第七章
函數

本章學習重點

- 認識 Java 的函數
- 學習函數引數傳遞的方式
- 學習遞迴函數的撰寫
- 認識函數的多載

7.1 函數的基本概念

Java 的慣例是把函數稱為 method。method 可用如下的語法來定義：

```
public static 傳回值型態 method名稱(型態 引數1, 型態 引數2,...)
{
    程式敘述;
    return 運算式;
}
```

method 的主體

格式 7.1.1
定義 method

7.1.1 簡單的範例

app7_1 是一個簡單的 method 實例。

```
/* app7_1 OUTPUT---
********************
Knowledge is power
********************
---------------------*/
```

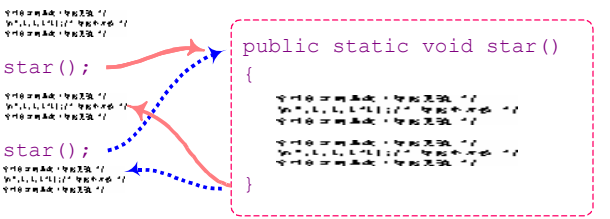
```
01  // app7_1, 簡單的範例
02  public class app7_1
03  {
04    public static void main(String args[])
05    {
06       star();  // 呼叫 star() method
07       System.out.println("Knowledge is power");
08       star();  //呼叫 star() method
09    }
10
11    public static void star()  // star() method
12    {
13       for(int i=0;i<20;i++)
14          System.out.print("*");  // 印出 20 個星號
15       System.out.print("\n");    // 換行
16    }
17  }
```

main() method

star() method

star() method 被呼叫與執行的流程如下 所示：

```
public class app7_1 {
    public static void main(String   args[])
    {
        star();
        star();
    }
}
```



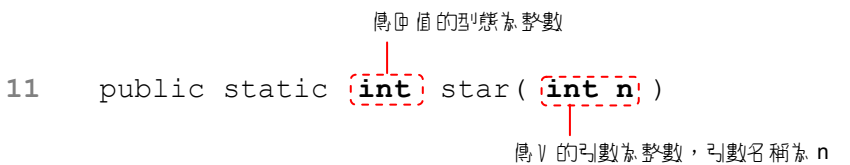
7.1.2 method 的引數與傳回值

app7_2 可接收一個整數引數 n，印出 2*n 個星號後，傳回整數 2*n。

```
01  // app7_2, 簡單的範例--method 的引數與傳回值
02  public class app7_2
03  {
04     public static void main(String args[])
05     {
06        int i;       // 宣告整數變數 i，此變數的有效範圍僅止於 main() method
07        i=star(8);   // 傳入 8 給 star()，並將 i 接收傳回的數值
08        System.out.println(i + " stars printed");
09     }
10
11    public static int star(int n)  // star() method
12     {
13        int i;  // 宣告整數變數 i，此變數的有效範圍僅止於 star() method
14        for(i=1;i<=2*n;i++)
15           System.out.print("*");       // 印出 2*n 個星號
16           System.out.print("\n");      // 換行
17        return 2*n;                    // 傳回整數 2*n
18     }
19  }
```

```
/* app7_2 OUTPUT---
***************
16 stars printed
--------------------*/
```

app7_2 的 star() method 之說明如下圖所示：



app7_3 是一個計算長方形對角線長度的範例。

```
01  // app7_3, 計算長方形對角線的長度
02  public class app7_3
03  {
04     public static void main(String args[])
05     {
06        double num;
07        num=show_length(7,3);  // 傳入 7 與 3 兩個引數到 show_length()裡
08        System.out.println("length = "+num);
09     }
10
11    public static double show_length(int m, int n)
12     {
13        return Math.sqrt(m*m+n*n);  // 傳回對角線長度
14     }
15  }
```

```
/* app7_3 OUTPUT----------
length = 7.615773105863909
--------------------------*/
```

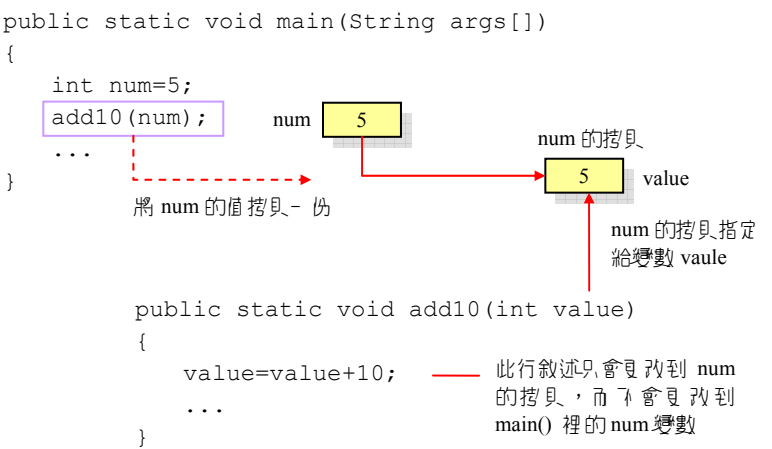
7.1.3 引數是如何傳遞給 method 的？

基本資料型態的變數傳遞到 method 的方式，均是以「傳值」的方式來進行，如下面的範例：

```
01  // app7_4, method 傳值的範例
02  public class app7_4
03  {
04     public static void main(String args[])
05     {
06        int num=5;
07        add10(num);            // 呼叫 add10(),並傳遞 num
08        System.out.println("in main(), num = "+num);
09     }
10
11     public static void add10(int value)
12     {
13        value=value+10;        // 將 value 的值加 10 之後，設回給 value
14        System.out.println("in add10(), value = "+value);
15     }
16  }
```

```
/* app7_4 OUTPUT---
in add10(), value = 15
in main(), num = 5
--------------------*/
```

於 app7_4 中，value 與 num 的值之變化如下圖所示：



7.2 傳遞陣列到 method 裡

7.2.1 傳遞一維陣列

app7_5 是傳遞一維陣列到 largest() 的範例。

```
01  // app7_5, 簡單的範例
02  public class app7_5
03  {
04     public static void main(String args[])
05     {
06        int score[]={5,3,8,12,6,7};  // 宣告一維陣列 score
07        largest(score);   // 將一維陣列 score 傳入 largest() method
08     }
09                  將陣列 score 傳入 largest() method 裡
10
11     public static void largest(int arr[])
12     {                                  接收一維的整數陣列
13        int max=arr[0];
14        for(int i=0;i<arr.length;i++)
15          if(max<arr[i])
16             max=arr[i];
17        System.out.println("largest num = "+max);
18     }
19  }
```

```
/* app7_5 OUTPUT---

largest num = 12
--------------------*/
```

7.2.2 傳遞二維陣列

app7_6 是傳遞二維陣列的範例：

```
01  // app7_6, 傳遞二維陣列
02  public class app7_6
03  {
04     public static void main(String args[])
05     {
06
07        int A[][]={{51,38,82,12,34},{72,64,19,31}}; // 定義二維陣列
08        print_mat(A);
09     }
10
11     public static void print_mat(int arr[][])
12     {
13        for(int i=0;i<arr.length;i++)
14        {
15           for(int j=0;j<arr[i].length;j++)
16              System.out.print(arr[i][j]+" ");    // 印出陣列值
17           System.out.print("\n");
18        }
19     }
20  }
```

將二維陣列A傳入 print_mat() method 裡

接收二維的整數陣列

```
/* app7_5 OUTPUT---

51 38 82 12 34
72 64 19 31
--------------------*/
```

7.2.3 傳回陣列的 method

app7_7 是傳回二維陣列的練習。

```
01  // app7_7, 設計傳回二維陣列的method
02  public class app7_7
03  {
04     public static void main(String args[])
05     {
06       int A[][]={{51,38,82,12,34},{72,64,19,31}};  // 定義二維陣列
07       int B[][]=new int[2][];          // 宣告陣列B，並設定列數
08       B[0]=new int[5];                 // 設定陣列B第一列的行數
09       B[1]=new int[4];                 // 設定陣列B第二列的行數
10
11       B=add10(A);   // 呼叫 add10()，並把傳回的值設給陣列B
12       for(int i=0;i<B.length;i++)    // 印出陣列的內容
13       {
14          for(int j=0;j<B[i].length;j++)
15             System.out.print(B[i][j]+" ");
16          System.out.print("\n");
17       }
18     }
```

傳回二維的整數陣列

```
19
20     public static int[][] add10(int arr[][])
21     {
22        for(int i=0;i<arr.length;i++)
23           for(int j=0;j<arr[i].length;j++)
24              arr[i][j]+=10;          // 將陣列元素加10
25        return arr;                   // 傳回二維陣列
26     }
27  }
```

```
/* app7_7 OUTPUT--

61 48 92 22 44
82 74 29 41
---------------------*/
```

7.2.4 陣列的傳遞機制

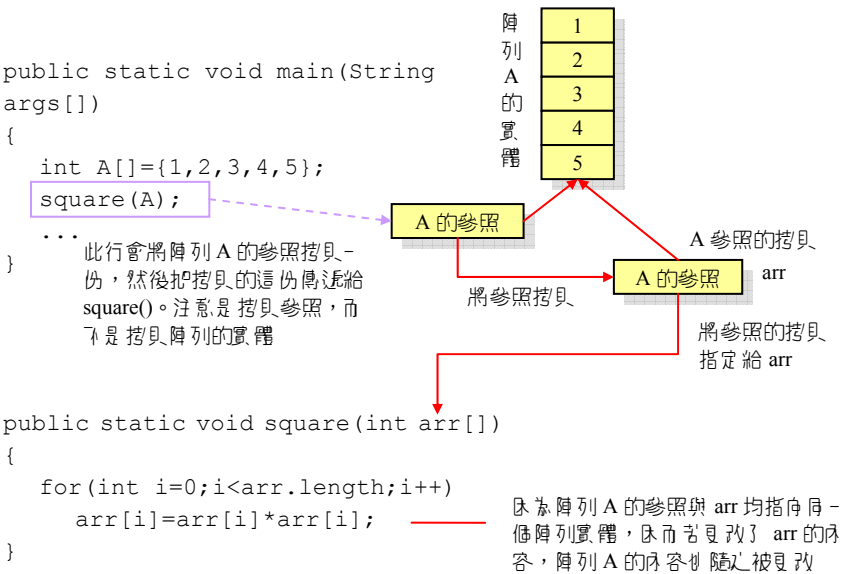
傳遞陣列時，是以「傳參照」（pass by reference）的方式來進行。下面的範例說明了「傳參照」和「傳值」的機制的不同。

```
01  // app7_8,「傳參照」的範例
02  public class app7_8
03  {
04     public static void main(String args[])
05     {
06        int A[]={1,2,3,4,5};
07
08        square(A);   // 呼叫 square()，並傳遞陣列 A
09
10        System.out.println("呼叫 square() method 之後...");
11
12        for(int i=0;i<A.length;i++)     // 印出陣列的內容
13           System.out.print(A[i]+" ");
14
15        System.out.println();
16     }
17
18     public static void square(int arr[])
19     {
```

```
20          for(int i=0;i<arr.length;i++)
21             arr[i]=arr[i]*arr[i];          // 將陣列的元素值平方
22       }
23    }
```

```
/* app7_8 OUTPUT--------
呼叫 square() method 之後...
1 4 9 16 25
--------------------------*/
```

square() method 的呼叫過程可用下圖來表示：



7.3 遞迴

遞迴就是 method 自己呼叫自己。

階乘函數（factorial function，$n!$）便可利用遞迴的方式來完成：

$$\text{fac}(n)=\begin{cases}1\times 2\times\cdots\times n; & n\geq 1\\ 1; & n=0\end{cases}$$ （非遞迴的運算方式）

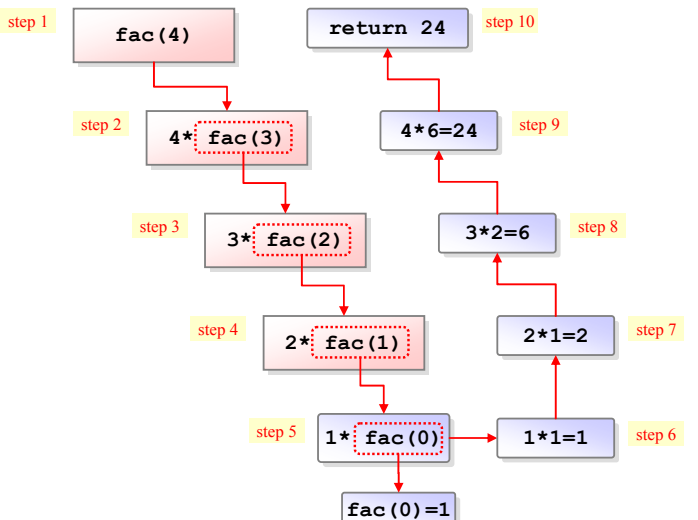
$$\text{fac}(n)=\begin{cases}n\times \text{fac}(n-1); & n\geq 1\\ 1; & n=0\end{cases}$$ （遞迴的運算方式）

以階乘函數來說明如何撰寫遞迴 method。

```
01  // app7_9, 簡單的遞迴 method
02  public class app7_9
03  {
04     public static void main(String args[])
05     {
06        System.out.println("1*2*..*4 = "+fac(4));
07     }
08     public static int fac(int n)  // fac() method
09     {
10        if(n==0)        // 設定終止條件
11           return 1;
12        else
13           return n*fac(n-1);   // 遞迴計算
14     }
15  }
```

```
/* app7_9 OUTPUT--
1*2*...*4=24
--------------------*/
```

計算 $fac(4)=4\times3\times2\times1=24$ ，其遞迴計算的過程如下圖所示：



7.4 method 的多載

多載（overloading）可將功能相似的 method，以相同名稱命名，編譯器會根據引數的個數與型態，自動執行相對應的 method。

下面的程式碼是說明引數型態不同的函數多載：

```
01  // app7_10,引數型態不同的函數多載
02  public class app7_10
03  {
04     public static void main(String args[])
05     {
06        int a=5, b[]={1,2,3,4};
07        show(a);      // 將整數 a 傳遞到 show()裡
08        show(b);      // 將整數陣列 b 傳遞到 show()裡
09     }
10
11     public static void show(int i)      // 定義 show(),可接收整數變數
12     {
13        System.out.println("value= "+i);
14     }
15
16     public static void show(int arr[]) // 定義 show(),可接收整數陣列
17     {
18        System.out.print("array=");
19        for(int i=0;i<arr.length;i++)
20           System.out.print(" "+arr[i]);
21        System.out.println();
22     }
23  }
```

```
/* app7_10 OUTPUT---

value= 5
array= 1 2 3 4
----------------------*/
```

多載會根據 method 的引數來判別哪一個 method 會被呼叫，而不是根據 method 的傳回值。

某個 method 的定義如下：

```
int func(int a, int b)        // 傳回值型態為 int 的method
{
   ....
}
```

這個函數原型會與下面 method 的定義相衝突而產生錯誤：

```
long func(int a, int b)       // 傳回值型態為 long 的method
{
   ....
}
```

app7_11 是利用引數個數的不同來多載 method 的範例。

```
01  // app7_11, 利用引數個數的不同來多載method的範例
02  public class app7_11
03  {
04     public static void main(String args[])
05     {
06        star();          // 呼叫 11~14 行所定義的star() method
07        star(7);         // 呼叫 16~21 行所定義的star() method
08        star('@',9);     // 呼叫 23~28 行所定義的star() method
09     }
10
11     public static void star()    // 沒有引數的star() method
12     {
13        star(5);  // 呼叫 16~21 行所定義的star()，並傳入整數 5
14     }
15
16     public static void star(int n)  // 有一個引數的star() method
17     {
18        for(int i=0;i<n;i++)
19           System.out.print("*");
20        System.out.println();
21     }
22
```


```
23      public static void star(char ch, int n) // 有兩個引數的 star() method
24      {
25         for(int i=0;i<n;i++)
26            System.out.print(ch);
27         System.out.println();
28      }
29   }
```

```
/* app7_11 OUTPUT---

*****
*******
@@@@@@@@@

----------------------*/
```

-The End-