第六章 方法與套件引用

6-1 類別與方法呼叫

6-1-1 程式最小單位 - class

『類別』(Class) 是 Java 最的程式單元，是編譯成**中間碼 (bytecode)** 的基本單位。一套由 Java 所發展而成的軟體，可能包含若干個類別程式，每一類別實現某一特殊功能。從軟體工程來講，可以分門別類的將各種功能程式建構獨立的類別。當我們需要某一專案系統時，再由眾多類別中挑選某些類別組合而成，這就是軟體零件的基本概念。然而一個類別程式也可能非常龐大，需要由多個『方法』(Method) 所構成。類別程式包含有兩種成員：

(1) 變數成員：又稱為類別變數，存活於該類別內。

(2) 方法成員：實現該類別功能的程式。

最簡單的 Java 程式可以是單單只有一個主類別構成的程式，其基本架構如圖 6-1 所示。

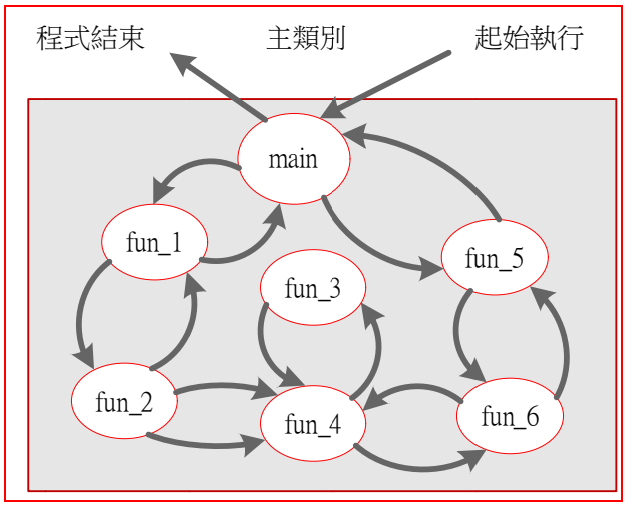

圖 6-1 主類別與方法架構

6-1-2 Java 程式專案的架構

吾人利用 Java 來發展軟體，該專案內的架構可歸納出下列重點：(如圖 6-2 所示)

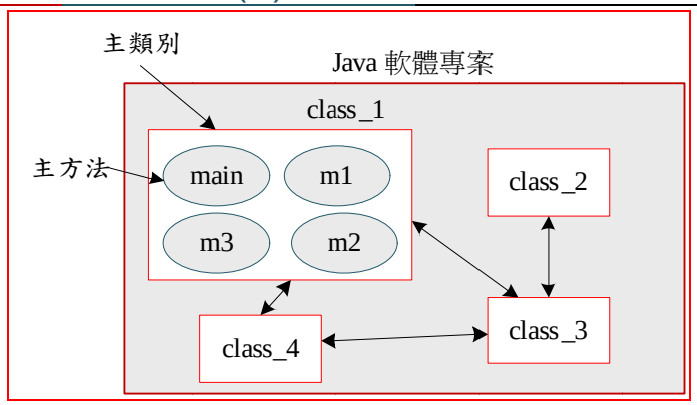

圖 6-2 Java 軟體專案

(1) 一套 Java 軟體可能包含**若干個類別**程式。

(2) 一個類別程式可能由**多個方法程式**（或稱方法成員）與**若干個變數**成員所構成。

(3) 方法名稱為 main 者，稱之為**『主方法』(main method)**，它是啟動 Java 軟體，第一個執行的程式。

(4) 具有 main 方法的類別程式，稱之為**『主類別』(main class)**；它的名稱必須與檔案名稱相同，也是第一個被導入的中間碼。

(5) 當由主類別啟動**主方法（main）**之後，主方法可直接呼叫同類別內的方法；相同的，主類別內的其他方法也可以互相呼叫；此運作程序稱之為**『函數呼叫』(Function call)**。

(6) 如果程式正常運作的話，主方法應該是最後結束的執行函數。也就是說，Java 程式由 **main 函數開始執行，也是由他最後結束的**。

(7) 其他函數也如同 main 函數一樣，被呼叫執行完成之後，一定會返回程式呼叫的原來地方。

在傳統程式設計的理念上，為了簡化程式開發的複雜性，會將一套系統分割成若干個子系統來實現，然後針對子系統內某些特定功能，獨立開發成『函數』程式。如此一來，某一套系統也許會被分割成多個程式模組，每一模組再由若干的程式『函數』所構成。軟體發展步驟是，首先分別獨立實現各個函數程式，再組合各個程式模組，最後再將多個模組整合成一套軟體系統，這種製作理念又稱為『結構化程式設計』。

Java 不再延用結構化設計理念，而是以物件導向的設計理念。雖然程式模組或函數都能表現系統中某一種特殊功能，但他們之間是獨立的；實現每一只函數或模組，都必須從頭開始，很難相互運用。由許多發展軟體的經驗得知，當您發展某一應用系統時，這些程式模組大多是大同小異

的。因此，我們就思考如何設計一套可以自由伸展的『程式因子』(prime)，再由小的程式因子擴充成較大的因子，或組合較若干個因子成為另一功能的程式因子，這個程式因子即是『物件導向』的理念。實現程式因子的最小單元就是『類別』，類別程式內也可能包含多個『方法』; 僅以傳統語言的設計理念，我們暫時將『方法』視為『函數』的功能。

6-2 方法宣告與流程

6-2-1 方法的宣告

在某一程式內，我們大多會將些特殊功能並會重複使用的程式，編寫成一個方法 (或稱為函數)，讓它可以被重複呼叫使用。另一方面，為了實現方法特殊功能 (如，加法器)，它被呼叫時允許呼叫者植入相關『引數』(如，加數與被加數); 執行完後，也會回覆執行結果 (如，兩數的和)。為了達到上述功能，宣告方法的語法如下：

宣告方法 (函數) 語法：	範　　例

方法屬性 傳回值型態 方法名稱 (引數_1, 引數_1) {
　　方法實體 …
　　……
　　return 傳回值;
}
	static int ADD (int a, int b) {
	　　int c;
	　　c = a + b;
	　　return c;
	}

呼叫方法	範　　例
資料型態 變數名稱=方法名稱(引數, ..);	int sum = ADD(3, 5);

相關參數如下：

(1) 方法屬性：可被引用 (呼叫) 的屬性，常用有下列組合型態：

(a) public static：允許類別外部呼叫的靜態 (或稱類別) 方法。

(b) private static：僅允許類別內其他方法呼叫的靜態方法。

(c) static：如同 public static 屬性。

(2) 傳回值型態：宣告執行該方法後，得到結果的資料型態，可宣告為一般資料型態(int、float、char...等)、或某一物件型態 (object)，如果不回傳任何值，則宣告為 void (空)。

(3) 方法名稱：指定該方法（或函數）名稱，與一般變數名稱相同。

(4) 引數：呼叫該函數時，可以攜帶的引數，可以是變數、或數值。

(5) { ……}：函數的程式主體。

(6) return 變數（或敘述句）：執行後將變數（或敘述句）內容回傳，該資料型態必須與『傳回值型態』相同。

譬如，**『主方法』（或稱主函數）**是不需經由產生物件再執行，而可以直接執行的（static 屬性），並允許外部環境呼叫它（public 屬性），執行後也不會傳回任何資料（void 傳回值型態）；因此，宣告主方法的相關參數為 public static void main(..)。

宣告方法 (函數) 範例：

public static void main(String args[]){
……
……　　// “空” (void) 傳回值，不需要 return
}

6-2-2 範例研討：加法器函數製作

（A）系統功能：Ex6_1.java

請製作一個兩數相加的加法器函數，於主程式會要求輸入兩個整數，再呼叫加法器球兩整數的和，最後印出兩數相加的結果。期望操作介面如下：

```
請輸入第一個整數 =>35
請輸入第二個整數 =>48
35 + 48 = 83
```

（B）製作技巧研討：

題目要求製作一個加法器（Add()），並將主程式（main()）所輸入的兩個整數相加後，顯示其結果。由此可見，主類別中包含兩個函數（或稱方法，method），如圖 6-3 所示。Main 方法利用 sum = Add(value1, value2) 敘述句呼叫 Add 方法，並且攜帶了兩個引數。依照引數的相對應位置，將 value1 內容填入 a 變數內；也將 value2 內容複製到 b 變數內。Add 函數將執行結果 sum，利用 return 敘述句，傳回給 main 方法，並存入 main 方法內的 sum 變數內。另外，main 與 Add

方法內都有 sum 變數，但兩者皆是獨立不相干的『區域變數』。

```
1  static int ADD (int a, int b) {
2        int c;
3        c = a + b;
4        return c;
5  }
```

圖 6-3-1

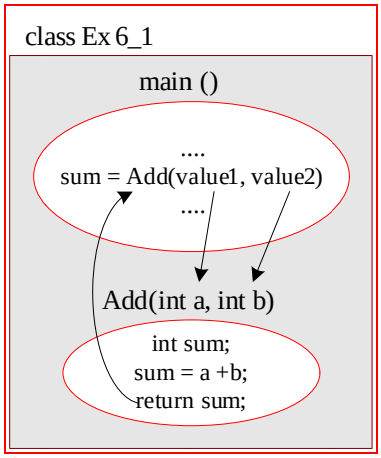

圖 6-3 呼叫加法器函數的運作情形

（C）程式範例：

```
01  //Ex6_1.java
02
03  import java.util.*;
04  /* Ex6_1 類別開始 */
05
06  public class Ex6_1 {
07        /* main 方法開始 */
08        public static void main(String args[]) {
09              Scanner keyin = new Scanner(System.in);
10              int sum;                                    // 區域變數
11              int value1, value2;
12              System.out.printf("請輸入第一個整數 =>");
13              value1 = keyin.nextInt();
14              System.out.printf("請輸入第二個整數 =>");
15              value2 = keyin.nextInt();
16              /* 呼叫 Add() 函數 */
17              sum = Add(value1, value2);
18
19
```

```
20              System.out.printf("%d+%d = %d\n", value1, value2, sum);
21          } /* main 方法結束 */
22
23
24          /* Add 方法開始 */
25          static int Add(int a, int b) {
26              int sum;                                    // 區域變數
27              sum = a + b;
                return sum;
            }   /* Add 方法結束 */
        }   /* Ex6_1 類別結束 */
```

（D） 程式重點說明：

(1) 第 5~29 行：『**public class Ex6_1 { … }**』。宣告主類別的範圍。

(2) 第 7~18 行：『**public static void main(…) {….}**』。宣告 main 方法的範圍。

(3) 第 23~27 行：『**static int Add(int a, int b) { …}**』。宣告 Add 方法的範圍。

(4) 第 9 行：『**int sum;**』。宣告 main 方法的區域變數 sum。

(5) 第 23 行：『**int sum;**』。宣告 Add 方法的區域變數 sum。

6-2-3 範例研討：比較大小方法製作

（A） 程式功能

請製作一套系統，它允許連續輸入五個數字，再輸出其中最大值與最小值的數值。系統功能如下：(請參考 Ex3_5 範例)

```
請輸入第 1 整數 =>34
請輸入第 2 整數 =>56
請輸入第 3 整數 =>12
請輸入第 4 整數 =>98
請輸入第 5 整數 =>67
最大數值是 98
最小數值是 12
```

(B) 製作技巧

吾人嘗試製作兩個比較大小方法，一者 max(a, b) 則傳會兩數較大者，另一者 min(a, b) 則傳回兩數中較小則，程式架構如下：

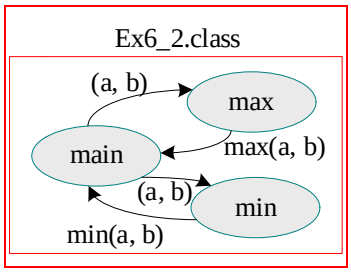

圖 6-4 Ex6_2 程式架構

(C) 程式範例

```
01  //Ex6_2.java
02  import java.util.*;
03  public class Ex6_2 {    // Ex6_2 類別開始
04
05          public static void main(String args[]) {
06                  Scanner keyin = new Scanner(System.in);
07                  int max=0, min=999;
08                  int value;
09                  for(int i=0; i<5; i++){
10                      System.out.printf("請輸入第 %d 整數 =>", i+1);
11                      value = keyin.nextInt();
12                      max = Max(max, value);
13                    min = Min(min, value);
14                  }
15                  System.out.printf("最大數值是 %d\n", max);
16
17                  System.out.printf("最小數值是 %d\n", min);
18          } /* main 方法結束 */
19
20
21          static int Max(int a, int b) {    //Max 方法開始
22                 if (a > b)
23                   return a;
24                 else
25                   return b;
26          }    //Max 方法結束
27
28
29          static int Min(int a, int b){    //Min 方法開始
30          if(a <b)
```

```
31              return a;
32          else
33              return b;
            }//Min 方法結束
        } /* Ex6_2 類別結束 */
```

6-2-4 範例研討：麻將選擇出牌次序

（A）程式功能：Ex6_2.java

麻將桌上常出現需要客戶擲骰子來決定應先順序，每次是擲三個骰子（1～6），由它們的和計算大小。請製作一個程式，假設有四位牌友（東、西、南、北），分別擲三個骰子，並能選出最大數值。期望操作介面如下：

```
D:\Java1_book\chap6>java Ex6_2
請 東方 擲骰子 (按 Enter 鍵) =>
        擲出 => 6   5   2
東方擲的是: 13
請 南方 擲骰子 (按 Enter 鍵) =>
        擲出 => 1   1   5
南方擲的是: 7
請 西方 擲骰子 (按 Enter 鍵) =>
        擲出 => 1   3   4
西方擲的是: 8
請 北方 擲骰子 (按 Enter 鍵) =>
        擲出 => 1   1   1
北方擲的是: 3
擲出最大的是 = 13
```

（B）製作技巧研討：

每位牌友都需要擲 3 個骰子，我們可以將擲骰子動作製作成函數（game()）；另外，也需製作一只比較大小的函數（max()），如圖 6-3 所示。主程式 main 呼叫 game () 函數時，不需攜帶任何引數，但執行完畢後，會傳回 3 個骰子的總和；呼叫 max() 函數時，需攜帶兩個函數，但會

傳回其中較大者。

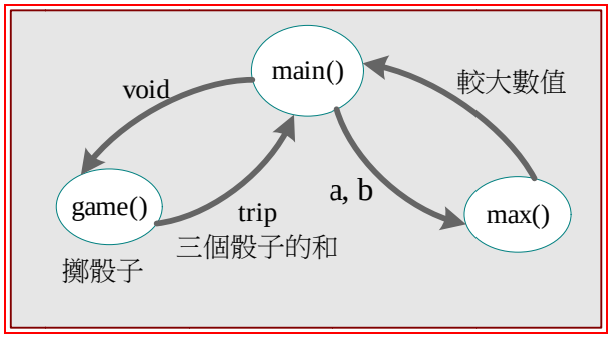

圖 6-5 Ex6_2 程式架構

（C）程式範例：

```
01  //Ex6_2.java
02
03  import java.util.*;
04  public class Ex6_2 {
05          public static void main(String args[]) {
06                  int maxNum, num1, num2, num3, num4;
07                  Scanner keyin = new Scanner(System.in);
08                  System.out.printf("請 東方 擲骰子 (按 Enter 鍵) =>");
09
10                  keyin.nextLine();       //暫停的功能
11                  num1 = game();
12                  System.out.printf("東方擲的是: %d\n", num1);
13
14
15                  System.out.printf("請 南方 擲骰子 (按 Enter 鍵) =>");
16                  keyin.nextLine();       //暫停的功能
17                  num2 = game();
18                  System.out.printf("南方擲的是: %d\n", num2);
19
20
21                  System.out.printf("請 西方 擲骰子 (按 Enter 鍵) =>");
22                  keyin.nextLine();       //暫停的功能
23                  num3 = game();
24                  System.out.printf("西方擲的是: %d\n", num3);
25
26
27                  System.out.printf("請 北方 擲骰子 (按 Enter 鍵) =>");
28                  keyin.nextLine();       //暫停的功能
29                  num4 = game();
30
31                  System.out.printf("北方擲的是: %d\n", num4);
32
```

```
33
34              maxNum = max(num1, max(num2, max(num3, num4)));
35              System.out.printf("擲出最大的是  = %d\n", maxNum);
36
37      }
38      static int game() {
39              Random random = new Random();
40              int k, sum = 0, ran;
41              System.out.printf("\t 擲出  => ");
42              for (k=1; k<=3; k++) {
43                      ran= 1 + random.nextInt(6);
44                      System.out.printf("%d      ", ran);
45                      sum = sum + ran;
46              }
47              System.out.printf("\n");
48              return sum;
49      }
50      static int max(int a, int b) {
                if (a > b)
                        return a;
                else
                    return b;
        }
}
```

（D） 程式重點說明

(1) 第 8 行：『**System.out.printf("請 東方 擲骰子 (按 Enter 鍵) =>")**』。顯示要求玩家自行啟動亂數產生。

(2) 第 9 行：『**keyin.readLine();**』。系統執行到此命令時，會暫停等待使用者由鍵盤輸入數值，並按下『Enter』鍵，再讀入所輸入資料。但此程式並沒有要求使用者輸入任何資料，讀入資料也沒有存入任何變數，其表示僅等待使用者敲入『Enter』鍵而已。

(3) 第 4~50 行：『**public Ex6_2 { …}**』。宣告主類別的範圍。

(4) 第 5~31 行：『**public static void main(..) { …}**』。宣告主方法的範圍。

(5) 第 32~43 行：『**static int game() { …}**』。宣告 game() 方法的範圍。

(6) 第 44~49 行：『**static int max(int a, int b) { …}**』。宣告 max() 方法範圍。

6-2-5 自我挑戰：擲骰子搏奕遊戲

（A）系統功能：PM6_1.java

以前夜市常出現擲骰子的搏奕遊戲，方法是參與賭博者依序擲出三個骰子到碗公內，總數最大者贏了此局。請編寫一程式來取代賭具（碗公與骰子），並假設只有四個人參與賭博。程式功能是參與者輸入自己名字，系統立即顯示所擲骰子的總數，最後顯示最高者命名與數值。期望操作介面如下：

```
G:\Examples\chap6>java PM6_1
請玩家輸入姓名並擲骰子 =>丁一
        您擲出 => 3   1   3   合計: 7
請玩家輸入姓名並擲骰子 =>劉二
        您擲出 => 6   6   2   合計: 14
請玩家輸入姓名並擲骰子 =>張三
        您擲出 => 6   2   3   合計: 11
請玩家輸入姓名並擲骰子 =>李四
        您擲出 => 4   6   1   合計: 11
恭喜!! 劉二 先生擲出 14 贏得此局
```

（B）製作技巧提示：

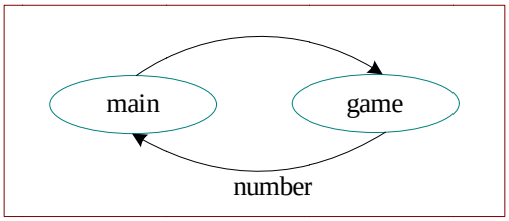

圖 6-6

吾人可規劃兩個方法（main 與 game）於主類別內（PM6_1）。game 方法處理擲出 3 個骰字的動作，包含分別顯示每只骰子的數字，以及計算總和，並傳回給主方法 main。虛擬碼提示如下：

```
導入相關套件；
主類別宣告範圍 {       // 主類別開始
        主方法 main 宣告範圍 {       // main 方法開始
```

```
            宣告擲出最大數值者姓名並設定初值（String maxName = “” ）;
            宣告擲出最大數值並設定初值（int maxNum = 0 ）;
            For (int i=1; i<=4; i++) {
                  讀入玩家姓名（  name ）;
                  呼叫擲骰子函數（number = game() ）;
                  If (number > maxNum) {
                        maxNum = number;
                        maxName = name;
                  }
            }
            輸出最大者姓名與數字（maxName, maxNum ）;
      }     // main 方法結束
      宣告 game 方法範圍 {   // game 方法開始
            宣告產生亂數物件；
            產生並顯示 3 個 1~6 之間亂數；
            計算總和並回傳；
      } // game 方法結束
} // 主類別結束
```

6-3 方法變數與類別變數

6-3-1 變數的生存範圍

某一變數被宣告產生之後，可以使用的範圍如何，這也值得商榷的，稱為生存範圍。基本上，變數的生存範圍僅限制於被宣告時的程式區塊內，越過了程式區塊，便失去了效用。但程式區塊可能經由多層次的程式區塊所構成，即是區塊內可能再出現子程式區塊，子程式區塊內又可能再出現更內圈的子程式區塊，依此類推層層包圍。每一層次區塊都可能會宣告變數，變數的生存範圍可能會影響到程式運作的正確性如何。

雖然每一層次所宣告變數的生存範圍不同，但也沒有那麼複雜。生存範圍最重要的概念是，『外層次所宣告的變數，在該層次內的任何子層次都是有效的；任何內部所宣告的變數，在它的外層次是無效的』。另一重點是『**程式區塊**』為何，以 Java 語言為例，可能出現的有：

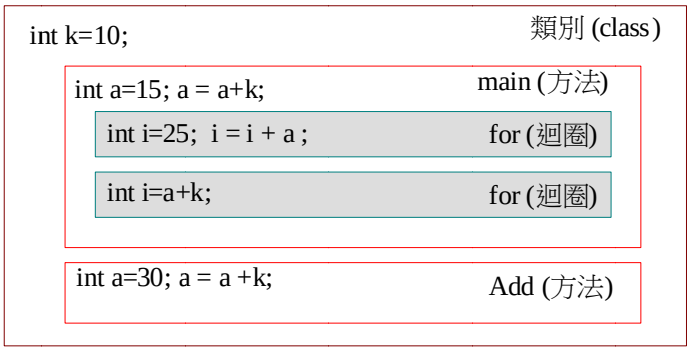

圖 6-7

(1) 類別實體：宣告類別後，利用左右大括號（{ ….}）所包圍的實體區塊（Body）。

(2) 函數實體：宣告函數（方法）後，利用左右大括號（{ ….}）所包圍的實體區塊（Body）。

(3) 選擇敘述區塊：選擇命令（如 if/else/switch/case）的敘述實體。

(4) 迴圈敘述區塊：迴圈命令（如 for/while/do-while）的敘述實體。

簡單的說，利用左右大括號（{ ….}）包起來，即是一個程式實體。大括號內所宣告的變數，超過大括號以外則無效，即是它的生存範圍。另外，任何敘述句所出現的大括號皆有此特性。

6-3-2 類別變數與方法變數

簡單的說，實體區塊內所宣告的變數，僅存活於該區塊內，區塊以外便不存在了。但區塊內可能再宣告另一個或多個子區塊，這種情況下，在父區塊或子區塊所宣告變數的存活空間有何不同？

在傳統程式語言(如 C 語言)之下，父區塊所宣告的變數，在子區塊還是存活著，稱之為『整體變數』；而子區塊所宣告的，在父區塊或同等父區塊的子區塊下是無效的。Java 語言大致上沿用傳統語言規範，要特別留意的是，不同方法之間變數要互相引用時，必須將其宣告成『靜態變數』（static variable），如圖 6-4 所示。

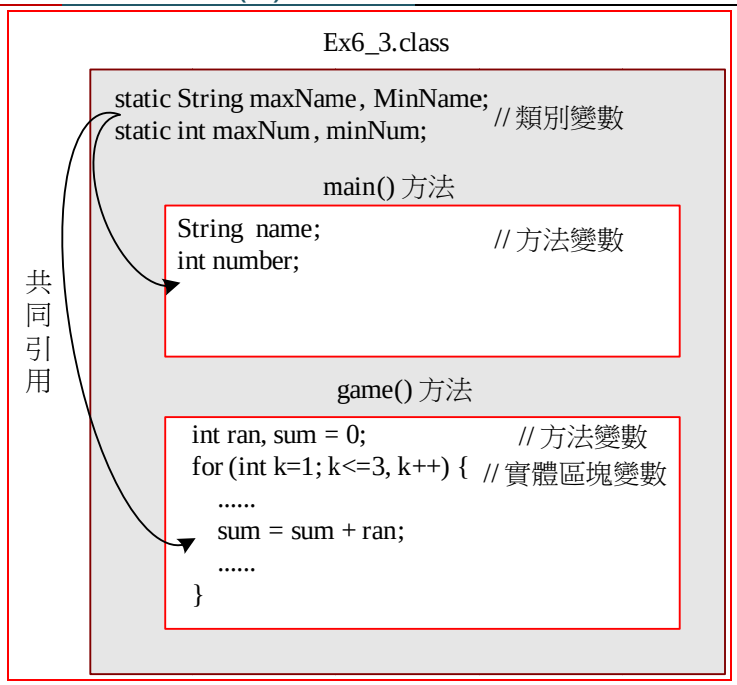

圖 6-8 類別變數與區域變數

6-3-3 範例研討：擲骰子大吃小搏奕

（A）程式功能：Ex6_3.java

擲骰子賭博大多不喜歡『通吃』規則，為了增加次數與趣味性，都採用大吃小的規則，越多人參與賭越好玩，擲出最大者吃最小者。請製作一套大吃小的擲骰子遊系統，參賭者輸入姓名後擲出 3 個骰子，隨時告知目前領先者與點數（隱藏最低者姓名與點數），直接鍵入『Enter』 表示結束參與者，立即顯示出贏家與輸家為何人，以及其擲出點數為何。期望操作介面如下：

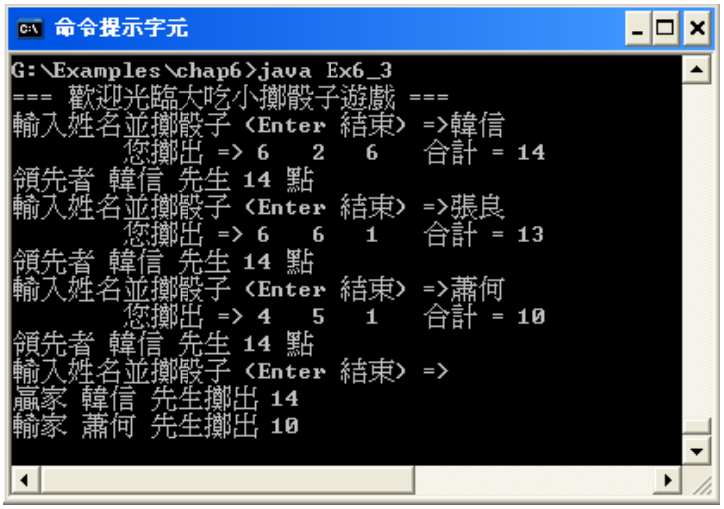

圖 6-9

（B）製作技巧研討：

依照系統要求，我們可以歸納出下列重點：(請參考圖 6_4)

(1) 必須隨時紀錄最高與最低者姓名，以及所擲出的點數，將其宣告成**類別變數(maxName、minName、maxNum 與 minNum)**。

(2) 系統希望參賽者人數不定，利用**『Enter』表示結束**。利用所讀入的字串變數 (name)，其內容的字元長度為 0，則表示輸入『Enter』; 因此，可利用 length() 方法判斷字串長度是否為 0。

(C) 程式範例：

```
01  //Ex6_3.java
02
03  import java.util.*;
04  public class Ex6_3 {
05      /* 宣告類別變數 (整體變數)*/
06       static String maxName="", minName="";
07       static int maxNum=0, minNum=99;
08
09
10       /* main 方法開始 */
11       public static void main(String args[]) {
12          Scanner keyin = new Scanner(System.in); // 區域物件
13
14          String name;                                    // 區域變數
15          int number;                                     // 區域變數
16
17          System.out.printf("=== 歡迎光臨大吃小擲骰子遊戲 ===\n");
18          System.out.printf("輸入姓名並擲骰子 (Enter 結束) =>");
19          name = keyin.nextLine();
20          while(name.length() !=0) {
21                game(name);
22                System.out.printf("領先者 %s 先生 %d 點\n", maxName,
23                                                          maxNum);
24                System.out.printf("輸入姓名並擲骰子 (Enter 結束) =>");
25                name = keyin.nextLine();
26            }
27           System.out.printf("贏家 %s 先生擲出 %d\n", maxName, maxNum);
28          System.out.printf("輸家 %s 先生擲出 %d\n", minName, minNum);
29       }
30       /* game 方法開始 */
31       static void game(String name) {
32
33
34
```

```
35              Random random = new Random();                   // 區域物件
36
37              int ran, sum=0;                                 // 區域變數
38              System.out.printf("\t 您擲出 => ");
39              for (int k=1; k<=3; k++) {                      // 區域變數
40                      ran= 1 + random.nextInt(6);
41                      System.out.printf("%d     ", ran);
42                      sum = sum + ran;
43              }
44              System.out.printf("合計 = %d\n", sum);
45              if (sum > maxNum) {
46                      maxName = name;
47                      maxNum = sum;
48              }
49              if(sum <= minNum) {
                        minName = name;
                        minNum = sum;
                }
        }
}
```

（D） 程式重點分析：

(1) 第 6~7 行：『**static String maxName=””, …**』。宣告 nmaxName、minName、maxNum 與 minNum 變數為靜態類別變數，可被直接存取於 main 與 game 方法內敘述。

(2) 第 12 行：『**String name;**』。宣告字串變數 name，此為 main 方法內的區域變數，也僅存活於 main 方法內。

(3) 第 17 行：『**while(name.length() !=0 { …})**』。此為 while 迴圈敘述，條件是判斷 name 變數內子串長度是否為 0 (name.length() !=0)，如長度大於 0 則表示有輸入任何字元，該字串即是玩家的姓名；如果長度為 0 的話，則表示直接敲入『Enter』鍵，妹有輸入任何字元，表示結束的功能。

(4) 第 18 行：『**game(name);**』。呼叫 game() 方法，並攜帶 name 變數 (玩家姓名) 內容為引數。

(5) 第 29~47 行：『**static void game(String name) { …}**』。宣告 game 方法程式區塊，功能是產生 3 次 1~6 的亂數 (擲骰子動作)，並顯出擲骰子結果。最後與 maxNum 與 minNum 比較是否最大或最小，如是的話，則更改 maxName、maxNum 內容，或 minName 與

minNum 內容 (以上皆是靜態變數)。

(6) 第 31 行：『**Random random = new Random();**』。宣告產生 Random 物件，為區域物件僅存活於 game 方法內。

(7) 第 32 行：『**int ran, sum =0;**』。區域變數，僅存活於 game 方法內。

(8) 第 34 行：『**for(int k=1; k<=3; k++) { ..}**』。變數 k 是 for 迴圈的區域變數，僅存活於 for 迴圈內。

6-3-4 自我挑戰：改良型博弈遊戲

(A)程式功能：PM6_X.java

在 Ex6_3 範例中，如果有兩人擲同樣的點數，會取得先擲的人，我們希望將它修改一下，讓後面擲出的點數與前面的人相同時，請他再重新擲一次。

(B)提示：請自行演練

6-4 遞迴函數

6-4-1 遞迴函數的流程

主程式呼叫函數後，系統轉移到函數上執行，但函數可能再呼叫其他函數，這是非常普遍的現象。如果執行某一函數當中，它會再呼叫自己的函數，則稱之為『遞迴函數』(Recursive function)。許多程式設計師喜歡利用遞迴函數來減低程式的設計量，但遞迴函數會佔用許多記憶體空間，並不是非常理想的程式設計手法。

所謂呼叫函數，即是將函數的程式碼倒入記憶體空間，再去執行他；如果呼叫許多函數，則導入了許多程式碼進入系統。當程式碼進入系統之後，隨之被啟動執行，與原來程式碼沒有任何關係，執行完畢之後，記憶體程式碼隨之被刪除，但磁碟空間內的程式碼還是保留著。如果某一函數一直呼叫自己，則是不斷的導入與自己相同的程式碼進入主記憶體，前後導入的程式碼之間並沒有任何關連，如圖 6-5 所示。

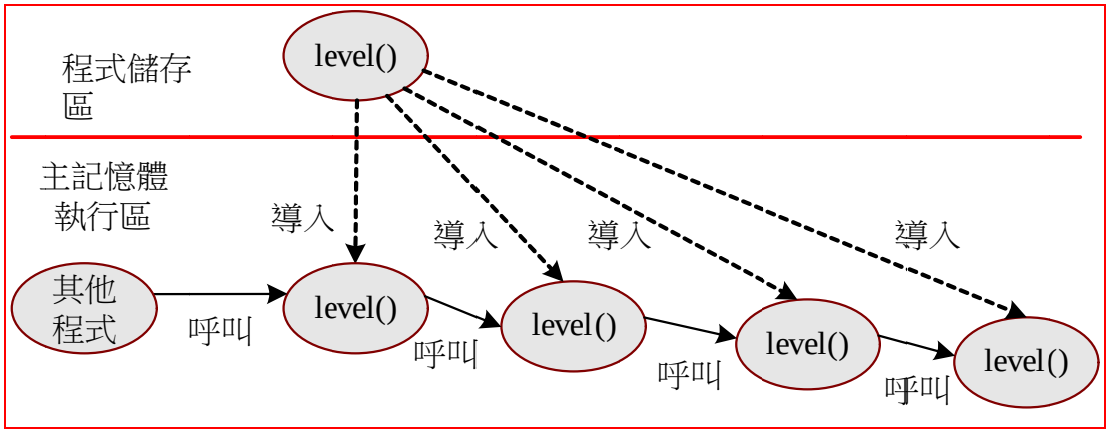

圖 6-10 遞迴函數的運作程序

讀者也許會納悶，如果某一函數會呼叫自己的函數，當然下一次呼叫的函數，還是會再呼叫自己，則將會無窮盡的延伸下去。為了能讓遞迴函數能有停止的時間，函數內必需加入判斷敘述，條件成立再呼叫自己函數，否則必須轉回。因此，遞迴函數有某種固定的格式，範例如下：

範例：level(n) = 1 * 2 *3 * 4 *, …, *n	說明：total = level(5) 的運作程序

static int level(int k) {
  if (k <=1)
  return 1;
  else
  return (k* level(k-1));
}
	圖 6-10-1

假設主程式呼叫上述遞迴函數，並給予引數 5 為例（total = level(5)），其運作程序說明如表 6-1 所示。第一次呼叫時 k=5，條件判斷不成立，則執行 return k * level(k-1)，當呼叫執行 level(4) 時，同樣再呼叫自己，依此類推，一直到 k=1。當 k=1 時，條件成立，則返回 1，接著 level(2) 返回得到 2、level(3) 返回得到 6...，最後 level(5) 返回得到 120。

表 6-1 遞迴函數 level(5) 執行步驟

次數	level(k)	K	K <=1	執行動作	返回
1	level(5)	5	no	return (5 * level(4))	5 * 24 = 120

2	level(4)	4	no	return (4 * level(3))	4 * 6 = 24
3	level(3)	3	no	return (3 * level(2)))	3 * 2 = 6
4	level(2)	2	no	return (2 * level(1))	2*1 = 2
5	level(1)	1	yes	return 1	1

6-4-2 範例研討：累乘遞迴程式

（A）程式功能：Ex6_4.java

請利用呼叫遞迴函數來編寫累乘程式，程式允需輸入一個整數 n，計算並輸出 total = 1 * 2 * 3 * 4 *, …, n（n!）；亦需顯示每次遞迴呼叫的執行內容，期望操作介面如下：

圖 6-11

（B）製作技巧研討：

利用任何迴圈敘述句（for、while）都很容易製作累乘程式，但系統要求利用遞迴函數，只好照辦。另外，系統要求列印每次呼叫遞迴函數的結果，吾人必須在函數內加入列印的功能。

（C）程式範例：

```
01  /Ex6_4.java
02
03  import java.util.*;
04  public class Ex6_4 {
05      public static void main(String args[]) {
06          Scanner keyin = new Scanner(System.in);
07          int total, num;
08          System.out.printf("請輸入一個整數 =>");
09          num = keyin.nextInt();
10          total = level(num);
11          System.out.printf("1*2*3*, ..., %d = %d\n", num, total);
12      }
13
```

```
14      /* 遞迴函數開始 */
15      static int level(int k) {
16          int sum;
17          if (k <= 1)
18              return 1;
19          else {
20              sum = k * level(k-1);
21              System.out.printf("%d * level(%d-1) = %d\n", k, k, sum);
22              return sum;
23          }
24      } /* 遞迴函數結束 */
    }
```

（D） 程式重點說明：

(1) 第 14~23 行：『**static int level(int k) {…}**』。遞迴函數主體。

(2) 第 16~17 行：『**if(k<=1) retrun 1;**』。表示遞迴函數結束。

(3) 第 18~22 行：『**else { sum = k * level(k-1); …}**』。函數呼叫與自己名稱相同的函數，但引數減少（k-1）。

(4) 第 20 行：『**System.out.printf(…)**』。印出每次呼叫函數的執行結果。

6-4-3 自我挑戰：曼波舞步表演系統

（A）程式功能：PM6-2.java

曼波舞步的步法是進一步再退一步、進兩步再退兩步、進三步再退三步，依此類推。前進多少步則相對應後退多少步，當然不會直線進退，可選擇任何路徑進行；又此可見，連續越多步法越困難。請製作一套曼波舞步表演系統，並可隨觀眾喜好選擇表演級數，期望操作系統介面如下：

```
G:\Examples\chap6>java PM6_2
== 曼波舞步表演系統 ==
請輸入表演級數 =>4
第 1 階舞步 =>1 1
第 2 階舞步 =>1 2 2 1
第 3 階舞步 =>1 2 3 3 2 1
```

```
第 4 階舞步 =>1 2 3 4 4 3 2 1
謝謝參觀
```

以第 2 階舞步為例，前進兩步（1, 2），之後緊接著後退兩步（2, 1）。

（B）製作技巧提示。

舞步中可區分『前進』與『後退』兩種步法，前進多少步則後退幾步；吾人可分別利用兩個遞迴函數來達成，前進遞迴函數(front_dance())可連續印出 1、2、3...等 k；後退遞迴函數(back_dance) 則 k...等 3、2、1。另外，由觀眾選擇希望觀賞的級數（num），表演出 n = 1, 2, .., k 階層的舞步。虛擬碼提示如下：

```
觀眾選擇觀賞級數（n）;
for (int k=1; k<=n; k++) {
        顯示表演階層（k）;
        呼叫前進遞迴函數（front_dance(k)）;
        呼叫後退遞迴函數（back_dance(k)）;
}
static void fornt_dance(int k) {        // 前進遞迴函數宣告
        if (k <= 1)
                System.out.printf("1 ");
        else {
                front_dance(k-1);
                System.out.printf("%d ", k);
         }
}
static void back_dance(int k) {         // 後退遞迴函數宣告
        if (k <= 1)
                System.out.printf("1 ");
        else {
                System.out.printf("%d ", k);
                back_dance(k-1);
        }
}
```

6-5 套件引用與方法產生

6-5-1 Java 套件架構

利用傳統語言編寫程式，一套完整的應用程大多是由多個函數所構成。因此，我們可將某些特殊功能編寫成函數程式（Function），再編譯成『目的檔』（Object file），讓其他程式引用，如此可減少許多應用系統開發的時程，也可達到函數重複使用的目的。一般會將所開發的函數集，以其功能歸類某些特定功能的群組，並可依照特定群組來引用，因此稱之為『庫存函數』（Library）。一套系統開發工具（如 C 語言）的強弱，其關鍵在於所提供庫存函數的多寡。某些開發工具除了提供標準庫存函數之外，也會提供某些特殊功能的函數庫，譬如，多媒體處理、繪圖功能、動畫功能等等。

既然 Java 應用系統是由多個類別所構成，我們也可以將某些常用的功能，編寫成類別，並且可重複被引用。但我們回顧一下 Java 的特性，如下：

(1) 類別內有兩個主要成員：**變數與方法**，如果物件（類別產生）著重於變數，則類似於『結構變數』，主要用描述真實實體的特性。如果著重於方法，則類似於『函數』功能，主要實現某一特定功能的程式設計。因此，Java 的函數庫，表示類別中較著重於方法的開發。

(2) 每一方法實現某一特定功能，一般會將功能相似的方法集中歸納於同一類別裡；即是某一類別裡可以取得相對應功能的方法。

(3) 反過來講，某一特殊功能並無法由一個類別承擔，必須由許多類別儲存較細部的分類。譬如，Windows 上的繪圖套件（awt），便需要多個類別來實現。因此，Java 將某一特定功能的類別，再組合成某一功能套件（Package）。

(4) Java 將所提供套件、類別、方法以目錄方式儲存，而目錄的根目錄以 java 表示，如，java.lang.String.substring()。其中 java.lang 為套件名稱（language 套件）、String 為類別名稱、另 substring() 為方法名稱。

圖 6-6 為 Java 套件的結構圖，表 6-2 為 Java 內定的子套件（節錄部分），當然還有許多子套件陸續會被發表出來。每一子套件都針對某一特殊領域，提出許多類別。

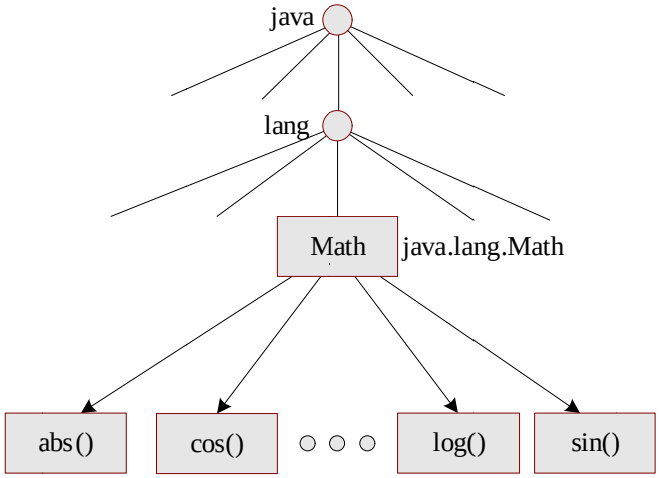

圖 6-12 Java 套件結構圖

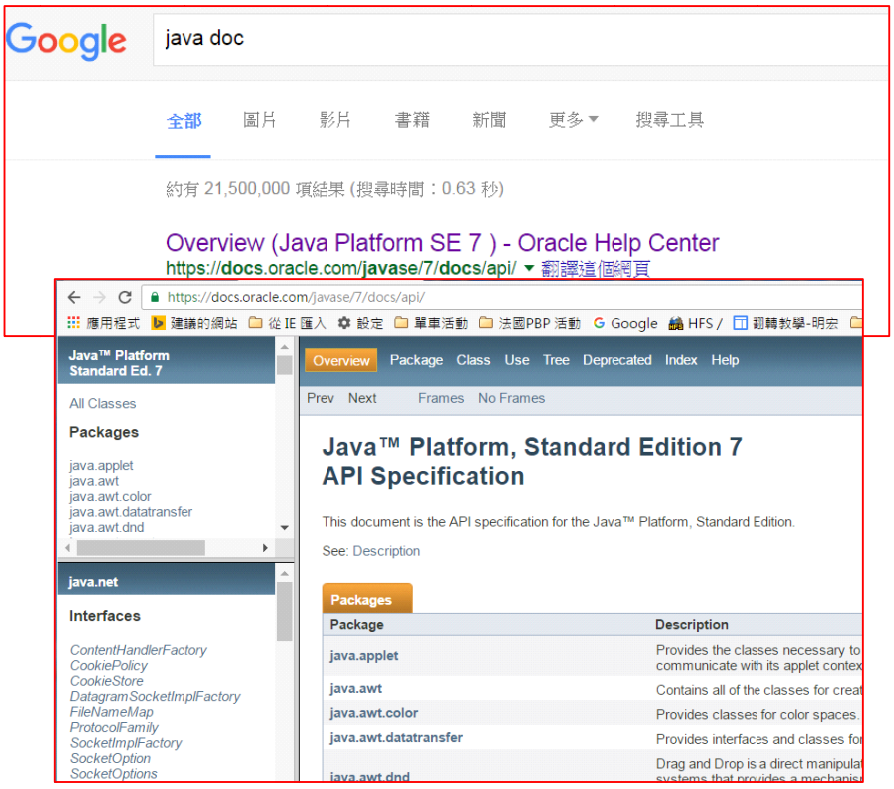

圖 6-13

表 6-2 Java 子套件（節錄）

套　件	說　明
java.applet	Java Apllet 套件含有 Applet 類別及其介面，可以建立 Applet/瀏覽器，以及多媒體表現平台。
java.awt	Java Abstract Windows Toolkit 套件包含所有相關視窗 GUI 的類別及介面。
java.awt.event	Java AWT 可能產生事件的類別與介面。

java.io	Java Input/Output 套件包含資料輸入/輸出的類別與介面。
java.lang	Java Language 套件包含許多 Java 程式可能使用到的類別及介面；編譯時會自動導入。
java.net	Java Networking 套件幫函網路功能的類別。
java.text	包含國際各種語言顯示的類別。
java.util	Java Utilities 套件包含共用程式的類別及介面。
javax.swing	Java Swing GUI Components 套件提供支援可攜式 GUI 類別；自 Java 2 之後，大多取代 java.awt 套件。
javax.swing.event	包含 Swing 可能產生事件的類別；自 Java 2 之後，大多取代 java.awt.event 套件。
……	….

6-5-2 套件導入 - import

套件是『**類別**』與『**介面**』的整合，Java 利用目錄方式來儲存不同的套件；也就是說，一個套件由一個目錄樹來整合該套件之下的類別，又利用目錄整合同一類別底下的方法。引用套件時，必須在程式的第一行開始利用 import 導入套件，語法如下：

導入套件語法：	範例：
imort packetName.subPacket.className; import packetName.*;	import java.util.Scanner; import java.util.*;

第一行表示導入 packageName 套件中的子套件（subPacket）內的 className 類別，這種表示法最完整。但如果同一子套件內多個類別時，也可簡化 packetName.*，其中『*』表示在此目錄之下所有的意思。

在 Java 語言中，除了 java.lang 套件會自動導入，而不用 import 標明外，其他套件都必須利用 import 命令導入。當套件內類別被導入後，有兩種呼叫方法：

(1) 物件方法：利用 new 命令，將類別衍生出物件，原類別上所描述的方法，便隨之轉換成物件方法。使用者可以利用物件方法引用所需的方法。

(2) 類別方法：不經由 new 產生物件，而使用者直接呼叫類別內的方法。許多較常用功能的套件都採用此方法，譬如 java.lang 套件內的方法。

6-5-3 範例研討：製作工程計算器

（A）程式功能：Ex6_5.java

請製作一套工程計算器，使用者輸入數值後，可選擇 log()、log10()、sqrt()、sin()、cos() 與 tan() 等函數計算，並輸出顯示其結果，期望操作介面如下：

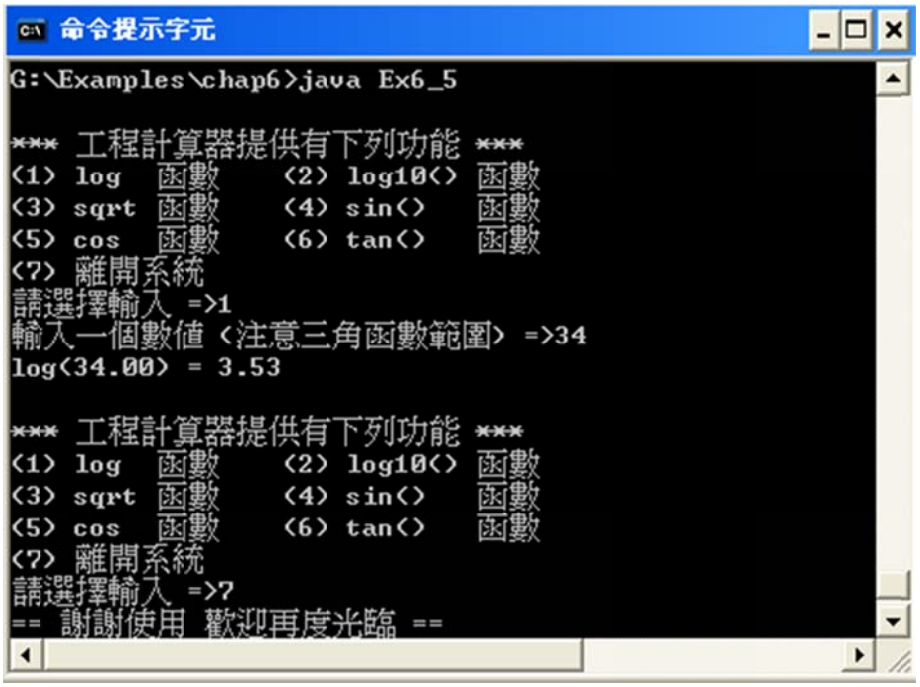

圖 6-14

（B）製作技巧研討：

本系統需引用包含有三角函數的數學套件（java.lang.Math），與允許鍵盤輸入套件，吾人選用 java.util.Scanner；當然採用 java.io 套件亦可。另外，系統希望能重複選擇輸入使用，需利用 while 迴圈判斷是否結束，與 switch/case 判斷選擇項目，並處理相關事項。選單視窗會重複出現，吾人可將其編寫成函數呼叫（disp_men()）；但它僅輸出顯示，則沒有引數與傳回值。

（C）程式範例：

```
01 //Ex6_5.java
02
03 import java.lang.*;                    // 導入子套件 (包含 Math 類別)
04
05 import java.util.Scanner;              // 導入套件內的類別
06 public class Ex6_5 {
07       public static void main(String args[]) {
```

```
08
09          /* 利用套件內類別 java.util.Sacnner 產生物件 keyin */
10          Scanner keyin = new Scanner(System.in);
11          double value, ans=0;
12          int select;
13          disp_men();
14
15          /* 呼叫引用 keyin 的物件方法 nextInt() */
16          select = keyin.nextInt();
17          while( select != 7) {
18                System.out.printf("輸入一個數值 (注意三角函數範圍) =>");
19
20                /* 呼叫 keyin 的 nextDouble 物件方法 */
21                value = keyin.nextDouble();
22                switch(select) {
23                      case 1:
24                            /* 直接引用類別 Math 的類別方法 */
25                            ans = Math.log(value);
26                            System.out.printf("log(%.2f) = %.2f\n", value, ans);
27                            break;
28                      case 2:
29                            /* 直接引用類別 Math 的類別方法 */
30                            ans = Math.log10(value);
31                            System.out.printf("log10(%.2f) = %.2f\n'', value, ans);
32                            break;
33                      case 3:
34                             /* 直接引用類別 Math 的類別方法 */
35                            ans = Math.sqrt(value);
36                            System.out.printf("sqrt(%.2f) = %.2f\n", value, ans);
37                            break;
38                      case 4:
39                            ans = Math.sin(value);
40                            System.out.printf("sin(%.2f) = %.2f\n", value, ans);
41                            break;
42                      case 5:
43                            ans = Math.cos(value);
44                            System.out.printf("cos(%.2f) = %.2f\n", value, ans);
45                            break;
46                      case 6:
47                            ans = Math.tan(value);
48                            System.out.printf("tan(%.2f) = %.2f\n", value, ans);
49                            break;
50                      default:
51                            System.out.printf("錯誤輸入 \n”);
52
53
54
55
```

```
56                  }
57                  disp_men();
58                  select = keyin.nextInt();
59              }
60              System.out.print("==  謝謝使用  歡迎再度光臨  ==\n");
61          }
62          static void disp_men() {
63                  System.out.printf("\n***  工程計算器提供有下列功能  ***\n");
64                  System.out.printf("(1) log    函數\t (2) log10()  函數\n");
65                  System.out.printf("(3) sqrt  函數\t (4) sin()     函數\n");
66                  System.out.printf("(5) cos    函數\t (6) tan()     函數\n");
67                  System.out.printf("(7)  離開系統\n");
68                  System.out.printf("請選擇輸入  =>");
          }
}
```

（D） 程式重點說明：

(1) 第 3 行：『**import java.lang.*;**』。導入 java.lang 下的所有類別；該套件系統會自動導入可不用再 import。

(2) 第 4 行：『**import java.util.Scanner;**』。僅導入 java.util 套件下 Scanner 類別，其他未被導入，可減少佔用記憶體空間。

(3) 第 9 行：『**Scanner keyin = new Scanner(System.in);**』。利用 Scanner 類別宣告產生 keyin 物件，keyin 物件繼承了 Scanner 類別的所有方法成員（Method member）。

(4) 第 12 行：『**disp_men();**』。呼叫 disp_mem() 函數，該函數無傳回值（void），也不需要傳任何引數。

(5) 第 15 行：『**select = keyin.nextInt();**』。呼叫 keyin 物件內的 nextInt() 物件方法。

(6) 第 24 行：『**ans = Math.log(value);**』。直接呼叫 Math 類別內的 log() 類別方法。

6-5-4 範例研討：密碼設定程式

（A）程式功能：Ex6_6.java

一般資訊系統大多利用『帳號/密碼』過濾用戶的身份。用戶輸入帳號與密碼，系統檢視密碼是否正確，正確的話，同意用戶登入系統；否則會拒絕登入。如此一來，系統需要紀錄用戶密碼，

作為判斷用戶身份之依據。但系統如果以明文儲存用戶密碼，他人只要入侵系統就可以觀察到所有用戶的密碼，因此密碼是不可以『明文儲存』的。也就是說，密碼必須經過加密或雜湊計算後，成為一串亂文格式再儲存。但密碼如設定太短的話，所產生的亂文變化就不大，有心人還是可以猜測出密碼。但一般人為了方便記憶，又不喜歡設定太長的密碼。

『醃製法』密碼計算就是為了解決這個問題。醃製法是系統產生一個亂數稱之為『鹽』,『鹽』與密碼明文串接成一個較長的字串，此新字串經過加密或雜湊演算後，再儲存於系統。請製作一套『醃製法』密碼儲存系統，利用使用者輸入密碼明文後，與系統產生一個『鹽』，再將密碼與鹽組合為另一字串，利用此字串產生一個雜湊值，並印出結果；接著，假設使用者欲進入系統，如密碼輸入正確，則允許進入系統；否則拒絕進入。期望操作介面如下：

```
D:\Java1_book\chap6>javac   -encoding   utf8   Ex6_6.java

D:\Java1_book\chap6>java Ex6_6
請輸入使用者名稱  =>user01
請輸入密碼明文  =>123456
產生鹽亂數  = 942  儲存於電腦系統內
密碼明文與鹽組合  = 123456942
user01  密碼雜湊  = -1867376844

請輸入密碼  (驗證密碼是否正確) =>123456
由系統中取出鹽  = 942
輸入密碼與鹽結合後  = 123456942
輸入密碼雜湊值  = -1867376844
user01  密碼設定正確，該帳戶可使用
```

（B）製作技巧提示：

圖 6-7 為密碼雜湊值產生方法，電腦的密碼系統紀錄著每位使用者的『帳戶名稱』、『密碼雜湊值』與『鹽』。設定密碼的程序是，使用者輸入密碼明文，與系統所產生的『鹽』串接後，再經過『雜湊演算法』計算得到一串固定長度的雜湊值，也將產生的雜湊值與鹽儲存於密碼檔案內。

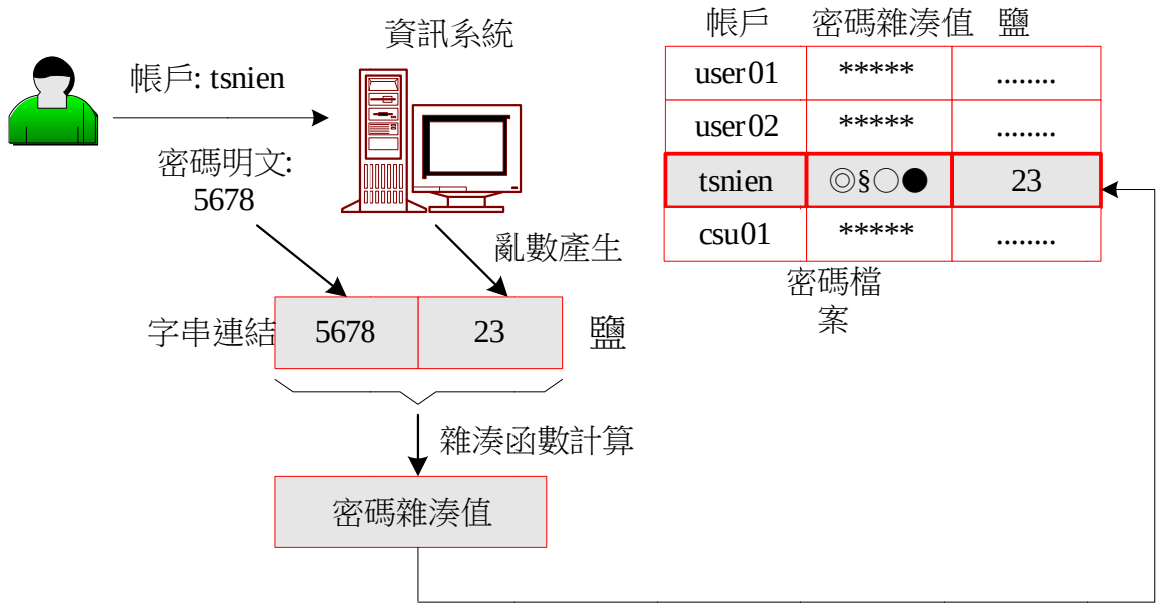

圖 6-15 密碼雜湊值的產生方法。

當使用者欲進入系統，則輸入帳戶名稱（如 tsnien）與密碼；系統則由密碼檔案搜尋出該帳戶的『鹽』，再與使用者所輸入的『密碼』相連結之後，再經過相同的雜湊演算法計算，得到雜湊值，此值與密碼檔案內的『密碼雜湊值』相同的話，則判斷輸入密碼正確；否則拒絕該使用者登入。吾人將以上所敘述的運作程序，編寫虛擬碼如下：

```
導入相關套件（java.util.*;）;
宣告紀錄檔：帳號（String name）、鹽（String salt）、密碼雜湊值（int hash_pas）;
宣告相關變數與物件；
/*  帳號/密碼設定程式  */
讀取使用帳號（name）;
讀取使用者設定密碼（passwd1）;
產生鹽亂數（int value1 = 1 + (int)(Math.random()* 999)）;
轉換鹽亂數成字串（salt    = String.valueOf(value1)）;
鹽與密碼明文連結（passwd2 = passwd1.concat(salt)）;
產生密碼雜湊值（hash_pas = passwd2.hashCode()）;
輸出顯示該帳號（name）、密碼雜湊值（hash_pas）與鹽（salt）;
/*  密碼驗證程式  */
讀取使用者輸入密碼（passwd1）;
鹽與密碼明文連結（passwd2 = passwd1.concat(salt)）;
```

產生密碼雜湊值（test_pas = passwd2.hashCode()）;
if (test_pas == hash_pas)
    顯示 "== 歡迎光臨 藝術化資訊系統 ==";
else
    顯示 "密碼不正確, 拒絕進入系統";

(A) 程式範例：

```
01  //Ex6_6.java
02  /* 請製作一套密碼產生系統，當使用者輸入明文密碼，系統則會產生結果並印出
03  */
04
05  import java.util.Scanner;
06  public class Ex6_6 {
07      public static void main(String args[]) {
08
09          Scanner keyin = new Scanner(System.in);
10          String name, key;                    //登錄使用者名稱, 鹽
11          int hash_pas;                        // 登錄密碼雜湊值
12          String passwd1, passwd2;
13          int value1, value2;
14          System.out.printf("請輸入使用者名稱 =>");
15          name = keyin.nextLine();
16
17          System.out.printf("請輸入密碼明文 =>");
18          passwd1 = keyin.nextLine();
19          value1 = 1 + (int)(Math.random()* 999);
20          key = String.valueOf(value1);
21          System.out.printf("產生鹽亂數 = %s 儲存於電腦系統內\n", key);
22          passwd2 = passwd1.concat(key);
23          System.out.printf("密碼明文與鹽組合 = %s\n", passwd2);
24          hash_pas = passwd2.hashCode();
25          System.out.printf("%s 密碼雜湊 = %d\n", name, hash_pas);
26
27          System.out.printf("\n 請輸入密碼 (驗證密碼是否正確) =>");
28          passwd1 = keyin.nextLine();
29          System.out.printf("由系統中取出鹽 = %s \n", key);
30          passwd2 = passwd1.concat(key);
31          System.out.printf("輸入密碼與鹽結合後 = %s\n", passwd2);
32          value2 = passwd2.hashCode();
33
34
35
36
37
```

```
38              System.out.printf("輸入密碼雜湊值 = %d\n", value2);
39              if (value2 == hash_pas)
40                  System.out.printf("%s 密碼設定正確，該帳戶可使用 \n", name);
41              else
                    System.out.printf("%s 密碼設定不正確，請重新設定\n", name);
        }
    }
```

6-6 專題研討

6-6-1 範例研討：製作 DES 加密/解密工具

(A) 程式功能：Ex6_7.java

請製作一只 **DES (Data Encryption Standard)** 密碼系統加密/解密功能的雛形工具，系統允許使用者輸入明文與產生鑰匙元素後，即顯示加密後密文，以及解密後明文，操作介面如下：

```
D:\Java1_book\chap6>javac -encoding utf8 Ex6_7.java

D:\Java1_book\chap6>java Ex6_7
請輸入明文 =>大家恭喜、新年快樂
請輸入密碼元素(8 個字元) =>12345678
密碼元素：12345678
明文：大家恭喜、新年快樂
加密後密文：h\吽$:?!廄礓 v?    JX     5Q?
解密後明文：大家恭喜、新年快樂
```

(B) 製作技巧提示

要將 DES 密碼系統演算法轉換為程式，是一件不容易的工作，我們可利用 Javax.crypto 套件下的類別程式來完成。

(B) 程式範例：

```
01  //Ex6_7 DES 加密/解密程式
02
03
04  import java.security.SecureRandom;
    import java.util.Scanner;
```

```
05  import javax.crypto.Cipher;
06  import javax.crypto.SecretKey;
07  import javax.crypto.SecretKeyFactory;
08  import javax.crypto.spec.DESKeySpec;
09
10  public class Ex6_7 {
11
12      public static void main(String[] args) {
13          Scanner keyin = new Scanner(System.in);
14          System.out.print("請輸入明文 =>");
15          String plain = keyin.nextLine();
16          System.out.print("請輸入密碼元素(8 個字元) =>");
17          String key = keyin.nextLine();
18          System.out.println("密碼元素：" + key);
19          System.out.println("明文：" + plain);
20          byte[] cipher = encrypt(plain, key);
21          System.out.println("加密後密文：" + new String(cipher));
22          String cipher_n = decrypt(cipher, key);
23          System.out.println("解密後明文：" + cipher_n);
24      }
25
26      // DES 加密程式
27      public static byte[] encrypt(String plain, String key) {
28          try {
29              SecureRandom random = new SecureRandom();
30              DESKeySpec desKey = new DESKeySpec(key.getBytes());
31              SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
32              SecretKey securekey = keyFactory.generateSecret(desKey);
33              Cipher cipher = Cipher.getInstance("DES");
34              cipher.init(Cipher.ENCRYPT_MODE, securekey, random);
35              byte[] result = cipher.doFinal(plain.getBytes());
36              return result;
37          } catch (Throwable e) {
38              e.printStackTrace();
39          }
40          return null;
41      }
42
43      //解密程式
44      public static String decrypt(byte[] content, String key) {
45          try {
46              SecureRandom random = new SecureRandom();
47              DESKeySpec desKey = new DESKeySpec(key.getBytes());
48              SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
49
50
51
52
```

```
53                  SecretKey securekey = keyFactory.generateSecret(desKey);
54                  Cipher cipher = Cipher.getInstance("DES");
55                  cipher.init(Cipher.DECRYPT_MODE, securekey, random);
56                  byte[] result = cipher.doFinal(content);
57                  return new String(result);
58              } catch (Throwable e) {
59                  e.printStackTrace();
60              }
                return null;
            }
        }
```

6-6-2 範例研討：產生 RSA 鑰匙配對程式

(A)程式功能：Ex6_8.java

請製作一套 1024 位元的 RSA 鑰匙配對產生工具，每次執行時會產生不同的鑰匙配對(公開鑰匙與私有鑰匙)，期望操作介面如下：

```
D:\Java1_book\chap6\RSA>javac -encoding utf8 Ex6_8.java

D:\Java1_book\chap6\RSA>java Ex6_8
產生 RSA 鑰匙配對中
將鑰匙配對寫入檔案，並顯示於螢幕
Public Key: 30819f300d06092a864886f70d010101050003818d003081890281810094be1274ff
cd65be48bc02a96fabf8bc5a614b715ff24bdf46afddca2ae5f3ab24a32bc6304699bdb92fe594a9
98e917320e2b2174c4a4a43f216acfdd4533c87e8019e46556cc572edde97b26b91416bc5777795b
90e9d44ebd979638c0acaa0a1842d6252c9d11e157ede3eb0b82cba138d9f3390cb917bcd88f8fb6
92c92d0203010001
Private Key: 30820275020100300d06092a864886f70d0101010500048202 5f3082025b0201000
281810094be1274ffcd65be48bc02a96fabf8bc5a614b715ff24bdf46afddca2ae5f3ab24a32bc63
04699bdb92fe594a998e917320e2b2174c4a4a43f216acfdd4533c87e8019e46556cc572edde97b2
6b91416bc5777795b90e9d44ebd979638c0acaa0a1842d6252c9d11e157ede3eb0b82cba138d9f33
90cb917bcd88f8fb692c92d0203010001028180205652 5ced60124694398bba9a74a0d7122f24dfb
.....
.....
由檔案讀出 RSA 鑰匙配對，並顯示於螢幕
Public Key: 30819f300d06092a864886f70d010101050003818d003081890281810094be1274ff
cd65be48bc02a96fabf8bc5a614b715ff24bdf46afddca2ae5f3ab24a32bc6304699bdb92fe594a9
98e917320e2b2174c4a4a43f216acfdd4533c87e8019e46556cc572edde97b26b91416bc5777795b
90e9d44ebd979638c0acaa0a1842d6252c9d11e157ede3eb0b82cba138d9f3390cb917bcd88f8fb6
92c92d0203010001
Private Key: 30820275020100300d06092a864886f70d0101010500048202 5f3082025b0201000
281810094be1274ffcd65be48bc02a96fabf8bc5a614b715ff24bdf46afddca2ae5f3ab24a32bc63
04699bdb92fe594a998e917320e2b2174c4a4a43f216acfdd4533c87e8019e46556cc572edde97b2
6b91416bc5777795b90e9d44ebd979638c0acaa0a1842d6252c9d11e157ede3eb0b82cba138d9f33
.....
```

```
.....
D:\Java1_book\chap6\RSA>dir/b *.key
private.key        // 私有鑰匙檔案
public.key         // 公開鑰匙檔案
```

(B)製作技巧提示

我們利用 Java.scurity 套件內，產生 RSA 鑰匙配對方法。

(C)程式範例：

```
01  // Ex6_8.java
02  import java.io.File;
03  import java.io.FileInputStream;
04  import java.io.FileOutputStream;
05  import java.io.IOException;
06  import java.security.KeyPair;
07  import java.security.KeyPairGenerator;
08  import java.security.SecureRandom;
09  import java.security.KeyFactory;
10  import java.security.NoSuchAlgorithmException;
11  import java.security.PrivateKey;
12  import java.security.PublicKey;
13  import java.security.spec.InvalidKeySpecException;
14  import java.security.spec.PKCS8EncodedKeySpec;
15  import java.security.spec.X509EncodedKeySpec;
16
17  public class Ex6_8 {
18          public static void main(String[] args) {
19          myKeyPair adam = new myKeyPair();
20          try {
21                  String path = ".";
22                  // Generate the key pair (public key and private key)
23                  KeyPairGenerator keygen = KeyPairGenerator.getInstance("RSA");
24                  SecureRandom random = new SecureRandom();
25                  // 給與亂數因子，指定鑰匙長度，再產生鑰匙
26
27                  System.out.println("產生 RSA 鑰匙配對中");
28                  random.setSeed("seedVaule".getBytes());
29                  keygen.initialize(1024, random);                // Generate 1024-bit keys
30                  KeyPair generatedKeyPair = keygen.generateKeyPair();
31                  // 儲存鑰匙配對
32
33                  System.out.println("將鑰匙配對寫入檔案，並顯示於螢幕");
34                  adam.dumpKeyPair(generatedKeyPair);             // Print the generated keys
35                  adam.SaveKeyPair(path, generatedKeyPair);   // Store the keys into two files
36                  // 讀取鑰匙配對
37
38                  System.out.println("\n由檔案讀出 RSA 鑰匙配對，並顯示於螢幕");
```

```
39                  KeyPair loadedKeyPair =
40                          adam.LoadKeyPair(path, "RSA");      // Load the keys from files
41                  adam.dumpKeyPair(loadedKeyPair);            // Print the loaded keys
42              } catch (Exception e) {
43                  e.printStackTrace();
44                  return;
45              }
46      }
47  }
48
49  class myKeyPair {
50      // 列印鑰匙配對方法
51      public void dumpKeyPair(KeyPair keyPair) {
52          PublicKey pub = keyPair.getPublic();
53          System.out.println("Public Key: " + getHexString(pub.getEncoded()));
54
55          PrivateKey priv = keyPair.getPrivate();
56          System.out.println("Private Key: " + getHexString(priv.getEncoded()));
57          }
58
59      public String getHexString(byte[] b) {
60          StringBuilder result = new StringBuilder();
61          for (int i = 0; i < b.length; i++)
62              result.append(Integer.toString((b[i] & 0xff) + 0x100, 16).substring(1));
63          return result.toString();
64          }
65
66      // 將鑰匙配對寫入檔案方法
67      public void SaveKeyPair(String path, KeyPair keyPair) throws IOException {
68          PrivateKey privateKey = keyPair.getPrivate();
69          PublicKey publicKey = keyPair.getPublic();
70
71          // 儲存公開鑰匙
72          X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(
73              publicKey.getEncoded());
74          FileOutputStream fos = new FileOutputStream(path + "/public.key");
75          fos.write(x509EncodedKeySpec.getEncoded());
76          fos.close();
77          // 儲存私有鑰匙
78          PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(
79              privateKey.getEncoded());
80          fos = new FileOutputStream(path + "/private.key");
81          fos.write(pkcs8EncodedKeySpec.getEncoded());
82          fos.close();
83      }
84
85      // 由檔案中讀取鑰匙配對方法
86      public KeyPair LoadKeyPair(String path, String algorithm)
87          throws IOException, NoSuchAlgorithmException, InvalidKeySpecException {
88
89
90
```

```
91              // 讀取公開鑰匙
92              File filePublicKey = new File(path + "/public.key");
93              FileInputStream fis = new FileInputStream(filePublicKey);
94              byte[] encodedPublicKey = new byte[(int) filePublicKey.length()];
95              fis.read(encodedPublicKey);
96              fis.close();
97
98              // 讀取私有鑰匙方法
99              File filePrivateKey = new File(path + "/private.key");
100             fis = new FileInputStream(filePrivateKey);
101             byte[] encodedPrivateKey = new byte[(int) filePrivateKey.length()];
102             fis.read(encodedPrivateKey);
103             fis.close();
104
105             // Generate KeyPair.
106             KeyFactory keyFactory = KeyFactory.getInstance(algorithm);
107             X509EncodedKeySpec publicKeySpec = new
108                     X509EncodedKeySpec(encodedPublicKey);
109             PublicKey publicKey = keyFactory.generatePublic(publicKeySpec);
110
111             PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(
112                     encodedPrivateKey);
113             PrivateKey privateKey = keyFactory.generatePrivate(privateKeySpec);
114
115             return new KeyPair(publicKey, privateKey);

        }
}
```