第 10 章

字串（String）

10-1 字串的產生

- 字串就是 String 物件, 所以宣告一個字串變數, 等於宣告一個指到 String 物件的參照, 然後再產生 String 物件。

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

字串的產生

程式 ConstructString.java 利用建構方法建立字串

```
01 public class ConstructString {
02
03   public static void main(String[] argv) {
04
05     char[] test = {'這','是','個','測','試','字','串'};
06     String a = new String();              // ""
07     String b = new String(test);          // "這是個測試字串"
08     String c = new String(test,3,4);      // "測試字串"
09     String d = new String(b);             // "這是個測試字串"
10
```

字串的產生

```
11      System.out.println("a : " + a);
12      System.out.println("b : " + b);
13      System.out.println("c : " + c);
14      System.out.println("d : " + d);
15
16      // d 是 b 的副本
17      System.out.println("b == d ?" + (b == d));
18    }
19  }
```

執行結果

```
a :
b : 這是個測試字串
c : 測試字串
d : 這是個測試字串
b == d ?false
```

10-1-1 Java 對於 String類別的特別支援

- 使用字面常數建立 String 物件

程式 StringConstant.java 使用字面常數建立字串

```
01 public class StringConstant {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     String b = "這是一個測試字串";
06     String c = new String("這是一個測試字串");
07
08     System.out.println("a == b ?" + (a == b));
09     System.out.println("b == c ?" + (b == c));
10     System.out.println("a == c ?" + (a == c));
11   }
12 }
```

使用字面常數建立 String 物件

- 可以把這 4, 5, 6 行看成是這樣：

```
String constant1 = "這是一個測試字串";
String a = constant1;
String b = constant1;
String c = new String(constant1);
```

使用字面常數建立 String 物件

- 如果要比對字串的內容, 就必須使用 String 類別的equals() 方法

連接運算

- 當運算元中有字串資料時,
  "+" 算符就會進行連接字串的動作。

10-1-2 String 物件的特性

- 自動轉型 (Implicit Conversion)

程式 Conversion.java Java 將物件自動轉型為 String

```
01 class Student {
02   String name;
03   public Student(String s) { name = s; }
04   public String toString() { return name; }
05 }
06
07 public class Conversion {
08   public static void main(String[] argv) {
09     Student a = new Student("Joy");
10     System.out.println("I am " + a); // 將會呼叫 a.toString()
11   }
12 }
```

執行結果

```
I am Joy
```

String 物件的內容無法更改

- String 物件一旦產生之後，其內容就無法更改
- 連接運算是以運算元連接之後的字串產生新的 String 物件作為運算結果。

10-2 String 類別的方法

- String 類別提供許多處理字串的方法，可以有效的幫助使用字串。
- 傳回值型別為 String 的方法都是『傳回新字串』，而不會修改原本的字串內容。

char charAt(int index)

- 傳回 index 所指定索引碼的字元
- 索引碼是從 0 開始算起

程式 CharAt.java 使用 charAt() 方法

```
01 public class CharAt {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05
06     System.out.println("索引 0 的字元：" + a.charAt(0));
07     System.out.println("索引 5 的字元：" + a.charAt(5));
08   }
09 }
```

執行結果

```
索引0的字元：這
索引5的字元：試
```

int compareTo (String anotherString)

- 以逐字元方式與 anotherString 字串比較
  - anotherString 比較大就傳回一個負數值
  - 字串內容完全相同, 就傳回 0
  - another-String 比較小, 就傳回一個正數值

int compareTo (String anotherString)

兩個字串 a,b 之間的大小是照以下規則決定：

1. 由索引 0 開始, 針對 a 與 b 相同索引碼的字元逐一比較其標準萬國碼 (Unicode), 一旦遇到相同位置但字元不同時, 就以此位置的字元決定 a 與 b 的大小。

int compareTo (String anotherString)

2. 如果 a 與 b 的長度相同，且逐一字元比較後，同位置的字元皆相同，就傳回 0。此時, a.equals(b) 或是 b.equals(a) 皆為 true。
3. 如果 a 與 b 長度不同，且逐一字元比較後，較短的一方完全和較長的一方前面部分相同, 就以較長的為大。

int compareTo (String anotherString)

程式 CompareTo.java 使用 compareTo() 方法

```
01 public class CompareTo {
02
03   public static void main(String[] argv) {
04     String a = "abcd";
05     System.out.println(a.compareTo("abcb"));
06     System.out.println(a.compareTo("abcd"));
07     System.out.println(a.compareTo("abce"));
08     System.out.println(a.compareTo("abcde"));
09     System.out.println(a.compareTo("Abcd"));
10   }
11 }
```

執行結果

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

int compareTo (String anotherString)

程式 CompareToIgnoreCase.java 將大小寫視為相同來比較字串

```
01 public class CompareToIgnoreCase {
02
03   public static void main(String[] argv) {
04     String a = "abcd";
05     System.out.println(a.compareToIgnoreCase("ABCB"));
06     System.out.println(a.compareToIgnoreCase("ABCD"));
07     System.out.println(a.compareToIgnoreCase("ABCE"));
08   }
09 }
```

執行結果

```
2
0
-1
```

boolean contains(CharSequence s)

- 傳回字串中是否包含有 s 所指字串的內容

程式 Contains.java 使用 contains() 方法

```
01 public class Contains {
02
03   public static void main(String[] argv) {
04     String a = "abcd";
05     System.out.println(a.contains("abcd"));
06     System.out.println(a.contains("abc"));
07     System.out.println(a.contains("abcde"));
08     System.out.println(a.contains("lkk"));
09   }
10 }
```

執行結果

```
true
true
false
false
```

boolean endsWith(String suffix)

- 傳回是否以指定的字串內容結尾。

程式 EndsWith.java 使用 endsWith() 方法

```
01 public class EndsWith {
02
03   public static void main(String[] argv) {
04     String a = "abcd";
05     System.out.println(a.endsWith("cd"));
06   }
07 }
```

執行結果

```
true
```

void getChars(int srcBegin, int srcEnd, char[ ] dst, int dstBegin)

- 將索引碼 srcBegin 到 srcEnd - 1 的字元, 複製到指定的字元陣列和索引位置

程式 GetChars.java 使用 getChars() 方法

```
01 public class GetChars {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     char[] chars = new char[4];
06     a.getChars(1,5,chars,0);
07     System.out.println(new String(chars));
08   }
09 }
```

執行結果

```
是一個測
```

int indexOf(int ch)

- 傳回 ch 字元在字串中第一次出現的索引碼

程式 IndexOf.java 使用 indexOf() 方法

```
01 public class IndexOf {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     System.out.println(a.indexOf('測'));
06     System.out.println(a.indexOf('空'));
07   }
08 }
```

執行結果

```
4
-1
```

int indexOf(int ch, int fromIndex)

- indexOf() 方法的多重定義版本, 可以用 fromIndex 來指定開始尋找的位置。

- 這個方法也有個對應的 lastIndexOf (int ch, int fromIndex)

int indexOf(String str)

- indexOf() 的多重定義版本, 尋找指定字串出現的位置。

程式 IndexOfString.java 使用 indexOf() 方法

```
01 public class IndexOfString {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     System.out.println(a.indexOf("測試"));
06     System.out.println(a.indexOf("字符"));
07   }
08 }
```

執行結果

```
4
-1
```

int indexOf(String str, int fromIndex)

- indexOf() 方法的多重定義版本, 可以用 fromIndex 來指定開始尋找字串的位置。

- 當然也有對應的 lastIndexOf(String str, int fromIndex)

boolean isEmpty()

- 判斷是否為空字串 (字串長度為 0),
  是空字串就傳回 true, 否則傳回 false。

boolean isBlank()

- 這是從 Java 11 開始才有的方法, 可判斷是否為空字串或字串中只包含空白符號
- 這裡的空白符號是指空白、定位、換行、換頁等字元, 以及 Unicode 空白字元 。

int length()

- 傳回字串的長度。
  例如 isEmpty() 傳回 true 時, 呼叫 length() 就會傳回 0。

String replace (char oldChar, char newChar)

- 將字串中所有 oldChar 字元取代為 newChar 字元。

程式 Replace.java 使用 replace() 方法

```
01 public class Replace {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     System.out.println(a.replace('測','考'));
06     System.out.println(a);
07   }
08 }
```

執行結果

```
這是一個考試字串
這是一個測試字串
```

String replace (CharSequence target, CharSequence replacement)

- 和上一個方法功能類似,將字串中所有target 字串都取代為 replace-ment 字串。

程式 ReplaceStr.java 使用 replace() 方法

```
01 public class ReplaceStr {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     System.out.println(a.replace("測試","正式"));
06     System.out.println(a);
07   }
08 }
```

執行結果

```
這是一個正式字串
這是一個測試字串
```

boolean startsWith(String prefix)
boolean startsWith(String prefix, int toffset)

- startsWith() 的用法和前面看過的 endsWith() 類似, 但功能相反
- startsWith() 是用來檢查目前字串是否是以參數字串 prefix 開頭。

boolean startsWith(String prefix)
boolean startsWith(String prefix, int toffset)

程式 CheckStarts.java 檢查字串開頭

```
01 public class CheckStarts {
02
03   public static void main(String[] argv) {
04     String a = "abcd";
05     System.out.println(a + " 的開頭是 cd：" +
06                        a.startsWith("cd"));
07     System.out.println(a + " 從第 3 個字開始算的開頭是 cd：" +
08                        a.startsWith("cd",2));
09   }
10 }
```

執行結果

```
abcd 的開頭是 cd：false
abcd 從第 3 個字開始算的開頭是 cd：true
```

String substring(int beginIndex)

- 傳回由 beginIndex 索引開始到結尾的部分字串。

String substring(int beginIndex, int endIndex)

- 傳回由 beginIndex 到 endIndex - 1 索引的部分字串。

程式 Substring.java 使用 substring() 方法

```
01 public class Substring {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     System.out.println(a.substring(4));
06     System.out.println(a.substring(4,6));
07   }
08 }
```

執行結果

```
測試字串
測試
```

String toLowerCase()

- 傳回將字串中的字元全部轉成小寫後的副本。

String toUpperCase()

- 傳回將字串中的字元全部轉為大寫後的副本。

String trim()

- 將字串中頭、尾端的空白符號去除，包含空白、定位、換行、換頁等字元。

程式 Trim.java 使用 trim 方法

```
01 public class Trim {
02
03   public static void main(String[] argv) {
04     String a = " 這是一個測試字串\t";
05     System.out.print(a.trim());
06     System.out.println("...定位字元不見了");
07     System.out.print(a);
08     System.out.println("...定位字元還在");
09   }
10 }
```

執行結果

```
這是一個測試字串...定位字元不見了
這是一個測試字串    ...定位字元還在
```

String strip()
String stripLeading()
String stripTrailing()

- 這是從 Java 11 開始才有的方法, 可分別去除字串中頭尾、頭端、尾端的空白符號。
- 前面介紹的 trim() 也可去除字串頭尾的空白符號, 但不包括 Unicode 空白字元。

String repeat(int count)

- 這也是從 Java 11 開始才有的方法,
  可傳回將字串內容重複 count 次的字串。

10-3 StringBuffer 與 StringBuilder 類別

- 如果想使用可以隨時更改內容的字串，那麼就必須改用 StringBuffer 或 StringBuilder 類別。

10-3-1 StringBuffer 類別

- 『可改變內容的 String 類別』。

建構方法	說明
StringBuffer()	建立一個不含任何字元的字串
StringBuffer(String str)	依據 str 的內容建立字串

StringBuf fer 類別

程式 StrBuf.java 建立 StringBuffer 物件

```
01 public class StrBuf {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     StringBuffer b = new StringBuffer(a);
06     System.out.println(b);
07   }
08 }
```

執行結果

```
這是一個測試字串
```

StringBuffer 類別

- Java 會產生一個 String 物件來代替程式中的字面常數,所以第 4、5 行也可寫成：

```
StringBuffer b = new StringBuffer("這是一個測試字串");
```

append() 方法

- 在字串尾端添加資料，
  並且擁有多重定義的版本，
  可以傳入基本型別、String 物件以及其他
  有定義 toString()方法的物件。

append() 方法

程式 Append.java 使用 append() 方法

```
01 public class Append {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     StringBuffer b = new StringBuffer(a);
06     System.out.println  (b.append(20)); // 會更改字串
07     System.out.println  (b.append("字串內容已經變了"));
08     System.out.println  (b.append(b));
09   }
10 }
```

執行結果

```
這是一個測試字串20
這是一個測試字串20字串內容已經變了
這是一個測試字串20字串內容已經變了這是一個測試字串20字串內容已經變了
```

insert() 方法

- 它可以透過第 1 個參數 offset 將第 2 個參數插入到字串中的特定位置。

程式 Insert.java 使用 insert() 方法

```
01 public class Insert {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     StringBuffer b = new StringBuffer(a);
06
07     System.out.println(b.insert(0,20)); // 插入到最開頭
08     System.out.println(b.insert(3,"字串內容已經變了"));
09
10     // 插入到尾端，等於append
11     System.out.println(b.insert(b.length(),b));
12   }
13     }
```

insert() 方法

執行結果

```
20這是一個測試字串
20這字串內容已經變了是一個測試字串
20這字串內容已經變了是一個測試字串20這字串內容已經變了是一個測試字串
```

StringBuffer delete (int start, int end)

- delete() 方法可以刪除由 start 索引碼到 end - 1 索引碼之間的一段字元。

程式 Delete.java 使用 delete() 方法

```
01 public class Delete {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05       StringBuffer b = new StringBuffer(a);
06
07       System.out.println(b.delete(1,2)); // 刪除1個字元
08       System.out.println(b.delete(0,3)); // 刪除3個字元
09   }
10 }
```

執行結果

```
這一個測試字串
測試字串
```

StringBuffer deleteCharAt(int index)

- 刪除由 index 所指定索引碼的字元。

StringBuffer replace (int start, int end, String str)

- 將 start 索引碼到 end - 1 索引碼之間的一段字元取代為 str 字串。

程式 ReplaceSubstring.java 使用 replace() 方法

```
01 public class ReplaceSubstring {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     StringBuffer b = new StringBuffer(a);
06
07     // 刪除1個字元
08     System.out.println(b.deleteCharAt(2));
09     // 取代2個字元
10     System.out.println(b.replace(1,3,"好像不是"));
11   }
12 }
```

執行結果

```
這是個測試字串
這好像不是測試字串
```

StringBuffer reverse()

- 將整個字串的內容頭尾反轉。

程式 Reverse.java 使用 reverse() 方法

```
01 public class Reverse {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     StringBuffer b = new StringBuffer(a);
06
07     System.out.println(b.reverse());
08     System.out.println(b.reverse());
09   }
10 }
```

執行結果

```
串字試測個一是這
這是一個測試字串
```

轉兩次就恢復原狀

void setCharAt(int index, char ch)

- 將 index 索引碼的字元取代成 ch 字元。這是唯一一個更改了字串內容,但卻沒有傳回自己的方法,在使用時要小心。

程式 SetCharAt.java 使用 setCharAt() 方法

```
01 public class SetCharAt {
02
03   public static void main(String[] argv) {
04     String a = "這是一個測試字串";
05     StringBuffer b = new StringBuffer(a);
06
07     b.setCharAt(2,'二');
08     System.out.println(b); // 字串內容已經變了
09   }
10 }
```

執行結果

```
這是二個測試字串
```

10-3-2 StringBuilder 類別

- 這個類別和 StringBuffer 的用途相同，唯一的差別就是此類別並不保證在多執行緒的環境下可以正常運作。

10 - 4 規則表示法 (Regular Expression)

- String 類別的 matches() 方法,
  可以描述字串內容規則的方式, 然後依據此一規則來驗證字串的內容是否相符。

10-4-1 甚麼是規則表示法

- 用來描述字串樣式的規則

10-4-1 甚麼是規則表示法

程式 CheckInteger.java 檢查輸入資料是否為一整數

```
01 import java.io.*;
02
03 public class CheckInteger {
04
05   public static void main(String[] argv) throws IOException {
06     BufferedReader br =
07       new BufferedReader(new InputStreamReader(System.in));
08
09     String str; // 記錄使用者輸入資料
10     boolean isInteger; // 使用者輸入是否為整數
11     do {
12       isInteger = true;
13       System.out.print("請輸入整數：");
14       str = br.readLine(); // 讀取使用者輸入資料
```

甚麼是規則表示法

```
15
16        for(int i = 0;i < str.length();i++) {
17          char ch = str.charAt(i); // 取出個別字元
18          if(ch < '0' || ch > '9') { // 不是數字
19            System.out.println("您輸入的不是整數！");
20            isInteger = false;
21            break; // 已檢查出非數字，不需繼續
22          }
23        }
24      } while (!isInteger);
25    }
26  }
```

執行結果

```
請輸入整數：123D
您輸入的不是整數！
請輸入整數：A12
您輸入的不是整數！
請輸入整數：1234
```

甚麼是規則表示法

程式 CheckIntegerByRegEx.java

```
11      do {
12        isInteger = true;
13        System.out.print("請輸入整數：");
14        str = br.readLine(); // 讀取使用者輸入資料
15
16        if(!str.matches("[0-9]+")) { // 如果不是整數
17          System.out.println("您輸入的不是整數！");
18          isInteger = false;
19        }
20      } while (!isInteger);
21    }
22  }
```

10-4-2 規則表示法入門

程式 RegExTest.java 規則表示法的練習程式

```
01 import java.util.*;
02
03 public class RegExTest {
04
05   public static void main(String[] argv)   {
06     Scanner sc = new Scanner(System.in);
07
08     String pat; // 記錄使用者輸入樣式
09     String str; // 記錄使用者輸入測試字串
```

規則表示法入門

```
10
11      System.out.print("請輸入樣式：");
12      pat = sc.next(); // 讀取樣式
13
14      System.out.print("請輸入字串：");
15      str = sc.next(); // 讀取字串
16
17      if(str.matches(pat))     // 進行比對
18        System.out.println("相符");
19      else
20        System.out.println("不相符");
21    }
22 }
```

直接比對字串內容

執行結果

```
請輸入樣式：print
請輸入字串：print
相符

請輸入樣式：print
請輸入字串：Print
不相符
```

限制出現次數

限制規則	說明
?	0 或 1 次
*	0 次以上 (任意次數)
+	1 次以上
{n}	剛好 n 次
{n,}	n 次以上
{n,m}	n 到 m 次

執行結果

```
請輸入樣式：ab?a
請輸入字串：aa
相符

請輸入樣式：ab?a
請輸入字串：aba
相符

請輸入樣式：ab?a
請輸入字串：abba
不相符
```

字元種類 (Character Classes)

執行結果

```
請輸入樣式：a[bjl]a
請輸入字串：aba
相符

請輸入樣式：a[bjl]a
請輸入字串：aka
不相符
```

字元種類 (Character Classes)

執行結果

```
請輸入樣式：a[0-9a-zA-Z]a
請輸入字串：a1a
相符

請輸入樣式：a[0-9a-zA-Z]a
請輸入字串：a#a
不相符
```

字元種類 (Character Classes)

執行結果

```
請輸入樣式：a[^a-z]a
請輸入字串：ada
不相符

請輸入樣式：a[^a-z]a
請輸入字串：a2a
相符
```

預先定義的字元種類 (Character Class)

執行結果

```
請輸入樣式：a\da
請輸入字串：a3a
相符

請輸入樣式：a\da
請輸入字串：aba
不相符
```

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

群組 (Grouping)

執行結果

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

請輸入樣式：a(c\dc){2}a
請輸入字串：ac1ca
不相符
```

10-4-3 replaceAll() 方法

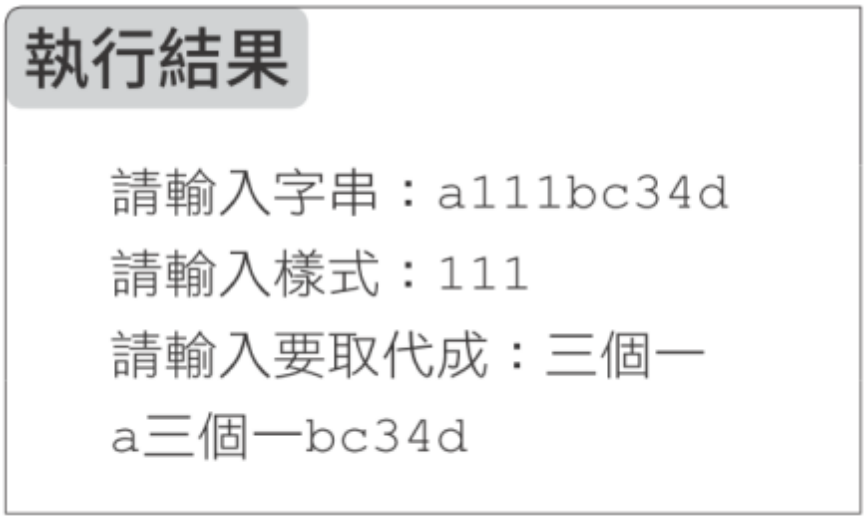
程式 ReplaceAll.java 測試 replaceAll() 方法

```
01 import java.io.*;
02
03 public class ReplaceAll {
04
05   public static void main(String[] argv) throws IOException {
06     BufferedReader br =
07       new BufferedReader(new InputStreamReader(System.in));
08
09     String src; // 記錄使用者輸入資料
10     String pat; // 記錄樣式
11     String rep; // 記錄要取代的結果
12
```

replaceAll() 方法


```
13      System.out.print("請輸入字串：");
14      src = br.readLine(); // 讀取使用者輸入字串
15
16      System.out.print("請輸入樣式：");
17      pat = br.readLine(); // 讀取使用者輸入樣式
18
19      System.out.print("請輸入要取代成：");
20      rep = br.readLine(); // 讀取使用者輸入字串
21
22      System.out.println(src.replaceAll(pat,rep));
23    }
24  }
```

簡單取代



使用樣式進行取代

執行結果

```
請輸入字串：a111bc34d
請輸入樣式：\d+
請輸入要取代成：數字
a數字bc數字d
```

使用群組

執行結果

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

使用群組

執行結果

```
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
```

10-5 綜合演練

- 10-5-1 檢查身份證字號的格式

程式 CheckIDFormat.java 檢查身份證字號的格式

```
01 import java.io.*;
02
03 public class CheckIDFormat {
04
05   public static void main(String[] argv) throws IOException {
06     BufferedReader br =
07       new BufferedReader(new InputStreamReader(System.in));
08
09    String str; // 記錄使用者輸入資料
10    boolean isID; // 使用者輸入的格式是否正確
11    do {
12      isID = true;
13      System.out.print("請輸入身份證字號：");
14      str = br.readLine(); // 讀取使用者輸入資料
15
```

檢查身份證字號的格式

```
16        if(!str.matches("[a-zA-Z]\\d{9}")) { // 如果不正確
17          System.out.println(
18            "身份證字號應該是1個英文字母接著9個數字！");
19          isID = false;
20        }
21      } while (!isID);
22    }
23  }
```

執行結果

```
請輸入身份證字號：aa45366
身份證字號應該是1個英文字母接著9個數字！
請輸入身份證字號：a1234567890
身份證字號應該是1個英文字母接著9個數字！
請輸入身份證字號：a123456789
```

10- 5 - 2 檢核身份證字號

檢核的規則如下：

1. 首先將第一個字母依據下表取代成 2 個數字：

A	10	B	11	C	12	D	13	E	14
F	15	G	16	H	17	I	34	J	18
K	19	L	20	M	21	N	22	O	35
P	23	Q	24	R	25	S	26	T	27
U	28	V	29	W	32	X	30	Y	31
Z	33								

10-5-2 檢核身份證字號

2. 將第一個數字乘以 1, 再從第 2 個數字開始, 第 2 個數字乘以 9、第 3 個數字乘以 8、...、第 9 個數字乘以 2、第 10 個數字乘以 1, 將這些乘法的結果相加總。
3. 以 10 減去加總值的個位數。
4. 如果上述減法的結果個位數和第 11 個數字相同, 此身份證字號即為合法, 否則即為不合法的身份證字號。

檢核身份證字號

程式 CheckID.java 檢查身份證字號的合法性

```
01 import java.io.*;
02
03 public class CheckID {
04
05   public static void main(String[] argv) throws IOException {
06     BufferedReader br =
07       new BufferedReader(new InputStreamReader(System.in));
08
09    String str; // 記錄使用者輸入資料
10    boolean isID; // 使用者輸入的格式是否正確
11    do {
12      isID = true;
13      System.out.print("請輸入身份證字號：");
14      str = br.readLine(); // 讀取使用者輸入資料
```

檢核身份證字號

```
15
16        if(!str.matches("[a-zA-Z]\\d{9}")) { // 不正確
17          System.out.println(
18            "身份證字號應該是1個英文字母接著9個數字！");
19          isID = false;
20        }
21      } while (!isID);
22
23      int[] letterNums = {10,11,12,13,14,15,16,
24        17,34,18,19,20,21,22,
25        35,23,24,25,26,27,28,
26        29,32,30,31,33};
27
28      str = str.toUpperCase(); // 先將第一個英文字母轉為大寫
29      char letter = str.charAt(0); // 取出第一個字母
```

檢核身份證字號


```
30      // 將第一個字母查表後取代成數字
31      str = letterNums[letter - 'A'] + str.substring(1);
32
33      int total = str.charAt(0) - '0'; // 開始加總
34      for(int i = 1;i < 10;i++) {
35        total += (str.charAt(i) - '0') * (10 - i); // 依序加總
36      }
37
38      // 以10減去加總值之個位數後取個位數
39      int checkNum = (10 - total % 10) % 10;
40
41      //計算結果和最後一位數比較
42      if(checkNum == (str.charAt(10) - '0')) {
43        System.out.println("檢核通過");
```

檢核身份證字號

```
44        } else {
45            System.out.println("檢核錯誤，請確實填寫");
46        }
47    }
48 }
```

執行結果

```
請輸入身份證字號：Z123456780
檢核通過

請輸入身份證字號：k223405678
檢核錯誤，請確實填寫
```