寫後傳回
String trim()	將字串兩端的空白字元移除

= 範例 =

```
        String str1="abcdefghijklmnopqrstuvwxyz";
        String str2="A Santa at NASA.";
        String str3="abcde";


        System.out.println(str1.length());  //要注意是 length()不是length
        System.out.println(str1.charAt(18));
        System.out.println(str1.indexOf('t'));
        System.out.println(str2.indexOf("at"));
        System.out.println(str2.indexOf('a', 4));
        System.out.println("abcefg".length());
        System.out.println("0123456789".charAt(5));
        System.out.println(str2 + str3);
        System.out.println("abc".substring(2,3));
        System.out.println(str3.substring(1, 2));
```

```
26
s
19
8
6
6
5
A Santa at NASA.abcde
c
b
```

10.2 StringBuffer與StringBuilder類別

String類別是immutable的字串，我們不能改變其內容java.lang.StringBuffer與java.lang.StringBuilder則是Mutable的字串，其字串預設保有16個字元的空間，透過insert()delete()append()replace()等method我們可以改變字串的內容。當預設的空間不足或過多時，它會自動增加或縮減空間。

StringBuffer與StringBuilder的差異在於StringBuffer是設計供多執行緒的情況下使用的，具有同步機制，確保字串在多執行緒環境下亦不會發生錯誤。但StringBuilder則是沒有同步機制，適用於單執行緒的情況。所以如果在單執行緒下，您應該使用StringBuilder以獲得較好的效能；如果是在多執行緒下操作，則應該

使用StringBuffer避免發生同步的問題。

在StringBuilder中，length()可傳回目前物件中的字元長度，而capacity()可傳回該物件目前可容納的字元容量，下面這個程式是個簡單的示範：

```
        StringBuilder strBuilder = new StringBuilder("You are beautiful!");

        System.out.println(strBuilder);
        System.out.println("length: " + strBuilder.length());
        System.out.println("capacity: " + strBuilder.capacity());

        strBuilder.insert(8, "so ");

        System.out.println(strBuilder);
        System.out.println("length: " + strBuilder.length());
        System.out.println("capacity: " + strBuilder.capacity());

        strBuilder.append(" We are crazy for you!");
        System.out.println(strBuilder);
        System.out.println("length: " + strBuilder.length());
        System.out.println("capacity: " + strBuilder.capacity());
```

執行結果：

```
You are beautiful!
length: 18
capacity: 34
You are so beautiful!
length: 21
capacity: 34
You are so beautiful! We are crazy for you!
length: 43
capacity: 70
```

10.3 StringTokenizer類別

StringTokenizer類別是java.util套件中的類別，可用以將字串拆解成tokens在許多的程式應用裡，這是一個常用到的類別。

請參考下面的例子(不要忘記載入java.util.StringTokenizer)

```
        String str = "This is a test for StringTokenizer.";

        StringTokenizer st = new StringTokenizer(str);
```

```
        while (st.hasMoreTokens())
        {
            System.out.println(st.nextToken());
        }
```

執行結果：

```
This
is
a
test
for
StringTokenizer.
```