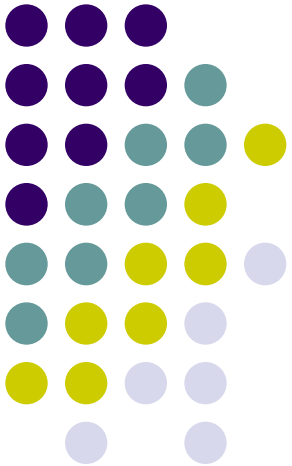
第七章
函 數

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



函數的認識

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

語法	說明
public static 傳回值型別 函數名稱(型別 引數,…){	定義函數
程式敘述 ; return 傳回值;	函數的主體
}	

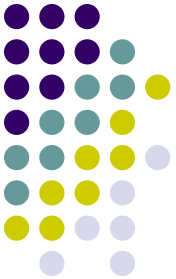
若函數沒有傳回值，return敘述可以省略

```
public class Ch7_1{    // 定義 public 類別 Ch7_1
  public static void main(String[] args){
    // main()本體
  }
  public static void star(){
    // star()本體
  }
}
```

函數存在於 class 之內

整個 public class Ch7_1 的範圍

函數要放在class裡面，無法單獨存在



簡單的範例 (1/2)

- 簡單的函數實例：

```
01  // Ch7_1, 簡單的範例
02  public class Ch7_1{
03    public static void main(String[] args){
04       star();   // 呼叫 star() 函數
05       System.out.println("Wonderful tonight");
06       star();   // 呼叫 star() 函數
07    }
08
09    public static void star(){    // star() 函數
10       for(int i=0;i<18;i++)
11          System.out.print("*");  // 印出 18 個星號
12       System.out.print("\n");    // 換行
13    }
14  }
```

main() 函數

star() 函數

- 執行結果：

```
******************
Wonderful tonight
******************
```

簡單的範例 (2/2)

- star() 被呼叫與執行的流程：

```
// Ch7_1, 簡單的範例
public class Ch7_1{
  public static void main(String[] args){
     star();   // 呼叫 star() 函數
     System.out.println("Wonderful tonight");
     star();   // 呼叫 star() 函數
  }
```

main() 函數

```
public static void star(){
   for(int i=0;i<18;i++)
      System.out.print("*");
   System.out.print("\n");
}
```

star() 函數

①②③④

沒有傳回值，要在函數名稱前加上void關鍵字

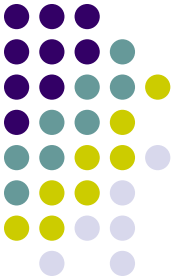
沒有傳遞引數，因此呼叫函數時，括號內保留空白

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

括號內不需填入任何文字

沒有傳回值

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



函數的引數與傳回值 (1/3)

- Ch7_2是函數使用的另一個範例：

```
01  // Ch7_2, 計算 1+2+..+end 的平均
02  public class Ch7_2{
03      public static void main(String[] args){
04          double avg=average(4);              // 呼叫 average()函數
05          System.out.printf("avg=%6.2f",avg);
06      }
07
08      public static double average(int end){  // 定義 average()函數
09          int sum=0;
10          double avg;
11          for(int i=1;i<=end;i++)
12              sum+=i;
13          avg=(double)sum/end;
14          return avg;                 // 傳回平均，其型別為 double
15      }
16  }
```

傳回值型態為double

傳入的引數型態為int，名稱為end

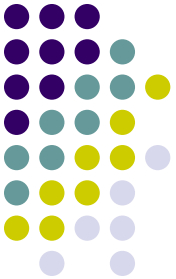
avg的傳回型態為double

- 執行結果：

```
avg=  2.50
```

函數的引數與傳回值 (2/3)

- VSCode的內嵌提示：



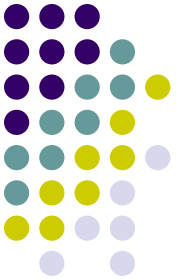
函數的引數與傳回值 (3/3)

- Ch7_3是一個計算長方形面積的範例：

```
01  // Ch7_3, 計算長方形的面積
02  public class Ch7_3{
03     public static void main(String[] args){
04        int rec_area;
05        rec_area=area(8,4);  // 傳入 8 與 4 兩個引數到 area()裡
06        System.out.println("area= "+rec_area);
07     }
08
09     public static int area(int width, int height){
10        return width*height;  // 傳回長方形面積
11     }
12  }
```

• 執行結果：

```
area= 32
```



引數的傳遞 (1/2)

- 變數傳遞到函數是以「傳值」的方式進行

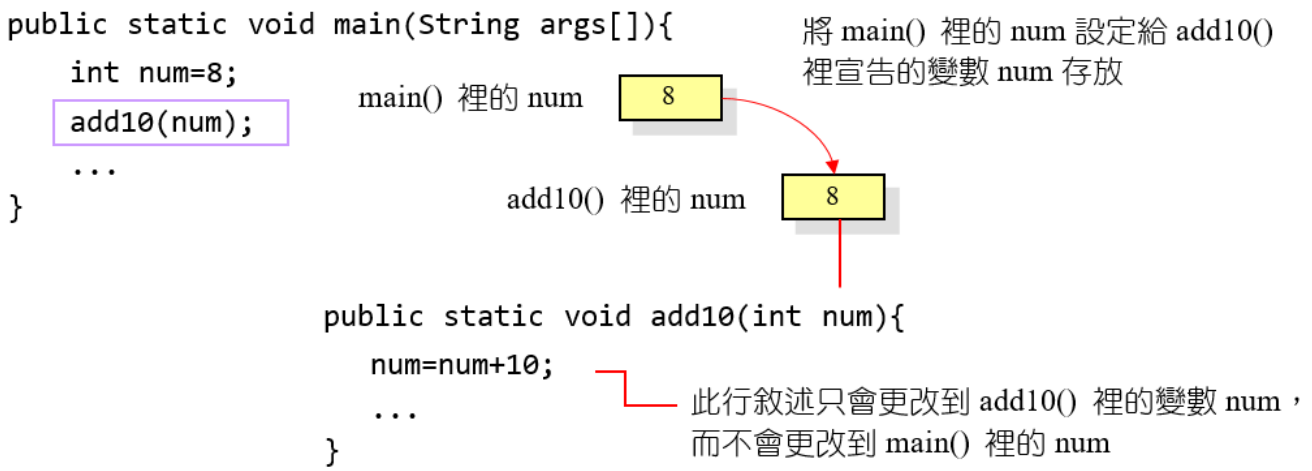
```
01  // Ch7_4, 函數傳值的範例
02  public class Ch7_4{
03      public static void main(String[] args){
04        int num=8;
05        add10(num);        // 呼叫 add10(),並傳遞 num
06        System.out.printf("in main(), num = %d\n",num);
07      }
08
09      public static void add10(int num){
10        num=num+10;        // 將 num 的值加 10 之後，設回給 num
11        System.out.printf("in add10(), num= %d\n",num);
12      }
13  }
```

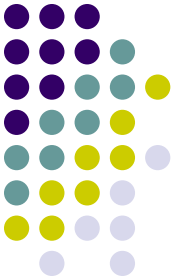
- 執行結果：

```
in add10(), num = 18
in main(), num = 8
```

引數的傳遞 (2/2)

- Ch7_4中，num值的變化如下圖：

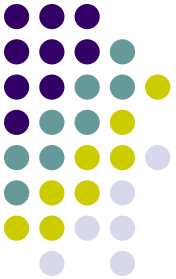




函數如何接收或傳遞陣列

- 接收或傳回陣列時，函數如何定義：

```
public static void fun(int[] a){敘述}          // 可接收一維的 int 陣列
public static void fun(double[][] a){敘述}     // 可接收二維的 double 陣列
public static int[] fun(int[] a){敘述};        // 接收與傳回的都是一維 int 陣列
public static double[][] fun(){敘述}           // 可傳回二維的 double 陣列
public static void fun(int[][][] a){敘述}      // 可接收三維的 int 陣列
```



傳遞一維陣列

- Ch7_5是傳遞一維陣列到函數的範例

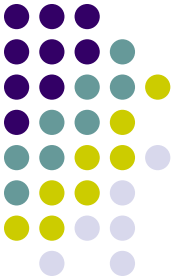
```
01  // Ch7_5, 簡單的範例
02  public class Ch7_5{
03     public static void main(String[] args){
04       int[] score={9,14,6,18,2,10};  // 宣告一維陣列 score
05       largest(score);     // 將一維陣列 score 傳入 largest() 函數
06     }
07
08     public static void largest(int[] arr){     // 利用 arr 接收傳進來的陣列
09        int max=arr[0];
10        for(int i=0;i<arr.length;i++)     // 找尋最大值
11          if(max<arr[i])
12             max=arr[i];
13        System.out.println("largest num = "+max);
14     }
15  }
```

將陣列score傳入largest() 函數裡

接收一維的整數陣列

- 執行結果：

```
largest num = 18
```



傳遞二維陣列

- Ch7_6是傳遞二維陣列的練習

```
01  // Ch7_6, 傳遞二維陣列
02  public class Ch7_6{
03      public static void main(String[] args){
04        int[][] mat={{18,32,65,27,30},{17,56,12,66}};     // 定義二維陣列
05        print_mat(mat);      // 將二維陣列 mat 傳到 print_mat()
06      }
07
08      public static void print_mat(int[][] arr){
09        for(int[] row:arr){      // 走訪陣列的內容
10           for(int e: row)
11              System.out.printf("%3d",e);          // 印出陣列值
12           System.out.print("\n");
13        }
14      }
15  }
```

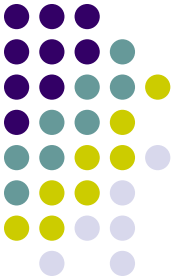
將二維陣列mat傳入print_mat() 函數裡

接收二維的整數陣列

- 執行結果：

```
 18  32  65  27  30
 17  56  12  66
```

傳回一維陣列的函數(1/2)



```
01  // Ch7_7, 傳回一維陣列的函數
02  public class Ch7_7{
03     public static void main(String[] args){
04        int[] a1={18,32,65,27,30}; // 宣告一維陣列 a1 並設定初值
05        int[] a2;                  // 宣告一維陣列 a2
06        a2=add10(a1);      // 呼叫 add10()，並把傳回的值設給陣列 a2
07        for(int e:a2)      // 印出陣列的內容
08              System.out.printf("%3d",e);
09        System.out.println();
10     }
11
12     public static int[] add10(int[] b1){
13        int[] b2=new int[b1.length];    // 宣告並配置一維陣列 b2
14        for(int i=0;i<b1.length;i++)
15              b2[i]=b1[i]+10;           // 將陣列 b2 的元素加 10
16         return b2;                     // 傳回陣列 b2
17     }
18  }
```

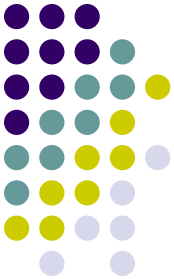
傳回一維的整數陣列

傳回一維的整數陣列

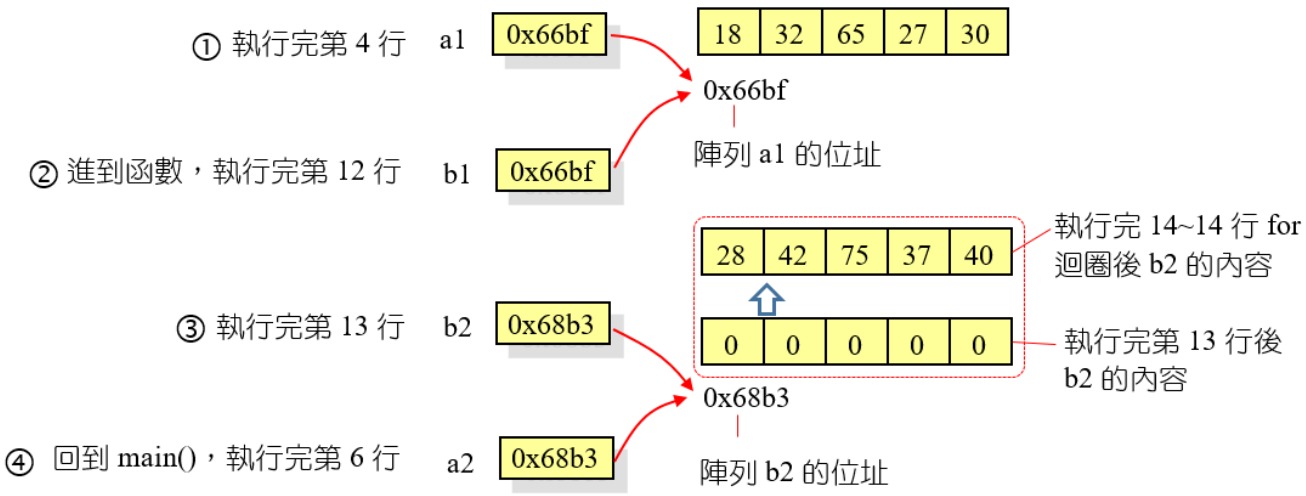
- 執行結果：

```
28 42 75 37 40
```

傳回一維陣列的函數(2/2)



- Ch7_7中，陣列之間的關係：

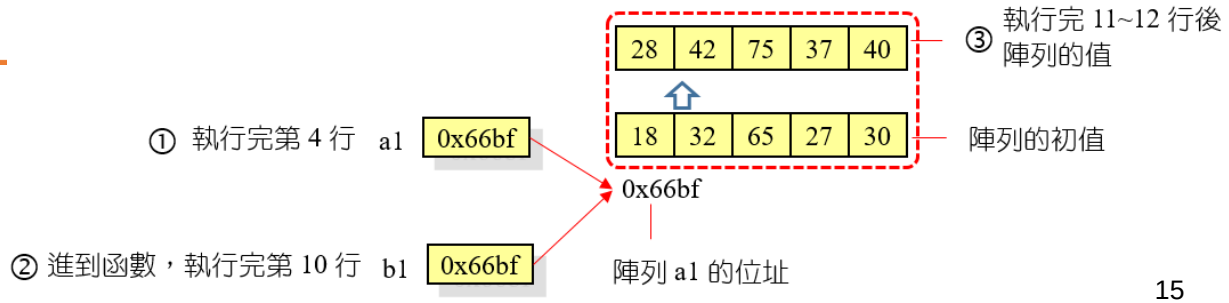


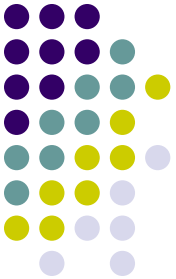
直接修改陣列裡的元素

```
01  // Ch7_8,直接修改陣列裡的元素
02  public class Ch7_8{
03     public static void main(String[] args){
04        int[] a1={18,32,65,27,30}; // 宣告一維陣列 a1
05        add10(a1);         // 呼叫 add10()
06        for(int e:a1)      // 印出陣列的內容
07              System.out.printf("%3d",e);
08     }
09
10     public static void add10(int[] b1){
11        for(int i=0;i<b1.length;i++)
12              b1[i]=b1[i]+10;          // 將陣列元素加 10
13     }
14  }
```

- 執行結果：

```
28  42  75  37  40
```

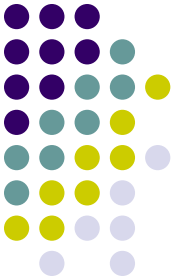




陣列的傳遞機制

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

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



遞迴的認識

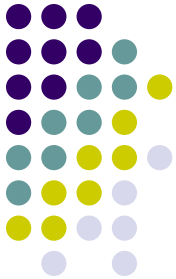
- 遞迴就是函數本身呼叫自己
- 階乘函數（factorial function，$n!$）可利用遞迴的方式完成

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

（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函數

```
01  // Ch7_9, 簡單的遞迴函數
02  public class Ch7_9{
03     public static void main(String[] args){
04       System.out.printf("1*2*...*4= %d\n",fac(4));
05     }
06
07     public static int fac(int n){       // fac() 函數
08        if(n>=1)                  // 當 n>=0 時
09           return n*fac(n-1);
10        else                      // 當 n=0 時
11           return 1;
12     }
13  }
```

- 執行結果：

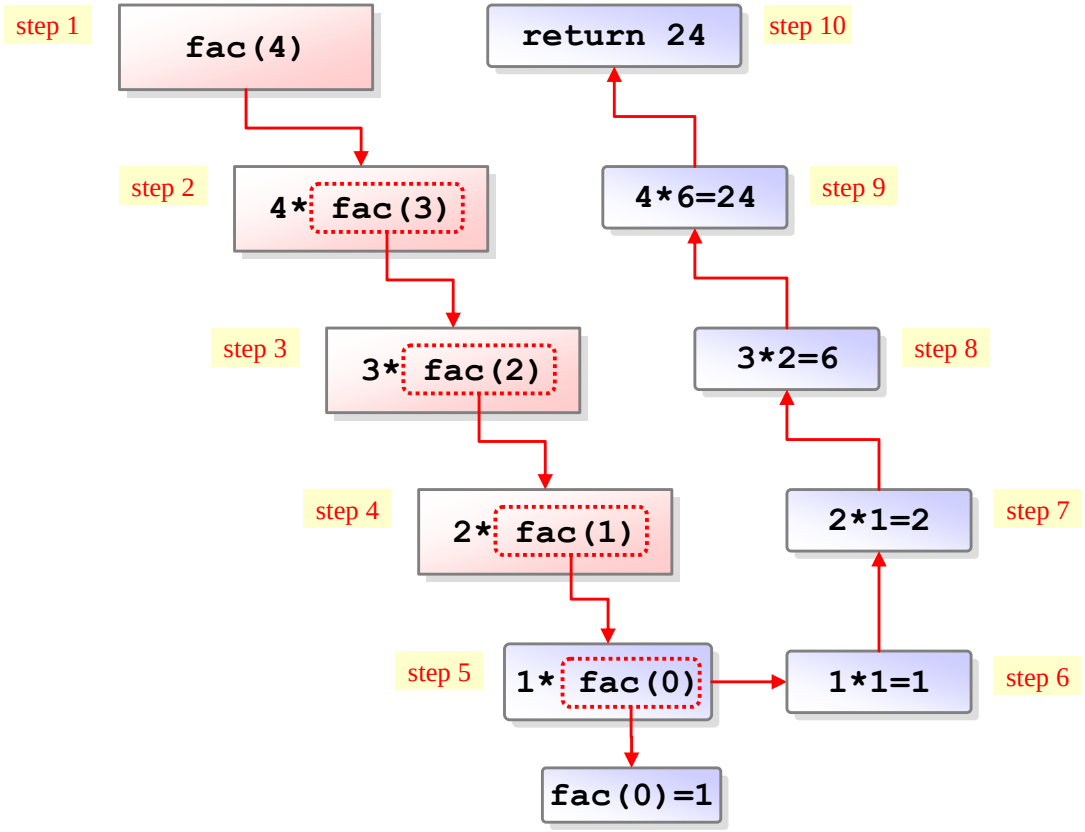
```
1*2*...*4=24
```

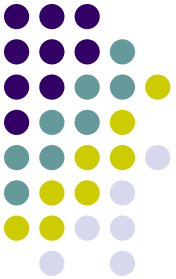
$$\text{fac}(n)=\begin{cases} n\times\text{fac}(n-1); & n\geq 1 \\ 1; & n=0 \end{cases}$$

遞迴的使用 (2/2)

- fac(4) 的遞迴計算過程

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

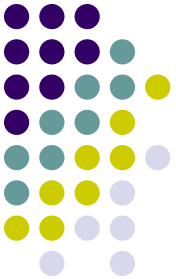




多載的概念

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

- 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)



函數多載的使用

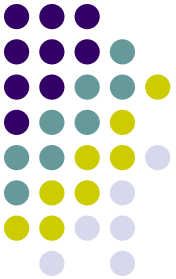
- 函數引數型態不同的多載：

```
01  // Ch7_10, 函數的多載-引數型別不同
02  public class Ch7_10{
03     public static void main(String[] args){
04        int a=5, b[]={1,2,3,4};
05        show(a);                          // 將整數 a 傳遞到 show()裡
06        show(b);                          // 將整數陣列 b 傳遞到 show()
07     }
08
09     public static void show(int i){     // 定義 show(),可接收整數變數
10        System.out.println("value= "+i);
11     }
12     public static void show(int arr[]){ // 定義 show(),可接收整數陣列
13        System.out.print("array=");
14        for(int i=0;i<arr.length;i++)
15           System.out.printf("%2d",arr[i]);
16     }
17  }
```

引數型態不同

- 執行結果：

```
value= 5
array= 1 2 3 4
```



多載的注意事項

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

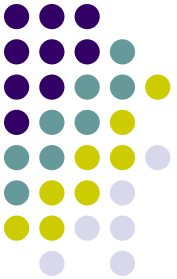
```
int func(int a, int b){          // 傳回值型別為 int 的函數
    ....
}
```

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函數

它會與下面這個函數的定義相衝突：

```
long func(int a, int b){         // 傳回值型別為 long 的函數
    ....
}
```

使用函數的多載



```
01  // Ch7_11, 利用引數個數的不同來多載函數的範例
02  public class Ch7_11{
03      public static void main(String[] args){
04         star();            // 呼叫 9~11 行的 star()函數
05         star(7);           // 呼叫 13~15 行的 star()函數
06         star('@',9);       // 呼叫 17~21 行的 star()函數
07      }
08
09      public static void star(){    // 沒有引數的 star()函數
10        star(5);            // 呼叫 13~15 行的 star()，並傳入整數 5
11      }
12
13      public static void star(int n){              // 有一個引數的 star()函數
14          star('*',n);     // 呼叫 17~21 行的 star()，並傳入'*'和 n
15      }
16
17      public static void star(char ch,int n){     // 有兩個引數的 star()函數
18         for(int i=0;i<n;i++)
19            System.out.print(ch);
20         System.out.println();
21      }
22  }
```

引數個數不同

- 執行結果：

```
*****
*******
@@@@@@@@@
```


-The End-