

字串 String

- 字串就是 String 物件，因此，宣告一個字串變數時會先指到一個 String 的參照，再產生一個字串物件
- String 類別定義的常用建構方法

建構方法	說明
String()	建立一個空的字串
String(char[] value)	由 value 所指的字元陣列建構字串
String(char[] value, int offset, int count)	由 value 所指的字元陣列中第 offset 個元素開始, 取出 count 個字元來建構字串
String(String original)	建立 original 所指 String 物件的副本
String(StringBuffer buffer)	由 StringBuffer 物件建構字串
String(StringBuilder builder)	由 StringBuilder 物件建構字串

字串 String (Cont.)

```
5 import java.io.*;
6
7 class Test{
8     static final int x = 0;
9     public static void print(){
10         System.out.println("呼叫 static method");
11     }
12 }
13 public class StaticMethod{
14     public static void main(String[] argv) throws IOException{
15         BufferedReader br =
16             new BufferedReader(new InputStreamReader(System.in));
17         char[] test = {'亞', '洲', '大', '學', '資', '訊', '工', '程', '學', '系'};
18         String a = new String();
19         String b = new String(test);
20         String c = new String(test, 3, 4);
21         String d = new String(b);
22         System.out.println("a:"+a);
23         System.out.println("b:"+b);
24         System.out.println("c:"+c);
25         System.out.println("d:"+d);
26         System.out.println("b == d -> "+(b==d));
27     }
28 }
```

字元陣列中索引碼
為 3 的元素開始, 取
出 4 個元素建構字串

由剛剛建立的
字串 b 產生副本

雖然字串 d 和字串 b 的內容一樣, 但卻是不同的物件個體,
所以 == 運算比較參照值的結果並不相等。如果要進行字串
內容的比對, 必須使用 equals() 方法

```
a:
b:亞洲大學資訊工程學系
c:學資訊工
d:亞洲大學資訊工程學系
b == d -> false
```

字串 String (Cont.)

- Java 對 String 類別的特別支援
 - 和陣列一樣，使用字面常數來產生物件

```
5 import java.io.*;
6
7 class Test{
8     static final int x = 0;
9     public static void print(){
10         System.out.println("呼叫 static method");
11     }
12 }
13 public class StaticMethod{
14     public static void main(String[] argv) throws IOException{
15         BufferedReader br =
16             new BufferedReader(new InputStreamReader(System.in));
17         String tmpStr = "這是一個測試字串";
18         String tmpStr1 = "這是一個測試字串";
19         String tmpStr2 = new String("這是一個測試字串");
20         System.out.println("tmpStr == tmpStr1 -> "+(tmpStr==tmpStr1));
21         System.out.println("tmpStr1 == tmpStr2 -> "+(tmpStr1==tmpStr2));
22         System.out.println("tmpStr == tmpStr2 -> "+(tmpStr==tmpStr2));
23     }
24 }
```

```
tmpStr == tmpStr1 -> true
tmpStr1 == tmpStr2 -> false
tmpStr == tmpStr2 -> false
```

直接使用字面常
數建立物件

設定給 b 的參照值其
實和給 a 的是一樣的

當程式中有字面常數時,Java
編譯器其實會產生一個 String
物件來代表所有相同內容的字
面常數字串

字串 String (Cont.)

- 如果要比對字串內容，必須使用 String 類別中的 equals() 方法

```
5 import java.io.*;
6
7 class Test{
8     static final int x = 0;
9     public static void print(){
10         System.out.println("呼叫 static method");
11     }
12 }
13 public class StaticMethod{
14     public static void main(String[] argv) throws IOException{
15         BufferedReader br =
16             new BufferedReader(new InputStreamReader(System.in));
17         String tmpStr = "這是一個測試字串";
18         String tmpStr1 = "這是一個測試字串";
19         String tmpStr2 = new String("這是一個測試字串");
20         System.out.println("tmpStr equals to tmpStr1? -> "+tmpStr.equals(tmpStr1));
21         System.out.println("tmpStr1 equals tmpStr2? -> "+tmpStr1.equals(tmpStr2));
22         System.out.println("tmpStr equals to == tmpStr2? -> "+tmpStr.equals(tmpStr2));
23     }
24 }
```

```
tmpStr equals to tmpStr1? -> true
tmpStr1 equals tmpStr2? -> true
tmpStr equals to == tmpStr2? -> true
```

字串 String (Cont.)

- equals() 方法在比對字串時，如果內容是英文字母，則大小寫會被視為不同，例如 『 abc 』 跟 『 ABC 』 是不同的
- 如果要比較英文字串又不想管大小寫，則可用 equalsIgnoreCase() 方法

```
12 public class StaticMethod{
13     public static void main(String[] argv) throws IOException{
14         BufferedReader br =
15             new BufferedReader(new InputStreamReader(System.in));
16         String tmpStr = "Asia university";
17         String tmpStr1 = "ASIA UNIVERSITY";
18         String tmpStr2 = new String("這是一個測試字串");
19         System.out.println("tmpStr equals to tmpStr1? -> "+tmpStr.equals(tmpStr1));
20         System.out.println("tmpStr1 equals tmpStr2? -> "+tmpStr1.equals(tmpStr2));
21         System.out.println("tmpStr equals to == tmpStr2? -> "+tmpStr.equals(tmpStr2));
22     }
23 }
```

```
tmpStr equals to tmpStr1? -> false
tmpStr1 equals tmpStr2? -> false
tmpStr equals to == tmpStr2? -> false
```

字串 String (Cont.)

- 連接運算

- 當運算元中有字串資料時，『 + 』運算子會進行文字串接的動作

- 自動轉型 (Implicit Conversion)

- 如果在進行連接動作中有非字串的運算元，則 Java 會嘗試將該運算元轉換成字串物件，轉換方式是呼叫該運算元的 toString() 方法

```
5 import java.io.*;
6 class Test{
7     static final int x = 0;
8     public static void print(){
9         System.out.println("呼叫 static method");
10    }
11    public String toString(){
12        return "This is a test object";
13    }
14 }
15 public class StaticMethod{
16     public static void main(String[] argv) throws IOException{
17         Test t = new Test();
18         String a = "Testing ";
19         String tmpStr = a + t;
20         System.out.println("a: -> "+a);
21         System.out.println("tmpStr: -> "+tmpStr);
22     }
23 }
```

★注意★ toString() 方法
必須傳回 String 物件, 而且
必須加上 public 存取控制

```
a: -> Testing
tmpStr: -> Testing This is a test object
```

字串 String (Cont.)

- String 物件一旦建立後其內容無法更改，即使是連接字串也是將字串物件予以接連後產生第三個字串物，並將該物件做為輸出
- String 類別中提供的所有方法均是回傳一個新的物件，而不是更改舊物件後傳回
- 如果必須直接修改字串物件的內容，則必須藉由 StringBuffer 或 StringBuilder 類別的協助

字串 String (Cont.)

- String 類的方法：

- charAt(int index)

- 傳回 index 索引所指到的字元，字串與陣列一樣，起始索引為 0

```
5 import java.io.*;
6 public class CaesarCipher{
7     public static void main(String[] argv)
8         throws IOException{
9         BufferedReader br =
10             new BufferedReader(new InputStreamReader(System.in));
11         System.out.print("請輸入加密金鑰->");
12         int key = Integer.parseInt(br.readLine());
13         System.out.print("請輸入明文->");
14         String planTex = br.readLine();
15         int texLen = planTex.length();
16         char[] data = new char[texLen];
17         for(int i=0;i<texLen;i++){
18             data[i] = planTex.charAt(i);
19             System.out.println(data[i]);
20         }
21     }
22 }
```

```
請輸入加密金鑰->3
請輸入明文->asia
a
s
i
a
```


字串 String (Cont.)

- `int compareTo(String anotherString)`
 - 以逐字方式 (Lexically) 與 `anotherString` 所指的字串進行比較，比較結果會是列三種之一
 - 如果 `anotherString` 比較大，則回傳負值
 - 如果與 `anotherString` 內容一模一樣，則回傳 0
 - 如果 `anotherString` 比較小，則回傳正值
 - 如何比大小？比較規則如下
 - 由初始索引位置 0 開始，兩字串相同索引碼對應到的字元，直到同一索引但各別字元不同時，則以萬國碼 (Unicode) 大者為大
 - 例：`tmpStr1 = "abcde"`, `tmpStr2 = "abedc"`，則在索引位置 2 時，`tmpStr1` 的為 c 而 `tmpStr2` 的為 e，由於 e 的 unicode 比 c 的大，`tmpStr1.compareTo(tmpStr2)` 的結果是負值

字串 String (Cont.)

- 若兩字串長度相同且內容一模一樣，則輸出為 0
- 例：tmpStr1 = "abcde", tmpStr2 = "abcde"，則 tmpStr1.compareTo(tmpStr2) 的結果是 0
- 若兩字串長度不同，則逐一比對兩字串相同索引的字元 Unicode，若比較到短的字串結束，且其與較長的字串前段相同，則以較短的字串為小
- 例：tmpStr1 = "abc", tmpStr2 = "abcde"，則 tmpStr1 小於 tmpStr2

```
15 public class StaticMethod{
16     public static void main(String[] argv) throws IOException{
17         String tmpStr = "abcd";
18         System.out.println(tmpStr.compareTo("abcb"));
19         System.out.println(tmpStr.compareTo("abcd"));
20         System.out.println(tmpStr.compareTo("abce"));
21         System.out.println(tmpStr.compareTo("abcde"));
22         System.out.println(tmpStr.compareTo("Abcd"));
23     }
24 }
```

```
2
0
-1
-1
32
```

字串 String (Cont.)

- `int compareToIgnoreCase(String anotherString)`
 - 即進行比對時忽略字每大小寫，將其視為相同

```
15 public class StaticMethod{  
16     public static void main(String[] argv) throws IOException{  
17         String tmpStr = "abcd";  
18         System.out.println(tmpStr.compareTo("abcb"));  
19         System.out.println(tmpStr.compareTo("abcd"));  
20         System.out.println(tmpStr.compareToIgnoreCase("ABCD"));  
21         System.out.println(tmpStr.compareTo("abce"));  
22         System.out.println(tmpStr.compareTo("abcde"));  
23         System.out.println(tmpStr.compareTo("Abcd"));  
24     }  
25 }
```

```
2  
0  
0  
-1  
-1  
32
```

字串 String (Cont.)

- boolean contains(CharSequence s)
 - 傳回字串中是否存在 s 字串在其中，有 → 回傳 true，無 → 回傳 false
 - 什麼是 CharSequence?
 - 不是類別
 - 是一個介面 (Interface) (在 12 章中介紹)

```
15 public class StaticMethod{  
16     public static void main(String[] argv) throws IOException{  
17         String tmpStr = "CSIE, Asia University";  
18         System.out.println(tmpStr.contains("ASIA"));  
19         System.out.println(tmpStr.contains("Asia"));  
20         System.out.println(tmpStr.contains("versi"));  
21         System.out.println(tmpStr.contains("test"));  
22     }  
23 }
```

```
false  
true  
true  
false
```

字串 String (Cont.)

- boolean endsWith(String suffix)
 - 檢查是不是以 suffix 所指的字串做結尾，是 → 回傳 true，否則回傳 false

```
15 public class StaticMethod{  
16     public static void main(String[] argv) throws IOException{  
17         String tmpStr = "CSIE, Asia University";  
18         System.out.println(tmpStr.contains("ASIA"));  
19         System.out.println(tmpStr.contains("Asia"));  
20         System.out.println(tmpStr.endsWith("ity"));  
21         System.out.println(tmpStr.endsWith("sit"));  
22     }  
23 }
```

```
false  
true  
true  
false
```

字串 String (Cont.)

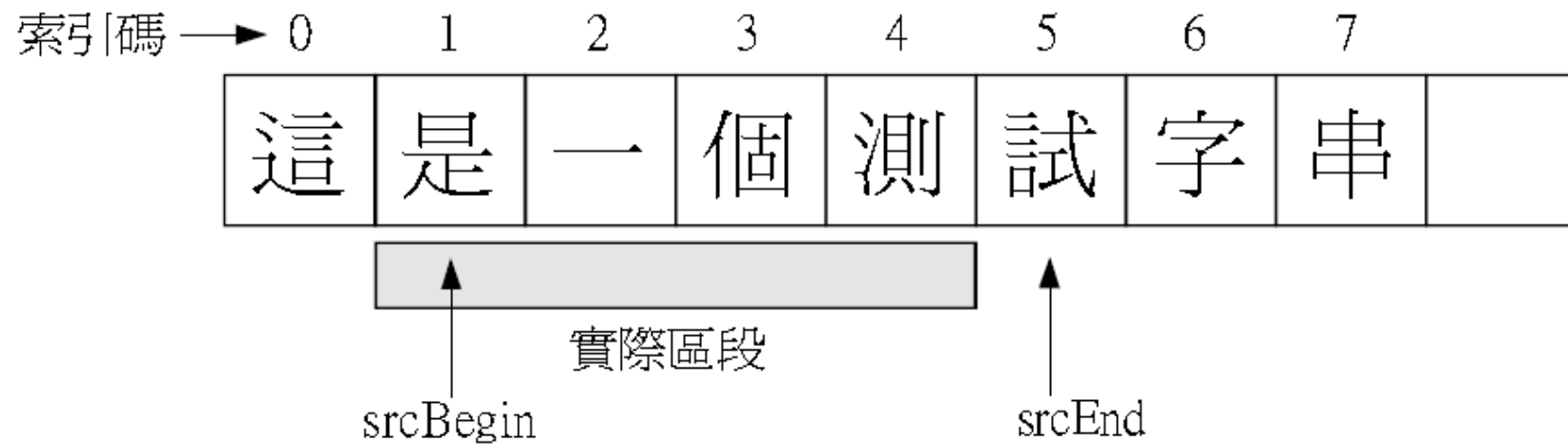
- void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
 - 將字串中索引碼為 srcBegin 的字元開始到索引碼為 srcEnd-1 的字元複製到 dst 所指的字元陣列中，並且從 dst 中的 dstBegin 索引位置開始放

```
15 public class StaticMethod{
16     public static void main(String[] argv) throws IOException{
17         String tmpStr = "CSIE, Asia University";
18         char[] tmpChar = {'I', 's', 't', 'u', 'd',
19                         'y', ' ', 'i', 'n', ' ', 'X', 'X',
20                         'X', 'X', ' ', 'U', 'n', 'i', 'v',
21                         'e', 'r', 's', 'i', 't', 'y', '.'};
22         System.out.println(tmpStr);
23         System.out.println(tmpChar);
24         tmpStr.getChars(6, 10, tmpChar, 10);
25         System.out.println(tmpChar);
26     }
27 }
```

```
CSIE, Asia University
Istudy in XXXX University.
Istudy in Asia University.
```

字串 String (Cont.)

- 區段表示法
 - 在 java 中，表示一個區段都是以開頭元素的索引碼及結束元素的下一個索引碼來表示



字串 String (Cont.)

- `int indexOf(int ch)`
- 傳回字元 `ch` 在字串中第一次出現的位置索引碼，如果字串中不含字元 `ch` 則回傳 `-1`
- 另一個雙胞胎版本 `int lastIndexOf(int ch)`: 從字串尾端找回來，第一次出現 `ch` 字元的位置索引碼

```
16 public class StaticMethod{
17     public static void main(String[] argv) throws IOException{
18         String tmpStr = "CSIE, Asia University";
19         System.out.println(tmpStr.indexOf('U'));
20         System.out.println(tmpStr.indexOf(65));
21         System.out.println(tmpStr.lastIndexOf('i'));
22         System.out.println(tmpStr.lastIndexOf(99));
23     }
24 }
```

```
<已終止>
11
6
18
-1|
```


字串 String (Cont.)

- `int length()`
 - 傳回字串的長度
- `String replace(char oldChar, char newChar)`
 - 將字串中的 `oldChar` 部份以 `newChar` 替換
 - 並不會改變原始字串的內容
 - 產生新的字串回傳

```
15 public class OOTest{  
16     public static void main(String[] argv){  
17         String str1 = "This is a book!";  
18         String str2 = str1.replace('i', 'K');  
19         System.out.println(str1);  
20         System.out.println(str2);  
21     }  
22 }
```

```
This is a book!  
ThKs Ks a book!
```

字串 String (Cont.)

- String replace(CharSequence target, CharSequence replacement)
 - 在字串中找出 target 的字元串，並以 replacement 字元串置換
 - 回傳結果是一個新的物件，而非修改原始字串資料

```
16 public class OOTest{
17     public static void main(String[] argv){
18         String str1 = "This is a book!";
19         String str2 = str1.replace("oo", "pq");
20         System.out.println(str1);
21         System.out.println(str2);
22         String str3 = str1.replace("oo", "p_q");
23         System.out.println(str3);
24     }
25 }
```

```
This is a book!
This is a bpqk!
This is a bp_qk!
```

字串 String (Cont.)

- boolean startsWith(String prefix, int offset)
 - 檢查自字串的 offset 位置起是不是由 prefix 字串開始
 - 如果是 → 回傳 true
 - 如果不是 → 回傳 false

```
16 public class OOTest{
17     public static void main(String[] argv){
18         String str1 = "This is a book!";
19         if(str1.startsWith("is", 3)){
20             System.out.println("^_^");
21         }else{
22             System.out.println("\\_/_");
23         }
24         if(str1.startsWith("is", 2)){
25             System.out.println("q_p");
26         }else{
27             System.out.println("-_-");
28         }
29     }
30 }
```

/
q_p

字串 String (Cont.)

- String substring(int beginIndex)
 - 取字串中第 beginIndex 字元到字串的最後一個字元
- String substring(int beginIndex, int endIndex)
 - 取字串中第 beginIndex 起的字元到第 endIndex 位置的字元為止，形成一個子字串回傳

```
16 public class OOTest{  
17     public static void main(String[] argv){  
18         String str1 = "This is a book!";  
19         String str2 = str1.substring(3);  
20         String str3 = str1.substring(3, 10);  
21         System.out.println("str1 -> "+str1);  
22         System.out.println("str2 -> "+str2);  
23         System.out.println("str3 -> "+str3);  
24     }  
25 }
```

```
str1 -> This is a book!  
str2 -> s is a book!  
str3 -> s is a
```

字串 String (Cont.)

- String toLowerCase()
 - 將字串裡的每一個字元轉換成小寫後回傳
 - 回傳另一個字串
- String toUpperCase();
 - 將字串裡的每一個字元轉換成大寫後回傳

```
16 public class OOTest{  
17     public static void main(String[] argv){  
18         String str1 = "Java is a famous programming LANGUAGE in the world.";  
19         System.out.println(str1+"\n"+str1.toLowerCase()+"\n"+str1.toUpperCase());  
20     }  
21 }
```

```
Java is a famous programming LANGUAGE in the world.  
java is a famous programming language in the world.  
JAVA IS A FAMOUS PROGRAMMING LANGUAGE IN THE WORLD.
```

字串 String (Cont.)

- String trim()
 - 將字串的頭、尾端的空白字元去除

```
16 public class OOTest{
17     public static void main(String[] argv){
18         String str1 = "    AAA  BBBB   XXX   ";
19         System.out.println("->|" + str1 + "|<-\\n->|" + str1.trim() + "|<-");
20     }
21 }
```

```
->|    AAA  BBBB   XXX   |<-
->| AAA  BBBB   XXX|<-
```

字串 String (Cont.)

- StringBuffer 與 StringBuilder 類別
 - 適用於直接更改字串的內容
 - 直接使用 String 類別，可不需因為字串長度改變而重新配置新的記憶體空間
 - 若需要隨時更改字串的內容，StringBuffer 與 StringBuilder 是個好選擇
- StringBuffer
 - 想成『可改變內容的 String 類別』
 - 提供了各種改變字串內容的方法

StringBuffer

- StringBuffer 提供的方法除了會修改字串的內容外，會傳回修改後的結果
- StringBuffer 所產生的物件不能使用『+』運算子進行字串的串接
- 在 StringBuffer 物件下，文字的串接必須要用 append() 或 insert() 方法

```
16 public class OOTest{
17     public static void main(String[] argv){
18         StringBuffer sb1 = new StringBuffer();
19         StringBuffer sb2 = new StringBuffer("Java language");
20         System.out.println(sb1+"\n"+sb2);
21     }
22 }
```

<圖形化> OOTest.java 圖

Java language

StringBuffer (Cont.)

- append()
 - 會在字串尾端添加資料
 - 具多重定義版本，可傳入各式各樣的基本資料型別
 - 各資料型別的 toString() 方法會將資料轉換成 String 並添加於字串尾端
 - 完成添加後，將自身字串回傳

```
16 public class OOTest{
17     public static void main(String[] argv){
18         StringBuffer sb1 = new StringBuffer("Asia University");
19         StringBuffer sb2 = new StringBuffer("Java language");
20         int key = 3;
21         final double PI = 3.14159D;
22         System.out.println(sb1+"\n"+sb2+"\n"+sb1.append(sb2));
23         System.out.println(sb1.append(key)+"\n"+sb1.append(PI));
24     }
25 }
```

Yung-Chen Chou

```
Asia University
Java language
Asia UniversityJava language
Asia UniversityJava language3
Asia UniversityJava language33.14159
```

StringBuffer (Cont.)

- insert()
 - 此方法與 append() 類似，但資料是加在第一個參數 offset 所給定的位置之前
 - 提供多重定義，傳入基本資料型別的資料會使用 toString 方法轉成字串後插入到字串中

```
16 public class OOTest{
17     public static void main(String[] argv){
18         StringBuffer sb1 = new StringBuffer("Asia University");
19         StringBuffer sb2 = new StringBuffer("Java language");
20         int key = 3;
21         final double PI = 3.14159D;
22         System.out.println(sb1+"\n"+sb2+"\n"+sb1.insert(5, sb2));
23         System.out.println(sb1.insert(3, key)+"\n"+sb1.insert(10, PI));
24     }
25 }
```

```
Asia University
Java language
Asia Java languageUniversity
Asi3a Java languageUniversity
Asi3a Java3.14159 languageUniversity
```

StringBuffer (Cont.)

- `StringBuffer deleteCharAt(int index)`
 - 把字串中第 `index` 位置的字元刪掉
- `StringBuffer replace(int start, int end, String str)`
 - 將字串中第 `start` 位置的字元至第 `end-1` 位置的字元以 `str` 字串取代

```
16 public class OOTest{
17     public static void main(String[] argv){
18         StringBuffer sb1 = new StringBuffer("西西沙巴西亞是洋基王牌投手");
19         System.out.println(sb1);
20         sb1.replace(0, 6, "王建民");
21         System.out.println(sb1);
22         sb1.insert(3, '才');
23         System.out.println(sb1);
24         sb1.delete(sb1.length()-2, sb1.length());
25         System.out.println(sb1);
26     }
27 }
```

西西沙巴西亞是洋基王牌投手
王建民是洋基王牌投手
王建民才是洋基王牌投手
王建民才是洋基王牌

StringBuffer (Cont.)

- StringBuffer reverse()
 - 將整個字串內容反轉
- void setCharAt(int index, char ch)
 - 將字串中 index 指到的位置的字元以 ch 字元取代
 - 唯一一個沒有回傳值的方法

```
16 public class OOTest{
17     public static void main(String[] argv){
18         String str = "花蓮噴水池";
19         StringBuffer sb2 = new StringBuffer(str);
20         System.out.println("str -> "+str
21                             +"\nsb2 -> "+sb2+"\nsb2.reverse() -> "
22                             +sb2.reverse());
23         sb2.setCharAt(0, '清');
24         System.out.println(sb2);
25     }
26 }
```

```
str -> 花蓮噴水池
sb2 -> 花蓮噴水池
sb2.reverse() -> 池水噴蓮花
清水噴蓮花
```

StringBuilder

- 與 stringBuffer 用途相同，且提供的方法一模一樣
- 唯一差別：此類別不保證可以在多執行緒的環境下正常運作
- StringBuilder 與 StringBuffer 的使用時機
 - 無多執行緒環境 → StringBuilder 具較高執行效率
 - 多執行緒環境 → StringBuffer 確定可以正常運作

規則表示法 (Regular Expression)

- 想限制使用輸入的格式、長度該如何做？
- 例如：輸入身份證字號、電話號碼、伊妹兒
- 為利後續資料的處理，當使用者輸入時即進行資料形式的比對
- 如果要寫一支程式比對使用者輸入的電子郵件帳號是否符合，該如何設計？
 - 檢查字串中是不是有 『 @ 』 符號，且確定其只出現一次
 - 檢查帳號是不是數字開頭
 - ...

規則表示法 (Regular Expression) (Cont.)

- String 類別提供 matches() 方法，可協助規則表示法來達到檢查使用者輸入的資料是不是符合系統要求
- 如何檢查使用者輸入的是不是整數？
 - 使用迴圈讀取字串的每一個字元並檢查
 - 檢查是不是存在小數

規則表示法 (Regular Expression) (Cont.)

```
17 public class OOTest{
18     public static void main(String[] argv) throws IOException{
19         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
20         String str;
21         boolean isInteger;
22         do{
23             isInteger = true;
24             System.out.print("請輸入一個整數 -> ");
25             str = br.readLine();
26             for(int i=0;i<str.length();i++){
27                 char ch = str.charAt(i);
28                 if((ch < '0') || (ch > '9')){
29                     System.out.println("您輸入的不是整數");
30                     isInteger = false;
31                     break;
32                 }
33             }
34         }while(!isInteger);
35     }
36 }
37 }
```

```
請輸入一個整數 -> 12.8
您輸入的不是整數
請輸入一個整數 -> 21
```


規則表示法 (Regular Expression)

(Cont.)

- 搭配 String 類別提供的 matches() 方法
 - 『 [0-9] 』 指的是包含數字 0-9
 - 『 + 』 表示可以出現一次以上
 - 意思：由一個或多個以上的數字所構成的字串

```
17 public class OOTest{
18     public static void main(String[] argv) throws IOException{
19         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
20         String str;
21         boolean isInteger;
22         do{
23             isInteger = true;
24             System.out.print("請輸入一個整數 -> ");
25             str = br.readLine();
26             if(!str.matches("[0-9]+")){
27                 System.out.println("您輸入的不是整數");
28                 isInteger = false;
29             }
30         }while(!isInteger);
31     }
32 }
33 }
```

```
請輸入一個整數 -> a1323
您輸入的不是整數
請輸入一個整數 -> 12.3
您輸入的不是整數
請輸入一個整數 -> 12
```

規則表示法 (Regular Expression) (Cont.)

- 當字串符合 `matches()` 方法中給定的樣式，則回傳 `true`，否則回傳 `false`
- 使用 `matches()` 方法的好處：可以專注於比對樣式，真正的比對動作由 `matches` 方法負責
- 比對數字
 - Unicode 中 0-9 的數字是連續的，因此只要比對字元的 `unicode` 是不是介於 0-9 之間即可
- 比對字串
 - 直接表示要比對的字串內容傳入 `matches()` 方法裡

規則表示法 (Regular Expression) (Cont.)

- 限制出現次數

- 常用的限制規則

- 例

- $ab?a \rightarrow$ 出現 1 個 a 後
再最多出現 1 個 b ，最
後再接 2 個 a

- $aa \rightarrow \text{true}$
 - $aba \rightarrow \text{true}$
 - $abba \rightarrow \text{false}$

限制規則	說明
$?$	最多 1 次
$*$	0 或是多次
$+$	至少 1 次
$\{n\}$	剛好 n 次
$\{n,\}$	最少 n 次
$\{n,m\}$	最少 n 次，但不超過 m 次

規則表示法 (Regular Expression) (Cont.)

- 字元種類
 - 使用中括號 [] 來限定你認可的字
 - 例 : A[bjk]a
 - Aba \rightarrow true
 - Aja \rightarrow true
 - Ara \rightarrow false
 - 例 : A[0-9a-zA-Z]a
 - Ara \rightarrow true
 - A%a \rightarrow false

規則表示法 (Regular Expression) (Cont.)

- 排除你所限定的字
 - 在中左括號後加 ^ 符號
 - 例 :A [^a-z]a
 - Aka \rightarrow false
 - Aaa \rightarrow false
 - A2a \rightarrow true
 - A#a \rightarrow true

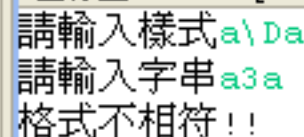
規則表示法 (Regular Expression) (Cont.)

- 預先定義字元種類

- 數字與英文字母的規則會用到，因此 Java 的規則表示法中提供預先定義的一些字元種類

字元種類	說明
.	任意字元
\d	數字
\D	非數字
\s	空白字元
\S	非空白字元
\w	英文字母或數字
\W	非英文字母也非數字

```
import java.io.*;
public class RETest{
    public static void main(String[] argv) throws IOException{
        BufferedReader br =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print("請輸入樣式");
        String patten = br.readLine();
        System.out.print("請輸入字串");
        String str = br.readLine();
        if(str.matches(patten)){
            System.out.println("格式相符!!");
        }else{
            System.out.println("格式不相符!!");
        }
    }
}
```



請輸入樣式 a\Da
請輸入字串 a3a
格式不相符!!

規則表示法 (Regular Expression) (Cont.)

- 群組 (Grouping)

- 使用括號將一段規則組合起來，搭配限制次數使用

```
import java.io.*;
public class RETest{
    public static void main(String[] argv) throws IOException{
        BufferedReader br =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print("請輸入樣式");
        String patten = br.readLine();
        System.out.print("請輸入字串");
        String str = br.readLine();
        if(str.matches(patten)){
            System.out.println("格式相符!!");
        }else{
            System.out.println("格式不相符!!");
        }
    }
}
```

```
請輸入樣式a(c\dc){2}a
請輸入字串ac1cc2ca
格式相符!!
```

```
請輸入樣式a(c\dc){2}a
請輸入字串ac1ca
格式不相符號!!
```

規則表示法 (Regular Expression) (Cont.)

- 以字面常數指定樣式
 - 在程式裡直接給定樣式
 - Java 編譯器將 `\` 視為跳脫序列的啟始字元，因此在程式裡使用字面常數指定樣式，則必須多加一個 `\` 字元

```
5 import java.io.*;
6 public class RETest{
7     public static void main(String[] argv)throws IOException{
8         BufferedReader br =
9             new BufferedReader(new InputStreamReader(System.in));
10        //System.out.print("請輸入樣式");
11        //String patten = br.readLine();
12        System.out.print("請輸入字串");
13        String str = br.readLine();
14        if(str.matches("a\\Dc")){
15            System.out.println("格式相符!!");
16        }else{
17            System.out.println("格式不相符!!");
18        }
19    }
20 }
```

請輸入字串 a@c
格式相符!!

```
import java.io.*;
public class RETest{
    public static void main(String[] argv)throws IOException{
        BufferedReader br =
            new BufferedReader(new InputStreamReader(System.in));
        System.out.print("請輸入樣式");
        String patten = br.readLine();
        System.out.print("請輸入字串");
        String str = br.readLine();
        if(str.matches("a\\Dc")){
            System.out.println("格式相符!!");
        }else{
            System.out.println("格式不相符!!");
        }
    }
}
```

Exception in thread "main" java.lang.Error: 尚未解決的編譯問題：
無效的 ESC 序列 (有效序列為 \b \t \n \f \r \" \' \\)

at RETest.main(RETest.java:14)

規則表示法 (Regular Expression) (Cont.)

- replaceAll() 方法

- 除了比對字串外，也可以用來將字串中符合指定格式的一段文字取代成另一段文字

```
5 import java.io.*;
6 public class RCTest{
7     public static void main(String[] argv) throws IOException{
8         BufferedReader br =
9             new BufferedReader(new InputStreamReader(System.in));
10        System.out.print("請輸入樣式");
11        String pat = br.readLine();
12        System.out.print("請輸入字串");
13        String str = br.readLine();
14        System.out.print("請輸入要取代成 -> ");
15        String rep = br.readLine();
16        System.out.println(str.replaceAll(pat, rep));
17    }
18 }
```

```
請輸入樣式111
請輸入字串a111bc34d
請輸入要取代成 -> 三個一
a三個一bc34d
```

規則表示法 (Regular Expression) (Cont.)

- 在某些特殊的場合會希望取代後又能保留原始字串資料，此時使用群組會是一個不錯的作法
- 其中取代成『數字 \$1』中的『\$1』意指比對相符的那段文字中與樣式中第一個群組相符的部份

```
請輸入樣式 (\d+)
請輸入字串 a111bc34d
請輸入要取代成 -> 數字$1
a數字111bc數字34d
```

- 以此類推，\$2, \$3 自然是指第 2、3、... 等個群組，而 \$0 指的是比對整段文字

```
請輸入字串 -> a111bc34d
請輸入樣式 -> ([a-z])(\d+)([a-z]+)(\d+)([a-z])
請輸入要取代成 -> 1:$1,2:$2,3:$3,4:$4,0:$0
1:a,2:111,3:bc,4:34,0:a111bc34d
```