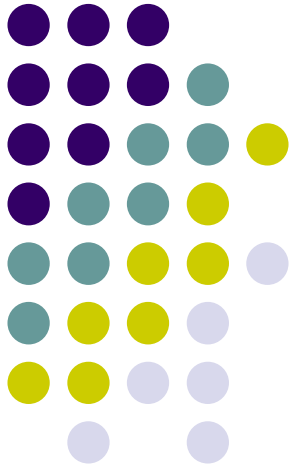
第七章
函 數



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

method的認識

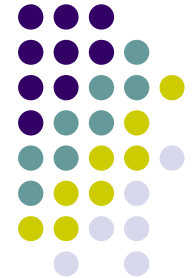
- Java把函數稱為method
- method可用如下的語法來定義：

定義method

```
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

若method沒有傳回值，return敘述可以省略



簡單的範例 (1/2)

- 簡單的method實例：

```
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

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

簡單的範例 (2/2)

- star() method被呼叫與執行的流程：

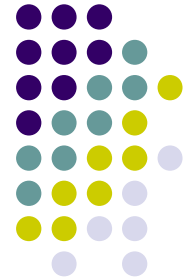
```
public class app7_1 {
    public static void main(String args[])
    {
        ...
        star();
        ...
        star();
        ...
    }
}
```

```
public static void star()
{
    ...
}
```

沒有傳回值，要在method名稱前加上void關鍵字

沒有傳遞引數，因此呼叫method時，括號內保留空白

也可以填上void關鍵字，如
public static void star (void)
{ … }



method的引數與傳回值 (1/2)

- app7_2是method使用的另一個範例：

```
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(9);    // 傳入 9 給 star()，並以 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----
******************
18 stars printed
--------------------*/
```

傳回值型態為整數

傳入的引數為整數，引數名稱為n

method的引數與傳回值 (2/2)

- 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(8,4);  // 傳入 8 與 4 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 = 8.94427190999916
        --------------------------*/
```

引數的傳遞 (1/2)

- 變數傳遞到method是以「傳值」的方式進行

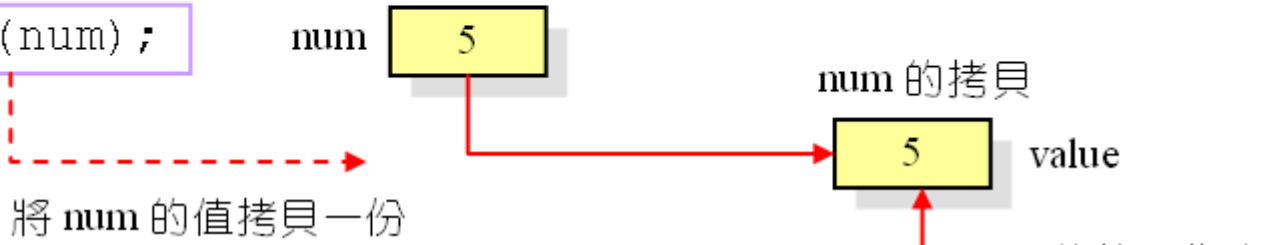
```
01  // app7_4, method 傳值的範例
02  public class app7_4
03  {
04     public static void main(String args[])
05     {
06        int num=5;
07        add10(num);            // 呼叫 add10(),並傳遞 num
08        System.out.println("in main(), num = "+num);
09     }
10
11     public static void add10(int value)
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
----------------------*/
```

引數的傳遞 (2/2)

- app7_4中，value與num的值之變化如下圖：

```
public static void main(String args[])
{
    int num=5;
    add10(num);
    ...
}
```




```
public static void add10(int value)
{
    value=value+10;
    ...
}
```

此行敘述只會更改到 num 的拷貝，而不會更改到 main() 裡的 num 變數

傳遞一維陣列

- app7_5是傳遞一維陣列到method 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[]={9,14,6,18,2,10};  // 宣告一維陣列 score
07        largest(score);        // 將一維陣列 score 傳入 largest() method
08     }
09
10
11     public static void largest(int arr[])
12     {
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 = 18
-------------------*/
```

將陣列score傳入largest() method裡

接收一維的整數陣列

傳遞二維陣列

- app7_6是傳遞二維陣列的練習

```
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[][]={{18,32,65,27,30},{17,56,12,66}}; // 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  }
```

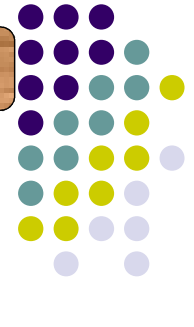
```
/* app7_6 OUTPUT---

18 32 65 27 30
17 56 12 66
------------------*/
```

將二維陣列A傳入print_mat() method裡

接收二維的整數陣列

傳回二維陣列的method



```
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     }
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  }
```

傳回二維的整數陣列

```
/* app7_7 OUTPUT---
28 42 75 37 40
27 66 22 76
-------------------*/
```

陣列的傳遞機制

- 傳遞數值時是以傳值(pass by value)的方式進行
  - 變數以「傳值」的方式傳遞到method時，在method裡更改變數的內容並不會影響到原先的變數

- 傳遞陣列時是以傳參照(pass by reference)的方式進行
  - 變數以「傳參照」的方式傳遞時，傳遞的是變數的參考位址。若是在method裡更動變數的內容，原先變數也會隨之更改

「傳參照」的範例(1/2)

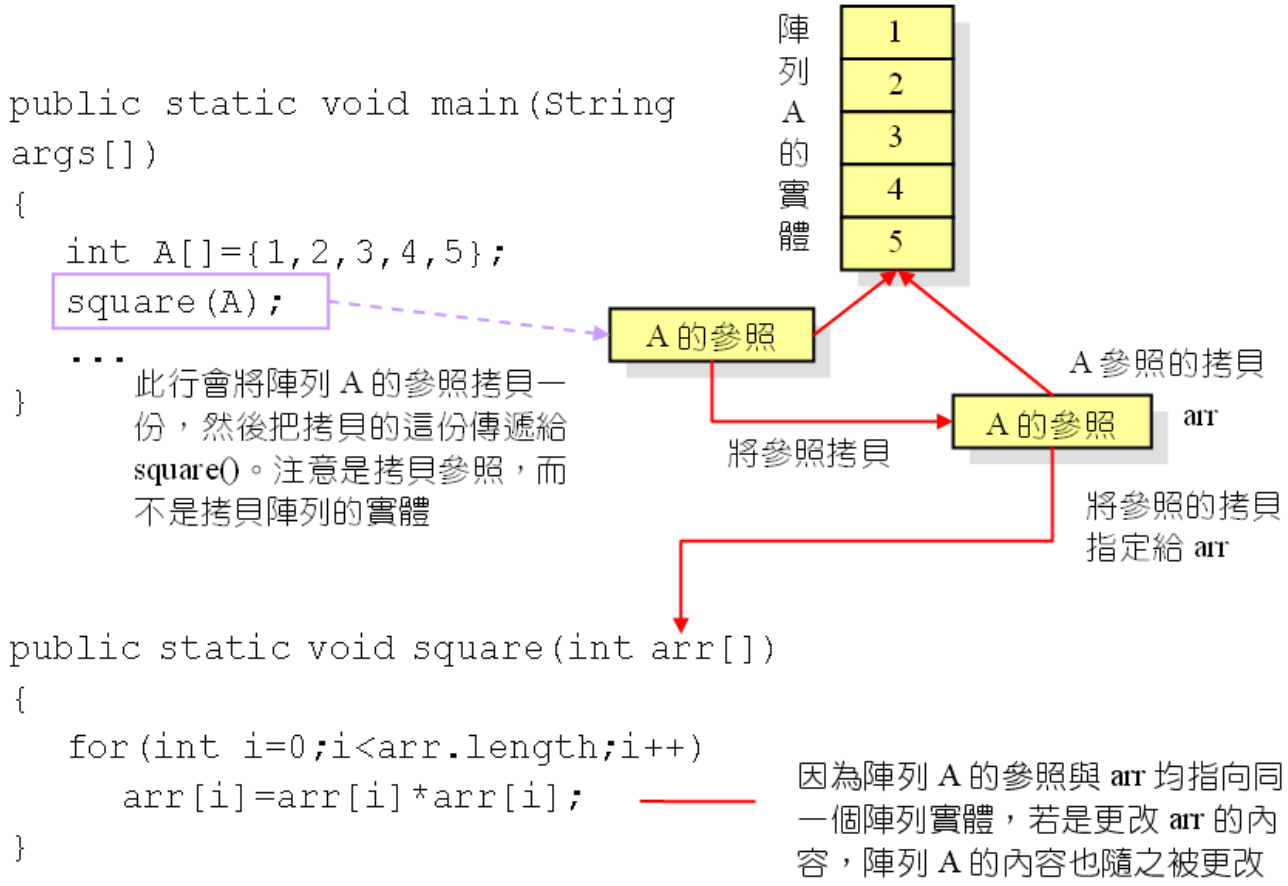
```
01  // app7_8,「傳參照」的範例
02  public class app7_8
03  {
04     public static void main(String args[])
05     {
06        int A[]={1,2,3,4,5};
07
08        square(A);                          // 呼叫 square()，並傳遞陣列A
09
10        System.out.println("呼叫 square() method 之後...");
11
12        for(int i=0;i<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  }
```

此例說明「傳參照」和「傳值」的不同

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

「傳參照」的範例(2/2)

- square() method的呼叫過程



遞迴的認識

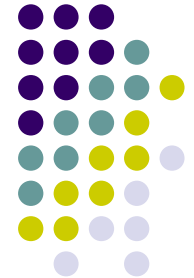
- 遞迴就是method本身呼叫自己
- 階乘函數（factorial function，$n!$）可利用遞迴的方式完成

$$\text{fac}(n)=\begin{cases}1\times 2\times\cdots\times n; & n\geq 1\\ 1; & n=0\end{cases}$$

（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）



遞迴的使用 (1/2)

- 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  }
```

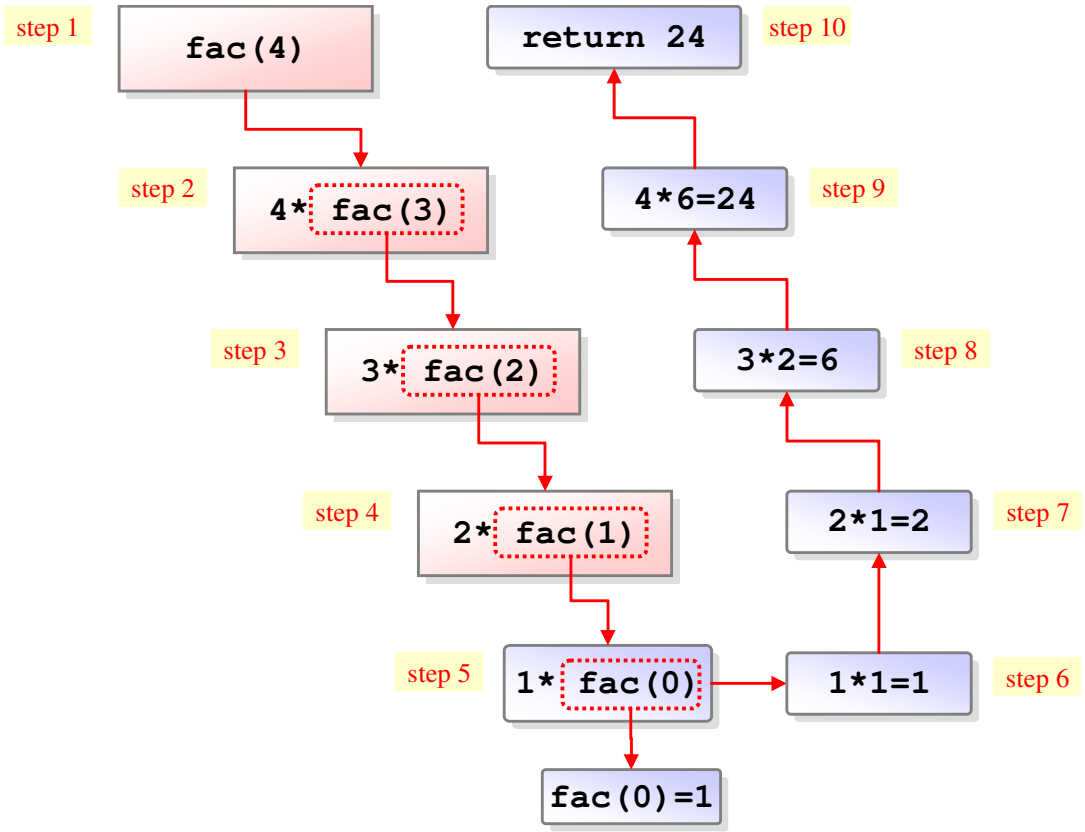
```
/* app7_9 OUTPUT----
1*2*...*4=24
--------------------*/
```

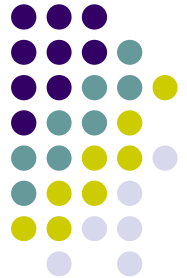
$$\text{fac}(n)=\begin{cases} n\times\text{fac}(n-1); & n\geq 1 \\ 1; & n=0 \end{cases}$$

遞迴的使用 (2/2)

- fac(4) 的遞迴計算過程

$$\text{fac}(n)=\begin{cases} n\times\text{fac}(n-1); & n\geq 1 \\ 1; & n=0 \end{cases}$$

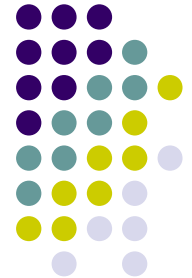




多載的概念

- 多載（overloading）：
  - 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

- 日常生活中也有許多「多載」的應用，如
  - 手機 (可以有打電話、照像、錄音等功能)
  - 冷氣 (可以有冷氣、暖氣、除濕等功能)



method多載的使用

- 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

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

多載的注意事項

- 多載會根據method的引數判別哪一個method會被呼叫
  - 舉例來說，某個method的定義如下：

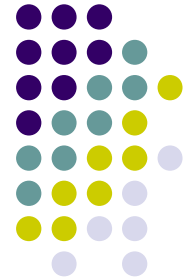
```
int func(int a, int b)          // 傳回值型態為 int 的 method
{
   ....
}
```

引數型態及個數皆相同，會讓編譯器難以分辨到底該使用哪一個method

它會與下面這個method的定義相衝突：

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

使用method的多載



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