第七章
函 數

1

7.1 認識Java的函數

補充

何謂Method

- 簡化主程式的結構
- 精簡重複的流程
  - 在程式設計時，常會遇到某些**程式片段需要在同一個類別或不同的類別重覆出現許多次**
  - 如果這些程式片段都分別在每個地方寫一次，非常浪費時間
  - 且程式變得**冗長而不易閱讀**
  - 萬一要調整此程式片段的部份功能，要**至不同的地方修改**，造成程式維護困難，此時可透過method解決以上困難。
  - method的使用方式為
    - 將此常用的程式片段賦予一個**名稱**，此稱為方法名稱，
    - 寫在某一類別的最前面或最後面
    - 當程式設計者需要使用此程式片段時，只要鍵入此方法名稱，即可執行此程式片段。

3

7.1 函數的基本概念

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

4

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

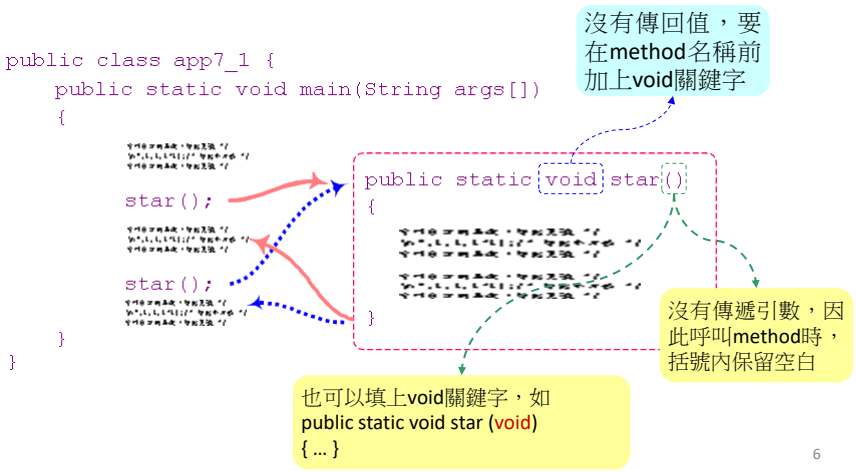
star() method

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

5

簡單的範例 (2/2)

- star() method被呼叫與執行的流程：



6

7.1 課堂練習

- **ex7_1_1.java**
  試撰寫void kitty() method，當呼叫kitty() 時，螢幕上會顯示出 “Hello Kitty” 之字串。

- Hint
  - 由主程式呼叫一個kitty()的方法
  - 撰寫一個kitty()的方法 裡面可以列印出Hello Kitty字串

```
/* output----
Hello Kitty
----------*/
```

7

補充

Calling Program

```
public static void main(String args[])
{
 float radius=10.0f,area;
 area =compute_area( radius , 3.14f );
}
```

Called Program

```
public static float compute_area(float r ,float pi )
{
 return r * r * pi ;
}
```

補充

Method語法說明

- **public static float compute_area(float r, float pi)**
  1. 封裝等級我們將於下一章之後開始介紹，此處使用public即可。
  2. 由於主函式被宣告為static，而static method內的程式碼只能呼叫static method，因此為了要讓主函式main呼叫，故必須宣告為static method。
  3. 若函式沒有回傳值，此時可將回傳值型態宣告為void。
     - 具有回傳值的函式，在函式主體內應該包含一個以上的return敘述，以便傳回資料。不具回傳值的函式則可以沒有return敘述。
     - 具有回傳值的函式，其return後面應該接上與回傳值相容資料型態的常數、變數或運算式。
  4. method名稱小括號內則是引數群
     - 每宣告一個輸入引數，都必須清楚地宣告該引數的資料型態及名稱。
     - 若無引數需要傳遞，則引數列可以為空白，但( )不可省略。

7.1 函數的基本概念

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(8);        // 傳入8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 }
```

傳回值型態為整數

傳入的引數為整數，引數名稱為n

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

10

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

sqrt()→開根號的method

11

7.1 課堂練習

- ex7_1_2.java
  試撰寫int cub(int x) method，其作用為傳回引數x的3 次方。
  **請用Scanner輸入X

```
/* output------------------
  請輸入一個整數：2
  2 的3 次方=8
----------------------------*/
```

12

引數的傳遞 (1/2)

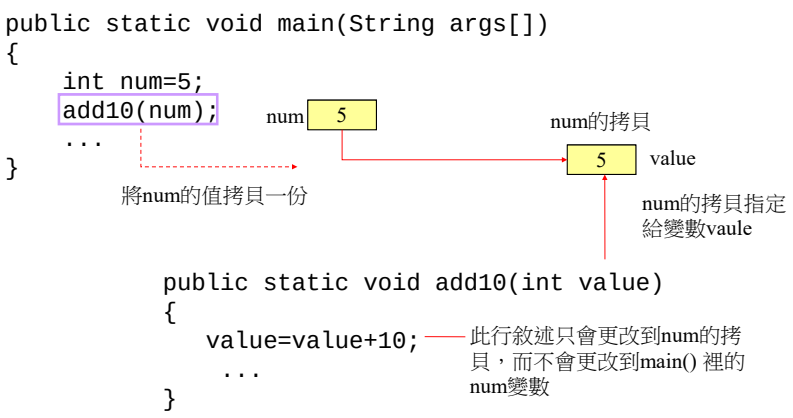
- 變數傳遞到method是以「傳值」的方式進行

```
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：



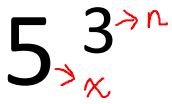
7.1 課堂練習

- ex7_1_3.java
  試撰寫int power(int x, int n) method，用來計算x 的 n 次方。

  **請用Scanner輸入x 與 n

```
/* output----
5 的3 次方=125
----------*/
```

15



int power(int x,int n)

```
int p=1;
for(int i=0;i<n;i++)
   p=p*x;
```

7.1 課堂練習

- ex7_1_4.java

設計一個函式void print_total_mul(int a,int b)，列印ab乘法表，例如a=5、b=8時，輸出如下（當a=8、b=5時，輸出也相同）：（當a=1、b=9時，恰恰為九九乘法表）

執行結果：

5*5=255*6=305*7=355*8=40
6*5=306*6=366*7=426*8=48
7*5=357*6=427*7=497*8=56
8*5=408*6=488*7=568*8=64

5*5=255*6=305*7=355*8=40
6*5=306*6=366*7=426*8=48
7*5=357*6=427*7=497*8=56
8*5=408*6=488*7=568*8=64

```
public class ex7_1_4
{
   public static void main(String args[])
   {
      print_total_mul(5,8);
      System.out.println();
      print_total_mul(8,5);
   }
```

7.2 學習函數引數傳遞的方式

傳遞一維陣列

- app7_5是傳遞一維陣列的練習

```
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
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  }
```

將陣列score傳入largest() method裡

接收一維的整數陣列

```
/* app7_5 OUTPUT---
largest num = 12
----------*/
```


傳遞二維陣列

- app7_6是傳遞二維陣列到method 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  }
```

```
/* app7_5 OUTPUT---
51 38 82 12 34
72 64 19 31
----------*/
```

將陣列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    }
```

傳回二維的整數陣列

```
/* app7_7 OUTPUT--
61 48 92 22 44
82 74 29 41
----------*/
```


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)

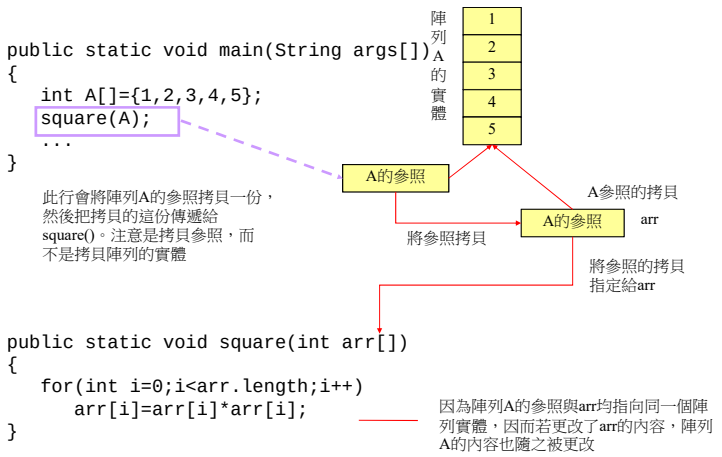
```
01 // app7_8,「傳參照」的範例
02 public class app7_8
03 {
04   public static void main(String args[])
05   {
06     int A[]={1,2,3,4,5};
07
08     square(A);  // 呼叫 square()，並傳遞陣列A
09
10     System.out.println("呼叫square() method之後...");
11
12     for(int i=0;i<A.length;i++)    // 印出陣列的內容
13       System.out.print(A[i]+"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



7.2 課堂練習(1/2)

- ex7_2_1.java
  試撰寫可傳回一維陣列最小值的**method**，並測試之。
- 此陣列為：{75,29,38,45,16}

```
/* output--------
陣列a 最小值為 16
---------------*/
```

25

7.2 課堂練習(2/2)

- ex7_2_2.java
  試寫一**method**，可傳回一維陣列中最大值與最小值的乘積。
  此陣列為：{75,29,10,38,45,16}

```
/* output----------------------
陣列a 最大值與最小值的乘積為 750
----------------------------*/
```

26

7.3 學習遞迴函數的撰寫

7.3 遞迴(recursive)的使用原則

- 遞迴的使用必須同時滿足以下兩個條件，否則程式會沒完沒了，終至堆疊用盡而使程式當掉。
- 使用遞迴解題的兩個條件如下：
  - 1.問題須**具有重複的特性**，也就是在某些特性之下，可以繼續呼叫自己。
  - 2.重複的**問題須有結束的條件**。如階乘的結束條件為n=1時1!=1，費式數列的結束條件為n≦2時f(2)=1，f(1)=1。
- 常用於有規則性的程式設計
  - 求最大公因素、排列、組合、階乘、費氏函數

28

階乘

- 階乘的定義是求小於等於某一數至1的連續整數之積。本例重複的特性為每次減1，即可自己呼叫自己。例如，

  6!=6*5!

  5!=5*4!

- 本例結束遞迴的條件是**當遞減至1時，結束遞迴**。所以，其演算法如下：

```
public static int fac(int n)
 {
   if (n= =1)
     return n;
   else
     return n*fac(n-1); //自己再乘以下一項
 }
```

7.3 遞迴

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）

30

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  }
```

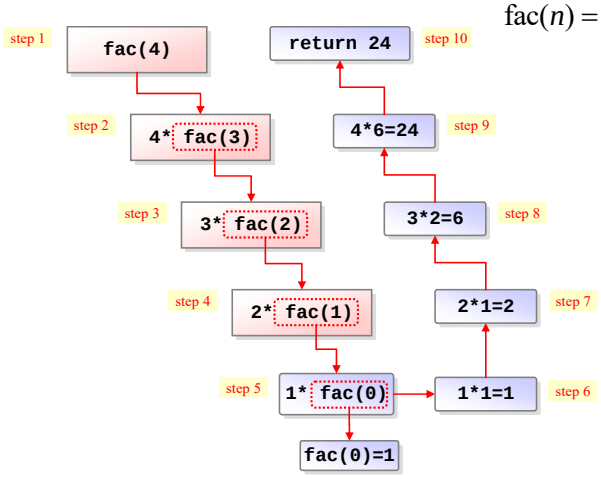
```
/* app7_9 OUTPUT--
1*2*...*4=24
--------------------*/
```

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

遞迴的使用 (2/2)

- fac(4) 的遞迴計算過程

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



以for來完成5階乘計算

```
public class test{
  public static void main(String argv[])
    {
        fac(5);
    }

 public static void fac(int n)
    {
      int product=1,j=0;
      for(int i=n;i>0;i--)
        {
          j++;
          product=product*i;
        }
      System.out.println("您要計算"+j+"!="+product);
    }
}
```

您要計算5!=120

33

範例說明

- 題目：試撰寫遞迴函數int rsum(int n)
- 求算 1x2＋ 2x3＋ 3x4＋…+(n-1)x(n) *之和*
- *n 值請設*5。

推理

if n=1→ 0x 1

if n=2→ 0x 1 ＋ 1x 2

if n=3→ 0x 1 ＋ 1x 2 ＋2x 3

if n=4→ 0x 1 ＋ 1x 2 ＋2x 3 ＋3x 4

if **n=5→** 0x 1 ＋ 1x 2 ＋2x 3 ＋3x 4 + **4x 5**

```
public class ex
{
  public static void main(String args[])
  {
    int n=5;

    System.out.print("1*2+2*3+3*4+...+("+n+"-
      1)*"+n+"=");
    System.out.println(rsum(n));
  }
  public static int rsum(int n)
  {
    if(n==1)
      return 0;
    else
      return (rsum(n-1)+n*(n-1));
  }}

/* output----------
1*2+2*3+3*4+...+(5-1)*5=40
----------*/
```

34

7.3 課堂練習(1/2)

- ex7_3_1.java
  - 試以遞迴的方式撰寫函數**int sum(int n)**
  - 利用遞迴公式**sum(n)=n+sum(n-1)， sum(1)=1**
  - 用來計算 1+2 + 3 + ... + *n*的值，**n** 值請設值為**10**。

```
/* output------
1+2+...+10=55
------------*/
```

35

7.3 課堂練習(2/2)

- ex7_3_2.java
  - 試利用下面的公式
  - **sum(n)=sum(n-1)+2*n， sum(1)=2**
  - 撰寫遞迴函數**int sum(int n)**
  - 用來計算 0 + 2 + 4 + 6 + ... + 2**n* 之和，**n** 值請設值為**5**。

```
/* output--------
0+2+4+6+...2*4+2*5=30
----------------*/
```

36

補充

迴圈可做，為什麼需要遞迴？

- 遞迴(recurrence)做出多重迴圈(multiple loop)
- 遞迴具有重複執行的特性，而可以使用遞迴求解的程式，實際上也可以使用迴圈來求解。
- 那麼使用遞迴好還是使用迴圈求解好？
  - 這並沒有一定的答案。
  - 不過通常由於遞迴本身有重複執行與記憶體堆疊的特性，所以若在求解時需要使用到堆疊特性的資料結構時，使用遞迴在設計時的邏輯會比較容易理解，程式碼設計出來也會比較簡潔
  - 然而遞迴會有函式呼叫的負擔，因而有時會比使用迴圈求解時來得沒有效率
  - 不過迴圈求解時若使用到堆疊時，通常在程式碼上會比較複雜。

37

7.4 認識函數的多載

7.4 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、照相、錄音等功能)
  - 冷氣 (可以有冷氣、暖氣、除濕等功能)

39

方法多載 (Overloading Methods)

- 在方法的設計裏，有時會有兩種以上的方法功能接近
  - 例如只是參數型別或參數個數不同，是否可以使用相同的方法名稱，以減少程式設計者的困擾？答案是肯定的。
  - 因為物件導向的程式設計均允許程式設計者在**參數型別與參數個數不同時，使用相同的方法名稱，這些相同的方法即稱為方法多載**。
- 例如，若以實做add 方法如下：

```
int add(int c, int d);          (1)
```

亦可在實做add方法如下：

```
int  add(int c, int d, int e)      (2)
double add(int c, int d, double e)  (3)
```

40

方法多載 (Overloading Methods)

- 其次，當呼叫add方法時，編譯器集會依照參數的數量與型別，而呼叫對應的方法。

- add(3,4);即會執行以上第（1）式。
- add(3,4,5);即會執行以上第（2）式。
- add(3,4,3.4) ;即會執行以上第（3）式。

- 但是，當有相同的參數個數與型別時，絕不可有相同的傳回值。例如，絕不能在實做add 方法如下：
  double add(int c,int d);

41

7.4 method的多載

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

42

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

7.4 課堂練習(1/2)

- ex7_4_1.java
  - 試撰寫max() method 的多載，其中max 引數的型態為int，且可以有兩個或三個引數，method 的傳回值為這些引數的最大值，傳回值的型態也是int。

```
/* output-----
max(8,12)=12
max(1,5,9)=9
-------------*/
```

45

7.4 課堂練習(2/2)

- ex7_4_2.java
- 試撰寫一組可以計算三角形面積的多載化method，格式為triangle(base, height)，base與height 可同為int 或float，傳回值的型態為float。（ 三角形面積 = (base*height)/2 ）

```
/* output--------------------------------
三角形的底為6,高為3,面積為9.0
三角形的底為4.2,高為3.3,面積6.9299994
---------------------------------------*/
```

46

回家作業

hw7_1.java

找出某個整數的質數。

- 以method ()撰寫
- 質數的定義是大於一的整數，除了1跟自己本身以外，沒有其他因數。
- 若某一個整數有任何數可以整除他，則不是質數

```
請輸入整數：20
20的質數為：
2 3 5 7 11 13 17 19
```

47

回家作業

hw7_2.java

製作三個method（Odd、Even、TotalSum函式），功能分別為計算奇數和、偶數和、整數和。
(此程式為需要透過鍵盤先輸入一個整數之後，再做三次選擇以計算奇數和、偶數和、整數和之後，才能結束程式，程式測試結果如下所示)。

```
/* output----------------------------------
1+2+...+n=?請輸入n=10
請問要做奇數和(1),偶數和(2),還是整數和(3)?請選擇:1
總和為25
---------------------------------------------------------
請問要做奇數和(1),偶數和(2),還是整數和(3)?請選擇:2
總和為30
---------------------------------------------------------
請問要做奇數和(1),偶數和(2),還是整數和(3)?請選擇:3
總和為55
---------------------------------------------------------------------*/
```

回家作業

hw7_3.java

撰寫一個計算矩形的方法,可以長、寬來計算矩形面積,也可以以左上角及右下角的座標來計算矩型面積。試撰寫int triangle () method 的**多載**，其中引數的型態為傳入兩個int，或傳入四個座標值。傳回值的型態也是int。

```
/* output----------------------------------------
長寬(10,20)的矩型面積：200
座標(-5,-5)與(15,25)的矩型面積：600
-------------------------------------------------*/
```

回家作業

hw7_4.java

撰寫一個計算矩形的方法,可以長、寬來計算矩形面積,也可以以左上角及右下角的座標來計算矩型面積。試撰寫int area2() method 與int area4() method，其中引數的型態為傳入兩個int，或傳入四個座標值。傳回值的型態也是int。

```
/* output----------------------------------------
長寬(10,20)的矩型面積：200
座標(-5,-5)與(15,25)的矩型面積：600
-------------------------------------------------*/
```

回家作業

hw7_5.java

試撰寫double centigrade(double f) method，傳入華氏溫度，計算並傳回對應的攝氏溫度。其轉換公式如下：
攝氏溫度=(5*華氏溫度-160)/9

```
/* output----------
華氏-40.0 度=攝氏-40.0 度
----------------------*/
```

回家作業

hw7_6.java

使用迴圈設計一個程式，找出2~100 中所有的質數，每印出5 個質數後換行顯示，執行結果如下。

```
2    3    5    7    11
13   17   19   23   29
31   37   41   43   47
53   59   61   67   71
73   79   83   89   97
```

回家作業

hw7_7.java

請撰寫一個程式來計1!+2!+...+m!,(0<m<10)

```
計算1!+2!+...+m!,(0<m<10)
請輸入m:12
請輸入m:-5
請輸入m:5
1!+2!+...+m!=153(m=5)
```

回家作業

hw7_8.java

1.請在同一個類別中，使用覆載 (OverLoad) 撰寫三個同樣名為 add 的方法。

2.宣告 main 方法，分別提供以下 a、b、c 等參數。

```
add(2, 3);
add(5.2, 4.3);
add("I love", "Java!!");
```

3.方法一：傳入兩個整數 (int, int)，計算兩整數的和 (int)。程式執行時，列出【Adding two integers: i, j】，請呼叫 main 方法中的 add(2, 3)，將參數代入 i, j 內。

4.方法二：傳入兩個浮點數 (double, double)，計算兩個浮點數的和 (double)，程式執行時，列出【Adding two double: i, j】，請呼叫 main 方法中的 add(5.2, 4.3)，將參數代入 i, j 內。

5.方法三：傳入兩字串 (String, String)，計算合併後的兩個字串 (String)，程式執行時，列出【Adding two string: i, j】，請呼叫 main 方法中的 add("I love", " Java!!")，將參數代入 i, j 內。

6.最後依序顯示此方個方法所計算出的列印值，執行結果顯示如下。

```
Adding two integers: 2, 3
Adding two doubles: 5.2, 4.3
Adding two strings: I love,  Java!!
5 9.500000 I love Java!!
```

回家作業

hw7_9.java

請撰寫一個method(int a, int b, int c)允許輸入3個整數，並找出這3個整數之間，相差最少的數字差距。

```
請輸個入三個數字不同的整數:2 3 6
三數中差距最小為1
```

回家作業

hw7_10.java

請撰寫一個方法的程式，傳遞西元年至此方法year()，利用if else if條件指令來執行潤年計算規則，以上使用者輸入西元年來判斷是否為潤年。潤年計算的基本規則是「四年一潤，百年不潤，四百年為一潤」。

```
請輸入西元年:2018
不是潤年
```

回家作業

hw7_11.java

以method()撰寫，傳入一個整數，此method的功能為可列印出從1到100之間，所有可被7整除，又不可以被3整除的數值。

```
請輸個入1-100以內的整數:48
可被7整除又不可被3整除的數值為：7 14 28 35
```

回家作業

hw7_12.java

以method()撰寫，傳入一個整數n(n傳入50)，此method的功能可利用for迴圈計算
$1^2-2^2+3^2-4^2+...+47^2-48^2+49^2-n^2$的值。

```
請輸入整數：50
-1275
```

回家作業

hw7_13.java

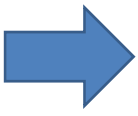
假如有一隻蝸牛爬一顆10公尺的大樹，白天往上爬2公尺，但晚上會掉下1公尺，請問要幾天才可爬到樹頂？請設計一程式，利用do-while迴圈指令來解決蝸牛爬樹問題。

```
請輸入樹高：10
蝸牛一共花了：9天爬到頂。
```

課堂練習

- hw7_14.java

請讓Output可以輸出這樣的結果



```
請輸入樹高：10
第1天的白天爬到：2.0
晚上下降成：1.0
---------------------
第2天的白天爬到：3.0
晚上下降成：2.0
---------------------
第3天的白天爬到：4.0
晚上下降成：3.0
---------------------
第4天的白天爬到：5.0
晚上下降成：4.0
---------------------
第5天的白天爬到：6.0
晚上下降成：5.0
---------------------
第6天的白天爬到：7.0
晚上下降成：6.0
---------------------
第7天的白天爬到：8.0
晚上下降成：7.0
---------------------
第8天的白天爬到：9.0
晚上下降成：8.0
---------------------
第9天的白天爬到：10.0
蝸牛一共花了：9天爬到頂。
```

回家作業

hw7_15.java

請設計一程式，組合for指令與break指令的設計，並且提供使用者輸入3次密碼的機會，並檢查密碼是否正確1。當輸入密碼正確，則顯示歡迎的訊息，正確的密碼為432，若密碼錯誤達3次，則顯示無法登入訊息。

回家作業

hw7_16.java

設$f(x)=3x^3+2x-1$，寫一函式double f(double x)，用來傳回f(x)的值，並於主程式裡分別計算f(-3.2)、f(-2.0)、f(0)與f(2.1)。