方法

人工智慧與無線感應設備開發專班

湜憶電腦知訊顧問股份有限公司

馬傳義

前言

- 在程式中若有一敘述區段，具有特定功能，而且能獨立測試執行，則可以將它編寫成一個獨立的程式單元。
- 這種程式單元在一般語言稱之為「副程式」（Subprogram），在C語言稱之為「函式」（Function），在Java語言稱之為「方法」（Method）。
- Java提供有很多種類別，如Math類別、String類別、Date類別 … 等，在這些類別中擁有很多實用方法。

前言

- 這些方法皆能提供某特定功能，如能好好使用，可以節省程式設計者研發程式時間。

方法

- 方法（Method）是類別（Class）中一個成員。
- 在前面敘述的問題，都比較簡單，撰寫的程式不會太長，所有敘述放在 main() 方法中就足以應付需求，程式的維護與可讀性上不會造成太大的困擾。
- 可是在大型軟體中，問題會變的很複雜，程式會變成很龐大，不適宜將所有敘述皆放在 main() 方法中。
- 同時一個程式設計師要撰寫與維護大型程式並不容易，目前都是以團隊合作的方式研發軟體，如沒有詳細規劃與分工合作是無法達成目標，而方法（Method）正是針對這樣的需求被大量應用。

方法

- 將「方法」適當的分類編輯，可以達到下列優點：

1. 方法中的結構與邏輯可以設計成比較簡單明確易讀，而且除錯容易。
2. 方法可以重複被使用，主程式（main()）也可著重於系統架構，程式較精簡。
3. 可以採用分工合作方式來撰寫「方法」，個別測試，不但能加快軟體研發時間，集各家的精華使軟體達到完美的境界。

方法

- 語法：

```
[修飾子][static]傳回值資料型別 方法名稱([引數串列])[throws 例外名稱]
{
        ⋮
    敘述區段;
        ⋮
    return 運算式;
}
```

方法

- 說明：
  - 修飾子
    - 有『public』、『private』、『protected』、『default』四種存取權限等級。
    - 修飾子可省略。
    - 若省略修飾子時，預設使用『default』修飾子。
  - static
    - Java的方法不能以單獨形式存在，需屬於某一類別。
    - 一般呼叫時，前面需加上類別物件名稱。
    - 若在方法前面加static（即靜態方法），則在同一類別中可以直接呼叫使用。
  - 傳回值資料型別
    - 若有需要將方法的處理結果回傳時，回傳值的資料型別。
    - 須與運算式結果（即 要回傳的值）的資料型別相同。

方法

- 方法名稱：
  - 該方法的名稱。
  - 方法名稱須符合命名規則。
- 引數串列：
  - 引數（或稱為參數）可以省略，若有兩個以上的引數，中間以逗號「,」分隔。
  - 在方法中的引數屬「虛引數」，虛引數可以是變數、物件或陣列，虛引數的資料型別與呼叫方法中的「實引數」資料型別必須一致 。
- throws 例外名稱：
  - 用來宣告某個方法可能會拋出哪些例外，並非所有的方法都會用到。
- return 運算式：
  - 若方法有傳回值，即需執行『return 運算式』。
  - 若沒有傳回值時，可以沒有return敘述，但方法名稱前面的傳回值資料型別要用「void」。

呼叫Java方法的語法

- 語法1：

  [類別名稱].方法名稱([引數串列]);

- 語法2：

  變數=[類別名稱].方法名稱([引數串列]);

  - 說明：
    - 類別名稱：
      - 被呼叫的類別名稱。
      - 若呼叫的是static的方法，則在同一類別中呼叫時，可以直接呼叫該靜態方法，不需要再加上類別名稱。

呼叫Java方法的語法

- 方法名稱：
  - 被呼叫的方法名稱。
- 引數串列：
  - 即「實引數」。
  - 須與「虛引數」對應的引數個數及資料型別相同。
  - 可與「虛引數」的名稱不同。
- 變數：
  - 若呼叫的方法有傳回值，則須要一個變數來承接回傳的值。

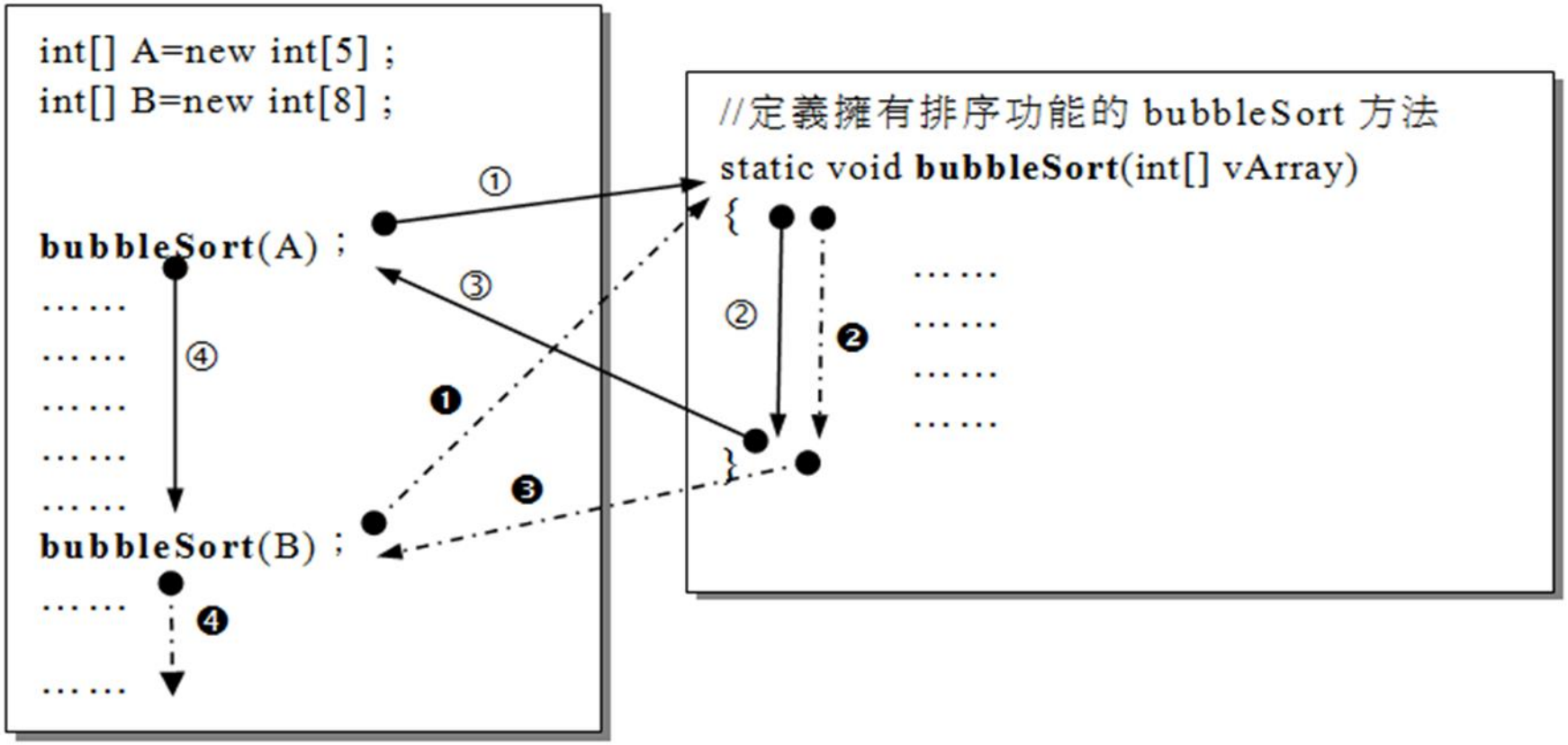
注意：

- 若呼叫的方法沒有傳回值，使用語法1；若有傳回值，則使用語法2。
- 若沒有傳回值，則在定義方法時，傳回值資料型別須為void。

呼叫Java方法的語法

- 被呼叫方法括號內的引數稱為「虛引數」，而呼叫方法括號內的引數稱為「實引數」。
- 「實引數」可以是變數、物件或陣列，也可以是一般資料與運算式。

呼叫Java方法的語法



呼叫方法時，程式執行的順序

呼叫Java方法的語法

- 程式：

```
public class CH05_01
{
    static double Money(float p, int n)
    {
        double m;
        m = p * n;
        return m;
    }

    public static void main(String[] arge)
    {
        float price = 2.3f;
        double tot;
        tot = Money(price, 11);
        System.out.println("tot = " + tot);
    }
}
```

呼叫Java方法的語法

```
1  public class CH05_01
2  {
3     static double Money(float p, int n)
4     {
5        double m;
6        m = p * n;
7        return m;
8     }
9
10    public static void main(String[] arge)
11    {
12       float price = 2.3f;
13       double tot;
14       tot = Money(price, 11);
15       System.out.println("tot = " + tot);
16    }
17 }
```

- 執行結果：

```
tot = 25.299999237060547
```

呼叫Java方法的語法

- 說明：
  - 行03~08：
    - Money()方法為計算「金額 = 單價 x 數量」的方法。
    - 行03：
      - 虛引數為變數，其中p（單價）為浮點數、n（數量）為整數。
    - 行05：
      - 變數m（金額）為倍精確浮點數。
  - 行14：
    - 實引數為變數price與數值資料11。
  - 行15：
    - 顯示tot的計算結果。

呼叫Java方法的語法

注意：

- 由於p（單價）、n（數量）、m（金額）的資料型別不同，轉換時會有些誤差。

傳值呼叫與參考呼叫

- Java呼叫「方法」時，引數傳遞有「傳值呼叫（Call by value）」與「參考呼叫（Call by reference）」兩種。
- 傳值呼叫：
  - 只是將「實引數」傳入到被呼叫的「方法」中，「方法」會另外配置記憶體來儲存「虛引數」（如下圖）。
  - 呼叫方法的「實引數」與被呼叫方法的「虛引數」，縱然引數的名稱相同，但在記憶體中所佔用的記憶位置不同。
  - 使用傳值呼叫，可以防止變數的「值」，讓被呼叫的「方法」改變。

傳值呼叫與參考呼叫

byVal(a , b) ; 將實引數的內容傳給虛引數

static void byVal(int x, int y)

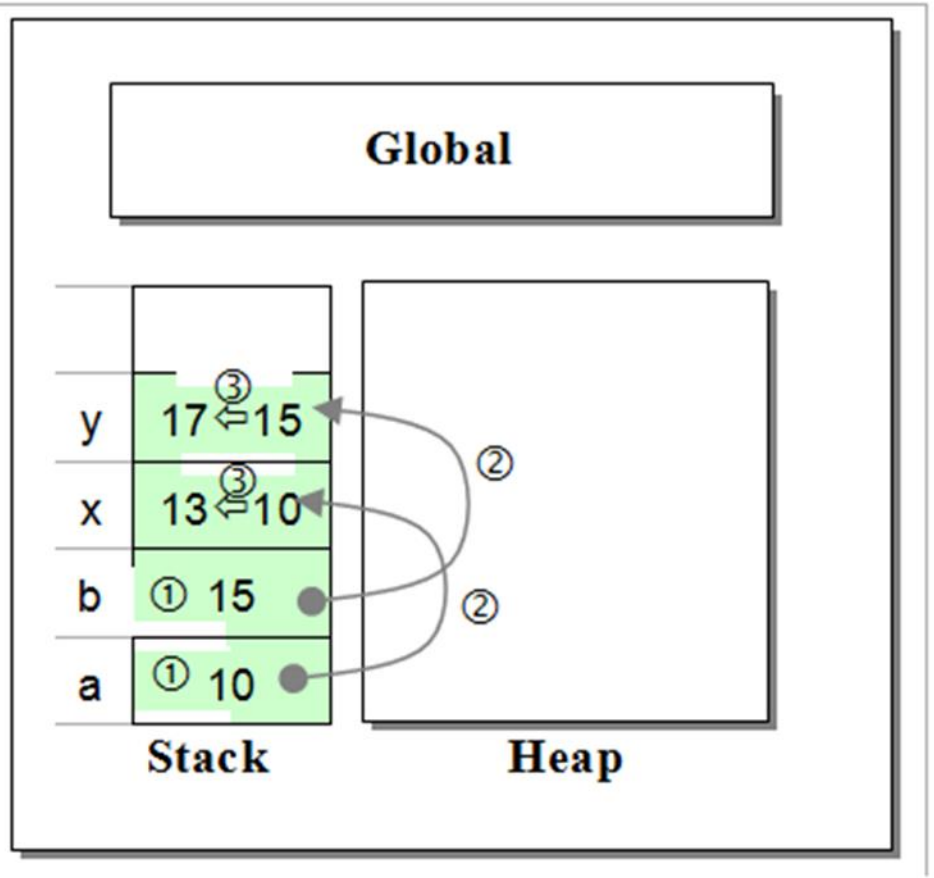
a 的值為 10 傳給 x

b 的值為 15 傳給 y

執行 x+=3 ; y+=2; 之後

則 x=13、y=17

而 a、b 則仍會保持原值



呼叫方法時，記憶體配置狀態

傳值呼叫與參考呼叫

- Java的boolean、char、byte、short、int、long、float、double等資料型別的「變數」，被用於方法中的引數皆屬傳值呼叫。

傳值呼叫與參考呼叫

- 程式：

```
public class CH05_02
{
    static void Change(int x, int y)
    {
        int t;
        t = x;
        x = y;
        y = t;

        System.out.println("Change方法中：");
        System.out.println("x = " + x + "\ty = " + y);
        System.out.println();
}

    public static void main(String[] args)
    {
        int x = 3, y = 5;
```

傳值呼叫與參考呼叫

```
        System.out.println("在main中呼叫Change方法前：");
        System.out.println("x = " + x + "\ty = " + y);

        System.out.println();

        Change(x, y);

        System.out.println("在main中呼叫Change方法後：");
        System.out.println("x = " + x + "\ty = " + y);
    }
}
```

傳值呼叫與參考呼叫

```
1  public class CH05_02
2  {
3     static void Change(int x, int y)
4     {
5        int t;
6        t = x;
7        x = y;
8        y = t;
9
10       System.out.println("Change方法中：");
11       System.out.println("x = " + x + "\ty = " + y);
12       System.out.println();
13    }
14
15    public static void main(String[] args)
16    {
17       int x = 3, y = 5;
18
19       System.out.println("在main中呼叫Change方法前：");
20       System.out.println("x = " + x + "\ty = " + y);
21
22       System.out.println();
23
24       Change(x, y);
25
26       System.out.println("在main中呼叫Change方法後：");
27       System.out.println("x = " + x + "\ty = " + y);
28    }
29 }
```

傳值呼叫與參考呼叫

- 執行結果：

```
在main中呼叫Change方法前：
x = 3     y = 5

Change方法中：
x = 5     y = 3

在main中呼叫Change方法後：
x = 3     y = 5
```

- 說明：
  - 行03~12：
    - Change()方法；將x與y的內容對調。
    - 行05：
      - 宣告整數變數t。
      - 用來暫存要交換的資料。
    - 行06~行08：
      - 交換資料。

傳值呼叫與參考呼叫

- 行10~12：
  - 將x、y對調以後的內容顯示出來。
- 行17：
  - 宣告整數變數，並指定初值。
- 行19~20：
  - 呼叫Change()方法前，顯示x與y的內容。
- 行22：
  - 顯示空行。
- 行24：
  - 呼叫Change()方法。
- 行26~27：
  - 顯示呼叫Change()方法後，x與y的內容。

傳值呼叫與參考呼叫

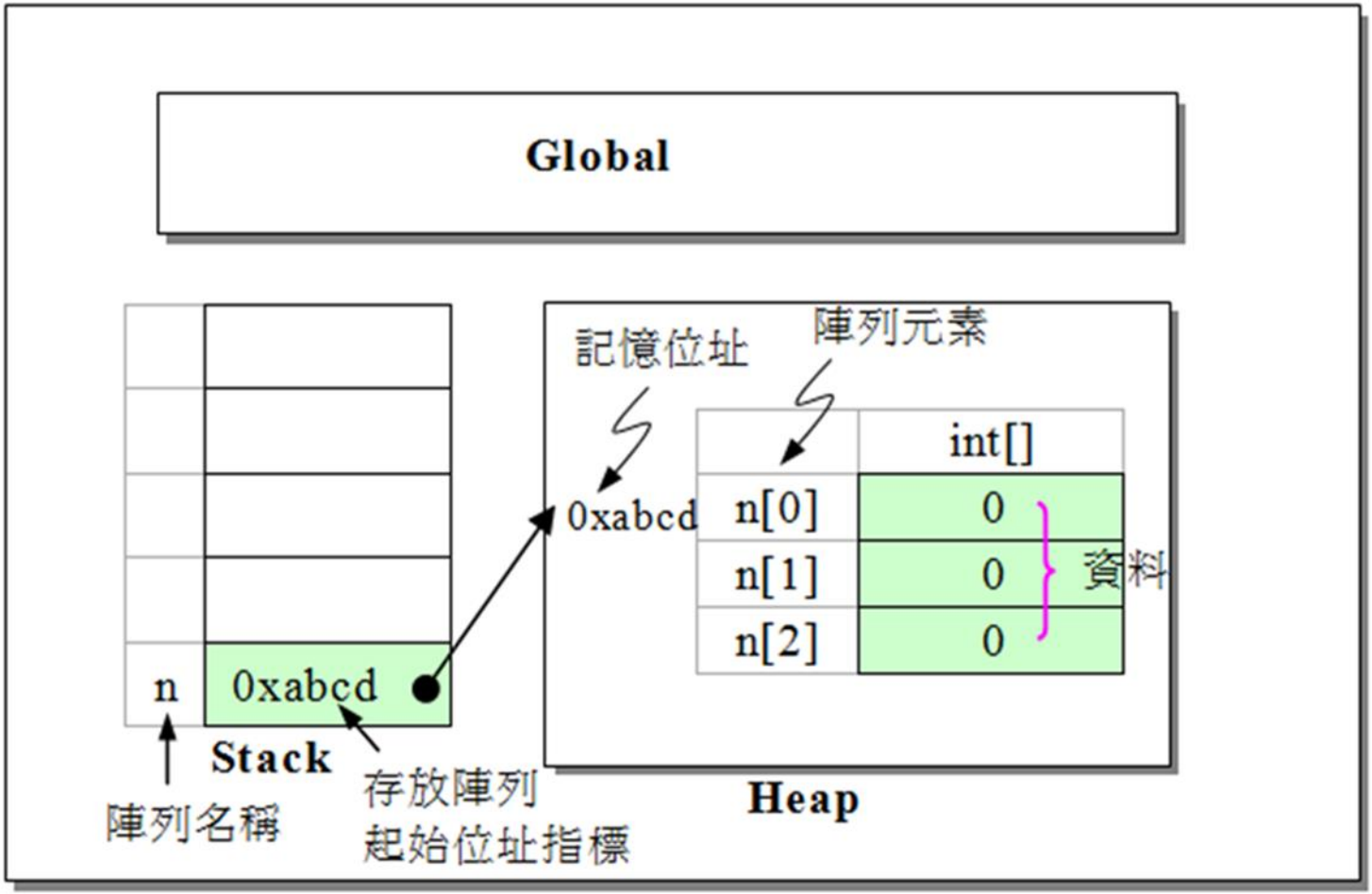
注意：

- Change方法內的x、y變數與main方法內的x、y變數名稱雖然相同，但所佔用記憶體的位置不同。
- Change方法內的x、y變數內容有改變時，main方法的x、y變數的內容，仍然為原來的內容。

傳值呼叫與參考呼叫

- 參考呼叫：
  - 將「實引數」的記憶體位置傳入到被呼叫的「方法」中，「方法」不會另外配置記憶體來儲存「虛引數」（如下圖）。
  - 呼叫「方法」的「實引數」與被呼叫「方法」的「虛引數」，縱然引數的名稱不相同，但在記憶體中所佔用的記憶位置是相同的。
  - 若在被呼叫的「方法」中，變更參數的「值」，也會同時改變呼叫「方法」的參數「值」。
  - 使用傳址呼叫，可以透過該引數，直接將「值」傳回給呼叫「方法」。
  - Java的陣列、物件、...等引數的傳遞，皆屬參考呼叫。

傳值呼叫與參考呼叫



呼叫方法時，記憶體配置狀態

傳值呼叫與參考呼叫

- 程式：

```
public class CH05_03
{
    static void Sort(char data[])
    {
        int i, j;
        char t;
        for (i = data.length - 2; i >= 0; i--)
            for (j = 0; j <= i; j++)
                if (data[j] > data[j + 1])
                {
                    t = data[j];
                    data[j] = data[j + 1];
                    data[j + 1] = t;
                }
        System.out.println("在Sort中，排序後：");
        for (i = 0; i <= data.length - 2; i++)
            System.out.print(data[i] + "，");
        System.out.println(data[data.length - 1]);
```

傳值呼叫與參考呼叫

```
}

public static void main(String[] args)
{
    char word[] = { 'B', 'p', 'D', 'a', 'G' };

    int i;

    System.out.println("在main中，排序前：");
    for (i = 0; i <= word.length - 2; i++)
        System.out.print(word[i] + "，");
    System.out.println(word[word.length - 1]);

    System.out.println();

    Sort(word);

    System.out.println();
```

傳值呼叫與參考呼叫

```
        System.out.println("在main中，排序後：");
        for (i = 0; i <= word.length - 2; i++)
            System.out.print(word[i] + "，");
        System.out.println(word[word.length - 1]);
    }
}
```

傳值呼叫與參考呼叫

```
1  public class CH05_03
2  {
3     static void Sort(char data[])
4     {
5        int i, j;
6        char t;
7        for (i = data.length - 2; i >= 0; i--)
8           for (j = 0; j <= i; j++)
9              if (data[j] > data[j + 1])
10             {
11                t = data[j];
12                data[j] = data[j + 1];
13                data[j + 1] = t;
14             }
15       System.out.println("在Sort中，排序後：");
16       for (i = 0; i <= data.length - 2; i++)
17          System.out.print(data[i] + "，");
18       System.out.println(data[data.length - 1]);
19    }
20
21    public static void main(String[] args)
22    {
23       char word[] = { 'B', 'p', 'D', 'a', 'G' };
24
25       int i;
26
```

傳值呼叫與參考呼叫

```
27          System.out.println("在main中，排序前：");
28          for (i = 0; i <= word.length - 2; i++)
29              System.out.print(word[i] + "，");
30          System.out.println(word[word.length - 1]);
31
32          System.out.println();
33
34          Sort(word);
35
36          System.out.println();
37
38          System.out.println("在main中，排序後：");
39          for (i = 0; i <= word.length - 2; i++)
40              System.out.print(word[i] + "，");
41          System.out.println(word[word.length - 1]);
42      }
43  }
```

傳值呼叫與參考呼叫

- 執行結果：

```
在main中，排序前：
B，p，D，a，G

在Sort中，排序後：
B，D，G，a，p

在main中，排序後：
B，D，G，a，p
```

- 說明：
  - 行03~19：
    - Sort()方法；將data字元陣列元素，依順序排序。
    - 行5：
      - 宣告整數變數。
    - 行6：
      - 宣告字元變數。
      - 暫時儲存要交換的資料。

傳值呼叫與參考呼叫

- 行07~行14：
  - 進行排序。
- 行15~行18：
  - 顯示排序後的結果。

- 行23：
  - 宣告字元陣列word，並指定陣列的元素值。
- 行25：
  - 宣告整數變數。
- 行27~30：
  - 顯示排序前，字元陣列中的元素值。
- 行32：
  - 顯示空行。
- 行34：
  - 呼叫Sort()方法，並傳入字元陣列word。

傳值呼叫與參考呼叫

- 行36：
  - 顯示空行。
- 行38~41：
  - 顯示排序後，字元陣列中的元素值。

注意：

- 此為參考呼叫，實引數為word陣列，虛引數為data陣列，兩者名稱雖不同，但所佔用的記憶體位置相同。
- data陣列在執行排序後，元素值的內容有所改變，縱使Sort方法沒有傳回值，但因為兩個陣列的參照位置相同，故在main中顯示word陣列的結果，會與在Sort方法相同。

遞迴

- 在方法中呼叫本身的方式稱為「遞迴」（Recursive）。
- 使用遞迴優點是程式較精簡；缺點是在遞迴方法中若沒有設定一個條件來結束該方法的執行，會不斷呼叫自己，造成無窮迴圈。
- 此種方法在科學問題上常被使用。

  例如：

  階乘、排列、組合、因數、...等。

遞迴

- 階乘
  - 一個正整數的階乘（factorial）是所有小於及等於該數的正整數的積，並且0的階乘為1。
  - 自然數n的階乘寫作n!。
    - 定義：

$$n! = n \times (n-1) \times \cdots \times 2 \times 1$$

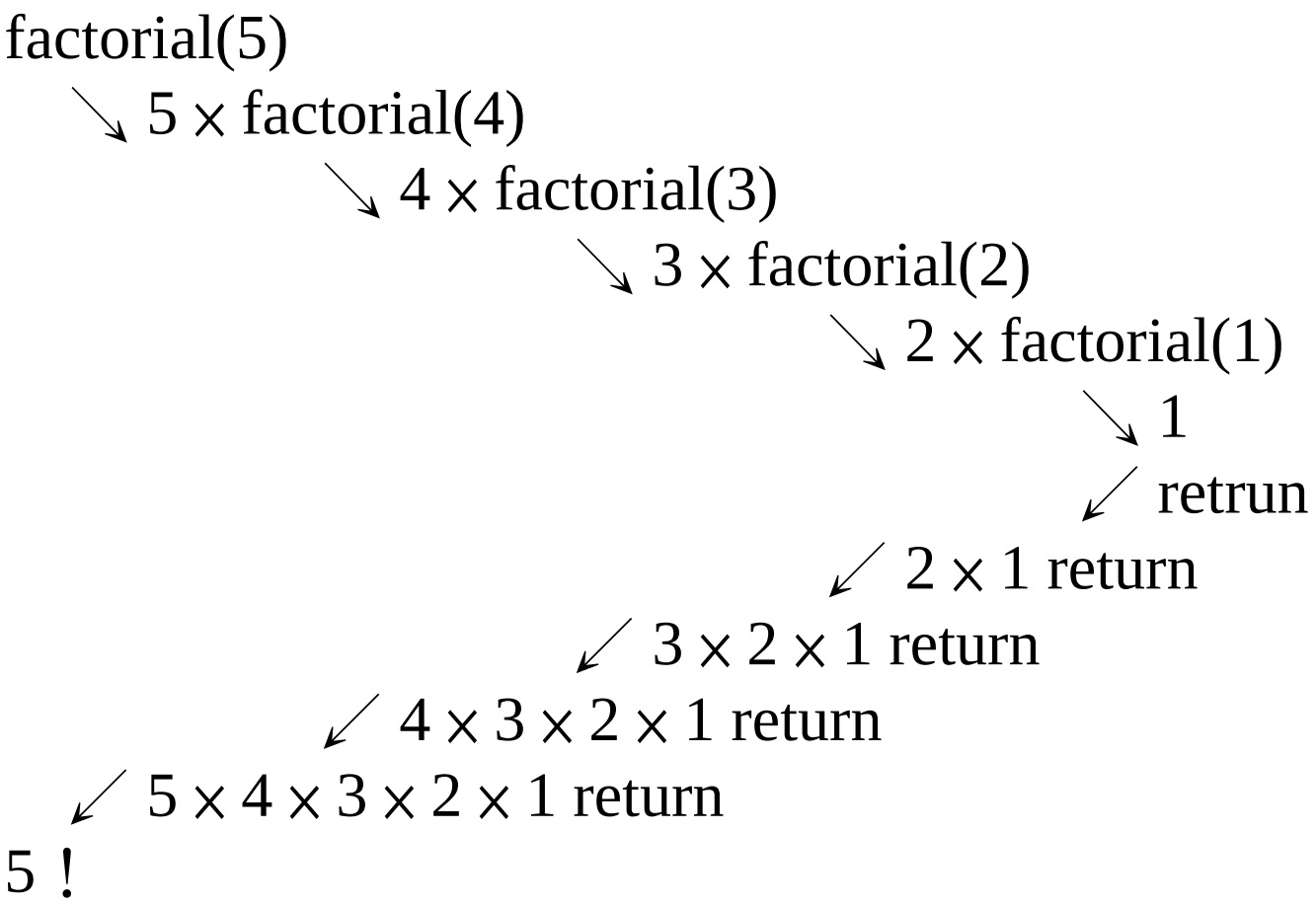
即 $n! = n \times (n-1)!$，且$0!=1$。

例如：

5!=5×4×3×2×1=120

遞迴

- 說明：


factorial(5)
↘ 5 × factorial(4)
↘ 4 × factorial(3)
↘ 3 × factorial(2)
↘ 2 × factorial(1)
↘ 1
↙ retrun
↙ 2 × 1 return
↙ 3 × 2 × 1 return
↙ 4 × 3 × 2 × 1 return
↙ 5 × 4 × 3 × 2 × 1 return
5！

遞迴

- 程式：

```
public class CH05_04
{
    static int Fact(int n)
    {
        if (n == 1)
            return 1;
        else
            return n * Fact(n - 1);
    }

    public static void main(String[] args)
    {
        int m, fact_val;

        System.out.println("5! = " + Fact(5));

        System.out.println();
```

遞迴

```
        m = 6;
        fact_val = Fact(m);
        System.out.println(m + "! = " + fact_val);
    }
}
```

遞迴

```
1  public class CH05_04
2  {
3     static int Fact(int n)
4     {
5        if (n == 1)
6           return 1;
7        else
8           return n * Fact(n - 1);
9     }
10
11    public static void main(String[] args)
12    {
13       int m, fact_val;
14
15       System.out.println("5! = " + Fact(5));
16
17       System.out.println();
18
19       m = 6;
20       fact_val = Fact(m);
21       System.out.println(m + "! = " + fact_val);
22    }
23 }
```

遞迴

- 執行結果：

```
5! = 120

6! = 720
```

- 說明：
  - 行03~行09：
    - Fact()方法，求n！。
    - 行06：
      - n=1時，結束遞迴呼叫，並傳回值1。
    - 行08：
      - n≠1時，呼叫Fact()方法本身。
  - 行15：
    - 顯示5！的結果。
  - 行20：
    - 顯示6！的結果。

遞迴

- 課堂練習：
  - 請將此（CH05_04）程式更改，以符合下列條件：
    由使用者『輸入』0~20間的數字，並計算該數字的階乘。

String類別的compareTo()方法

- 字串比較：
  - 語法：

字串1.compareTo(字串2);

  - 用途：
    - 按字典順序，逐字元比較字串1與字串2是否相同。
  - 說明：
    - 字串1：
      - 要比較的字串。
    - 字串2：
      - 被比較的字串。

String類別的compareTo()方法

例如：

```
String str1="String are immutable";
String str2="String are immutable";
int result=str1.compareTo(str2);
```

注意：

- 有傳回值。
  - 若兩個字串中的每個字元都相同就回傳 0。
  - 若兩個字串有不同，則回傳第一個不同字元的Unicode差值。
    - 當兩個字串出現第一個不同字元時，參數中的字元是一個按字典順序比較大的字元，傳回小於0的值。
    - 當兩個字串出現第一個不同字元時，參數中的字元是一個按字典順序比較小的字元，傳回大於0的值。

String類別的compareTo()方法

- 程式：

```
public class CH05_05
{
    public static void main(String[] args)
    {
        String str1 = "My name is George.";
        String str2 = "My name is George.";
        String str3 = "Your name is Joe.";

        int result = str1.compareTo(str2);
        System.out.println("str1與str2比較的結果：" + result);

        result = str2.compareTo(str3);
        System.out.println("\nstr2與str3比較的結果：" + result);

         result = str3.compareTo(str1);
        System.out.println("\nstr3與str1比較的結果：" + result);
    }
}
```

String類別的compareTo()方法

```
1  public class CH05_05
2  {
3     public static void main(String[] args)
4     {
5        String str1 = "My name is George.";
6        String str2 = "My name is George.";
7        String str3 = "Your name is Joe.";
8
9        int result = str1.compareTo(str2);
10       System.out.println("str1與str2比較的結果：" + result);
11
12       result = str2.compareTo(str3);
13       System.out.println("\nstr2與str3比較的結果：" + result);
14
15       result = str3.compareTo(str1);
16       System.out.println("\nstr3與str1比較的結果：" + result);
17    }
18 }
```

- 執行結果：

```
str1與str2比較的結果：0

str2與str3比較的結果：-12

str3與str1比較的結果：12
```

String類別的compareTo()方法

- 說明：
  - 行05~行07：
    - 宣告字串變數。
  - 行09~行10：
    - 顯示str1與str2的比較結果。
    - 因為str1與str2完全相同，故傳回值為0。
  - 行12~行13：
    - 顯示str2與str3的比較結果。
    - 因為str2與str3不相同，M在Y的前面，且相差12個位置，故傳回值為-12。
  - 行15~行16：
    - 顯示str3與str1的比較結果。
    - 因為str3與str1不相同，Y在M的後面，且相差12個位置，故傳回值為12。

Math類別

- 遇到較複雜的數學運算時，可用到Math類別所提供的方法。常用的有：
  - 常數：
    - 傳回自然對數函數的底數（即 e=2.718281828459045）

      Math.E

    - 傳回圓周率（即 π=3.141592653589793）

      Math.PI

Math類別

- 三角函數：
  - 將度數轉成弧度

    double Math.toRadians(double x)

  - 將弧度轉成度數

    Math.toDegrees(double x)

  - 正弦函數，x的單位為弧度

    double Math.sin(double x)

  - 餘弦函數，x的單位為弧度

    double Math.cos(double x)

Math類別

- 正切函數，x的單位為弧度

  double Math.tan(double x)

- 反正弦函數sin-1(x)，傳回值為弧度

  double Math.asin(double x)

- 反餘弦函數cos-1(x)，傳回值為弧度

  double Math.acos(double x)

- 反正切函數tan-1(y/x)，傳回值為弧度

  double Math.atan(double x)

Math類別

- 反正切函數tan-1(y/x)，傳回值為弧度

  double Math.atan2(double y, double x)

- 指數：
  - 以e為底的指數函數（即 e^x）

    double Math.exp(double x)

  - 以e為底的自然對數函數（即 ln(x)）

    double Math.log(double x)

  - x的y次方（即 x^y）

    double Math.pow(double x, double y)

Math類別

- 平方根（即$\sqrt{x}$，$x \geq 0$）

  double Math.sqrt(double x)

- 其它：
  - 取亂數n（$0 < n < 1$）

    double Math.random()

  - 無條件捨去法取整數（傳回小於x的最大整數部分）

    double Math.floor(double x)

  - 無條件進位法取整數（傳回大於x的最小整數部分）

    double Math.ceil(double x)

Math類別

- 以四捨五入方式取整數

  double Math.rint(double x)

- 以四捨五入方式取整數

  long Math.round(double x)

- 以四捨五入方式取整數

  int Math.round(float x)

- 取絕對值（即 |x|）

  Math.abs(x)

Math類別

- 取兩者間的較大值

  Math.max(x, y)

- 取兩者間的較小值

  Math.min(x, y)

注意：

- 角度與弧度的轉換，會因為電腦本身的問題，略有誤差。
- 用rint方法取整數時，當個位數為奇數時，小數第一位會四捨五入；當個位數為偶數時，則小數會被捨去。

Math類別

- 程式：

```
public class CH05_06
{
    public static void main(String[] args)
    {
        for (double deg = 0; deg <= 360; deg += 45)
            System.out.println("sin(" + deg + ") = " + Math.sin(Math.toRadians(deg)));
    }
}
```

Math類別

```
1  public class CH05_06
2  {
3     public static void main(String[] args)
4     {
5        for (double deg = 0; deg <= 360; deg += 45)
6           System.out.println("sin(" + deg + ") = " + Math.sin(Math.toRadians(deg)));
7     }
8  }
```

- 執行結果：

```
sin(0.0) = 0.0
sin(45.0) = 0.7071067811865475
sin(90.0) = 1.0
sin(135.0) = 0.7071067811865476
sin(180.0) = 1.2246467991473532E-16
sin(225.0) = -0.7071067811865475
sin(270.0) = -1.0
sin(315.0) = -0.7071067811865477
sin(360.0) = -2.4492935982947064E-16
```

Math類別

- 說明：
  - 行03~07：
    - 每隔45度計算sin(x)值。
    - 行06：
      - 先將度轉換為弧度。

注意：

- 由於電腦計算過程有誤差，因此sin(0°)與sin(360°)並不相等。

日期與時間

- 有關日期與時間類別，是放在util套件中，在此介紹Date與Calendar類別。
- Date物件可以獲得目前電腦完整日期與時間。
- Calendar類別可以分別傳回與設定年、月、日、小時、分、秒、毫秒等。
  - 日期、時間、日曆
    - 取得目前電腦完整日期與時間

      Date 物件名稱 = new Date();

    - 取得系統目前日曆物件

      Calendar 物件名稱 = Calendar.getInstance();

日期與時間

- 傳回目前年份（西元年）

  物件名稱.get(Calendar.YEAR)

- 傳回目前月份（0為1月、1為2月、...、11為12月）

  物件名稱.get(Calendar.MONTH)

- 傳回目前月份的日期（日期為1~31）

  物件名稱.get(Calendar.DAY_OF_MONTH)

- 傳回24小時制時間

  物件名稱.get(Calendar HOUR_OF_DAY)

日期與時間

- 傳回分

  物件名稱.get(Calendar MINUTE)

- 傳回秒

  物件名稱.get(Calendar SECOND)

- 傳回毫秒

  物件名稱.get(Calendar MILLISECOND)

- 傳回系統時間（從1970年1月1日的00:00:00.000開始到現在的時間，單位為毫秒〔1/1000秒〕）

  System.currentTimeMillis();

日期與時間

- 程式

```
import java.io.*;
import java.util.*;

public class CH05_07
{
    public static void main(String[] args) throws IOException
    {
        long t1, t2, tot_t;
        Date date_time1 = new Date();
        Calendar date_time2 = Calendar.getInstance();
        t1 = System.currentTimeMillis();
        System.out.println("\n開始：" + t1 + "毫秒\n");
        System.out.println(date_time1);
        System.out.println(date_time2);
        System.out.println(date_time2.get(Calendar.YEAR) + "年");
```

日期與時間

```
        System.out.println(date_time2.get(Calendar.MONTH) + 1 + "月");
        System.out.println(date_time2.get(Calendar.DAY_OF_MONTH) + "日");
        System.out.println(date_time2.get(Calendar.HOUR_OF_DAY) + "時");
        System.out.println(date_time2.get(Calendar.MINUTE) + "分");
        System.out.println(date_time2.get(Calendar.SECOND) + "秒");
        System.out.println(date_time2.get(Calendar.MILLISECOND) + "毫秒");

        System.out.println("\n按 << Enter >> 結束測試時間");
        int p = System.in.read();
        t2 = System.currentTimeMillis();
        System.out.println("\n結束： " + t2 + "毫秒");
        tot_t = t2 - t1;
        System.out.println("\n全部：" + (double) tot_t / 1000 + "秒");
    }
}
```

日期與時間

```
1  import java.io.*;
2  import java.util.*;
3
4  public class CH05_07
5  {
6     public static void main(String[] args) throws IOException
7     {
8        long t1, t2, tot_t;
9        Date date_time1 = new Date();
10       Calendar date_time2 = Calendar.getInstance();
11       t1 = System.currentTimeMillis();
12       System.out.println("\n開始：" + t1 + "毫秒\n");
13       System.out.println(date_time1);
14       System.out.println(date_time2);
15       System.out.println(date_time2.get(Calendar.YEAR) + "年");
16       System.out.println(date_time2.get(Calendar.MONTH) + 1 + "月");
17       System.out.println(date_time2.get(Calendar.DAY_OF_MONTH) + "日");
18       System.out.println(date_time2.get(Calendar.HOUR_OF_DAY) + "時");
19       System.out.println(date_time2.get(Calendar.MINUTE) + "分");
20       System.out.println(date_time2.get(Calendar.SECOND) + "秒");
21       System.out.println(date_time2.get(Calendar.MILLISECOND) + "毫秒");
22
23       System.out.println("\n按 << Enter >> 結束測試時間");
24       int p = System.in.read();
25       t2 = System.currentTimeMillis();
26       System.out.println("\n結束：" + t2 + "毫秒");
27       tot_t = t2 - t1;
28       System.out.println("\n全部：" + (double) tot_t / 1000 + "秒");
29    }
30 }
```

日期與時間

- 執行結果：

```
開始：1530969040612毫秒

Sat Jul 07 21:10:40 CST 2018
java.util.GregorianCalendar[time=1530969040596,areFieldsSet=true,areAllFieldsSet=
2018年
7月
7日
21時
10分
40秒
596毫秒

按 << Enter >> 結束測試時間


結束： 1530969062100毫秒

全部：21.488秒
```

日期與時間

- 說明：
  - 行01：
    - 載入java.io.*套件。
  - 行02：
    - 載入java.util.*套件。
  - 行09：
    - 取得目前電腦完整日期與時間。
  - 行10：
    - 取得系統目前日曆物件。
  - 行11：
    - t1為計時開始。

日期與時間

- 行13：
  - 顯示電腦目前「星期 月 日 小時:分:秒 時區 西元」。其中CST代表「中原標準時間」。
- 行14：
  - 顯示電腦目前「日曆」。
- 行15~21：
  - 分別顯示電腦目前「年、月、日、小時、分、秒、毫秒」。
- 行24：
  - 等待按鍵輸入資料才往下執行。
- 行25：
  - t2為計時結束。
- 行27：
  - 全部時間，單位為毫秒。

格式化資料

- 為了使資料顯示能夠精簡或美觀，就需要使用格式化資料的類別。
- 在java.text套件提供FORMAT類別能進行資料格式化。
- 在此應用到NumberFormat，DecimalFormat與DateFormat類別。
  - NumberFormat類別

    將數值資料型別進行格式化。
    - 整數部分預設以千分位顯示，小數取3位

    getInstance()

格式化資料

- 只取整數部分

  getIntegerInstance()

- 以百分比顯示資料，不顯示小數

  getPercentInstance()

- 資料最前面加$，小數取2位

  getCurrencyInstance()

- 小數部分最多取到n位

  setMaximumFractionDigits(n)

格式化資料

- 小數部分最少取到n位

setMinimumFractionDigits(n)

格式化資料

- 程式：

```
import java.text.*;

public class CH05_08
{
    public static void main(String[] args)
    {
        double d1 = 12345.765;
        NumberFormat nf1 = NumberFormat.getInstance();
        System.out.println(nf1.format(d1));
        System.out.println(NumberFormat.getIntegerInstance().format(d1));
        System.out.println(NumberFormat.getPercentInstance().format(d1));
        System.out.println(NumberFormat.getCurrencyInstance().format(d1));
    }
}
```

格式化資料

```
1  import java.text.*;
2
3  public class CH05_08
4  {
5     public static void main(String[] args)
6     {
7        double d1 = 12345.765;
8        NumberFormat nf1 = NumberFormat.getInstance();
9        System.out.println(nf1.format(d1));
10       System.out.println(NumberFormat.getIntegerInstance().format(d1));
11       System.out.println(NumberFormat.getPercentInstance().format(d1));
12       System.out.println(NumberFormat.getCurrencyInstance().format(d1));
13    }
14 }
```

- 執行結果：

```
12,345.765
12,346
1,234,576%
NT$12,345.76
```

格式化資料

- 說明：
  - 行01：
    - 載入java.text.*套件。
  - 行08~行09：
    - 可以將此兩行合併為 System.out.println(NumberFormat.getInstance().format(di));
  - 行10~行12：
    - 最後一位數會四捨五入。

格式化資料

- 程式：

```
import java.text.*;

public class CH05_09
{
   public static void main(String[] args)
   {
      double d1 = 12345678901d, d2 = 3.055, d3;
      d3 = d1 / d2;
      NumberFormat nf1 = NumberFormat.getInstance();
      nf1.setMinimumFractionDigits(1);
      nf1.setMaximumFractionDigits(2);
      System.out.println("12345678901 ： " + nf1.format(d1));
      System.out.println("3.055 ： " + nf1.format(d2));
      System.out.println("12345678901 / 3.055 = " + d3);
      System.out.println(d3 + " ： " + nf1.format(d3));
   }
}
```

格式化資料

```
1  import java.text.*;
2
3  public class CH05_09
4  {
5     public static void main(String[] args)
6     {
7        double d1 = 12345678901d, d2 = 3.055, d3;
8        d3 = d1 / d2;
9        NumberFormat nf1 = NumberFormat.getInstance();
10       nf1.setMinimumFractionDigits(1);
11       nf1.setMaximumFractionDigits(2);
12       System.out.println("12345678901 : " + nf1.format(d1));
13       System.out.println("3.055 : " + nf1.format(d2));
14       System.out.println("12345678901 / 3.055 = " + d3);
15       System.out.println(d3 + " : " + nf1.format(d3));
16    }
17 }
```

- 執行結果：

```
12345678901 : 12,345,678,901.0
3.055 : 3.06
12345678901 / 3.055 = 4.041138756464812E9
4.041138756464812E9 : 4,041,138,756.46
```

格式化資料

- 說明：
  - 行01：
    - 載入java.text.*套件。
  - 行07：
    - double資料型別，整數太大時後面加d。
  - 行10~行11：
    - 小數部分最少取1位，最多取2位。

格式化資料

- DecimalFormat類別
  - DecimalFormat類別可以自訂數字的輸出格式，其建構子如下：

    DecimalFormat（String pattern）

  - 使用下列符號來定義pattern格式：

符號	功能
0	顯示數字位數（數字0時，顯示0）
#	顯示數字位數（整數左邊為0，或小數右邊為0，不顯示0）
,	千分位符號（用於整數部分）
$	貨幣符號（用於數值的最左邊）
%	百分比符號（用於數值的最右邊）

格式化資料

- 程式：

```
import java.text.*;

public class CH05_10
{
    public static void main(String[] args)
    {
        double d1 = 0, d2 = 12345.6789;
        Format nf1 = new DecimalFormat("##.00");
        System.out.println(nf1.format(d1));
        System.out.println(nf1.format(d2));
        Format nf2 = new DecimalFormat("$, ##0.0#");
        System.out.println(nf2.format(d1));
        System.out.println(nf2.format(d2));
        Format nf3 = new DecimalFormat("#0%");
        System.out.println(nf3.format(d2));
    }
}
```

格式化資料

```
1  import java.text.*;
2
3  public class CH05_10
4  {
5     public static void main(String[] args)
6     {
7        double d1 = 0, d2 = 12345.6789;
8        Format nf1 = new DecimalFormat("##.00");
9        System.out.println(nf1.format(d1));
10       System.out.println(nf1.format(d2));
11       Format nf2 = new DecimalFormat("$, ##0.0#");
12       System.out.println(nf2.format(d1));
13       System.out.println(nf2.format(d2));
14       Format nf3 = new DecimalFormat("#0%");
15       System.out.println(nf3.format(d2));
16    }
17 }
```

- 執行結果：

```
.00
12345.68
$0.0
$12,345.68
1234568%
```

格式化資料

- 說明：
  - 行01：
    - 載入java.text.*套件。
  - 行09：
    - 小數點後面顯示二位，遇到0會顯示。
  - 行12：
    - 小數點後面顯示二位，但第二位的0不會顯示。

格式化資料

- DateFormat類別

  提供Date物件格式化日期的方法。

  - 顯示年、月、日，上午/下午 幾點幾分

    getInstance()

  - 顯示年、月、日

    getDateInstance()

  - 顯示 上午/下午 幾點幾分幾秒

    getTimeInstance()

格式化資料

- 顯示年、月、日，上午/下午 幾點幾分幾秒

getDateTimeInstance()

格式化資料

- 程式：

```
import java.util.*;
import java.text.*;

public class CH05_11
{
    public static void main(String[] args)
    {
        Date date1 = new Date();
        System.out.println(DateFormat.getInstance().format(date1));
        System.out.println(DateFormat.getDateInstance().format(date1));
        System.out.println(DateFormat.getTimeInstance().format(date1));
        System.out.println(DateFormat.getDateTimeInstance().format(date1));
    }
}
```

格式化資料

```
1  import java.util.*;
2  import java.text.*;
3
4  public class CH05_11
5  {
6     public static void main(String[] args)
7     {
8        Date date1 = new Date();
9        System.out.println(DateFormat.getInstance().format(date1));
10       System.out.println(DateFormat.getDateInstance().format(date1));
11       System.out.println(DateFormat.getTimeInstance().format(date1));
12       System.out.println(DateFormat.getDateTimeInstance().format(date1));
13    }
14 }
```

- 執行結果：

```
2017/10/17 下午 10:45
2017/10/17
下午 10:45:56
2017/10/17 下午 10:45:56
```

格式化資料

- 說明：
  - 行01：
    - 載入java.util.*套件。
  - 行02：
    - 載入java.text.*套件。
  - 行08：
    - 取得目前電腦完整日期與時間。
  - 行09~行12：
    - 顯示格式化的日期。

格式化資料

- SimpleDateFormat類別

  可以自訂日期輸出格式，其建構子如下：

  SimpleDateFormat(String pattern)

  - 使用英文字母來定義pattern格式（注意大小寫）：

字母	功能
y	年
M	月
d	日
E	星期
a	上午\下午

字母	功能
H	時（0~23）
h	時（1~12）
m	分
s	秒
S	毫秒

格式化資料

- 程式：

```
import java.util.*;
import java.text.*;

public class CH05_12
{
    public static void main(String[] args)
    {
        long sum = 0;
        for (int i = 0; i <= 10; i++)
        {
            Date date1 = new Date();
            DateFormat df = new SimpleDateFormat("yyyy 年 M 月 dd 日 E a hh 點
                                mm 分 ss秒");
            System.out.println(df.format(date1));
```

格式化資料

```
            for (long j = 1; j <= 1000000000; j++)
                sum++;
        }
        System.out.println("sum = " + sum);
    }
}
```

格式化資料

```
1  import java.util.*;
2  import java.text.*;
3
4  public class CH05_12
5  {
6     public static void main(String[] args)
7     {
8        long sum = 0;
9        for (int i = 0; i <= 10; i++)
10       {
11          Date date1 = new Date();
12          DateFormat df = new SimpleDateFormat("yyyy 年 M 月 dd 日 E a hh 點 mm 分 ss秒");
13          System.out.println(df.format(date1));
14          for (long j = 1; j <= 1000000000; j++)
15             sum++;
16       }
17       System.out.println("sum = " + sum);
18    }
19 }
```

格式化資料

- 執行結果：

```
2017 年 10 月 17 日 星期二 下午 11 點 06 分 42秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 43秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 43秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 43秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 44秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 44秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 44秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 45秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 45秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 46秒
2017 年 10 月 17 日 星期二 下午 11 點 06 分 46秒
sum = 11000000000
```

- 說明：
  - 行01：
    - 載入java.util.*套件。
  - 行02：
    - 載入java.text.*套件。

格式化資料

- 行09~16：
  - 反覆執行11次。
    - 行11：

      取得目前電腦完整日期與時間。
    - 行09：

      以自定格式顯示目前日期與時間。
    - 行14~15：

      反覆執行sum加十億次。

綜合練習（1）

題目：**製作『乘法測驗』4題**

要求：

1、以『亂數』產生『被乘數』與『乘數』。

- 『被乘數』範圍為5~20間。
- 『乘數』範圍為0~10間。

2、須計算成績。

- 每題25分。

3、若答錯不扣分，但須給正確答案。

4、須顯示現在電腦的西元年、月、日、星期。

5、須顯示總共花了多少時間完成作答。

綜合練習（2）

題目：**由使用者輸入一個一維陣列，請以『方法』來完成『氣泡排序法』**

要求：

1、『陣列長度』由『使用者輸入』，但必須限制在2~10之間。

2、陣列的『元素值』由『使用者輸入』，但必須限制在1~100之間。

3、建立一個『方法』，功能為降冪排序的『氣泡排序法』。

4、使用者輸入完成的陣列元素值，須『方法』進行降冪排序（最左邊的數字最大，最右邊的數字最小）。

綜合練習（3）

題目：**請以『遞迴』來解『河內塔』問題**

要求：

1、有三根柱子（A、B、C）。

2、『盤數』由『使用者輸入』，但必須限制在3~10之間。

3、建立一個『方法』，功能為解『河內塔』問題。公式為『2^n-1』。

4、須顯示盤子移動的過程。

5、須顯示盤子總共移動的次數。

資料來源

- 蔡文龍、何嘉益、張志成、張力元，JAVA SE 10基礎必修課，台北市，碁峰資訊股份有限公司，2018年7月，出版。
- 吳燦銘、胡昭民，圖解資料結構-使用Java(第三版)，新北市，博碩文化股份有限公司，2018年5月，出版。
- Ivor Horton，Java 8 教學手冊，台北市，碁峰資訊股份有限公司，2016年9月，出版。
- 李春雄，程式邏輯訓練入門與運用---使用JAVA SE 8，台北市，上奇科技股份有限公司，2016年6月，初版。
- 位元文化，Java 8視窗程式設計，台北市，松崗資產管理股份有限公司，2015年12月，出版。
- Benjamin J Evans、David Flanagan，Java 技術手冊 第六版，台北市，碁峰資訊股份有限公司，2015年7月，出版。
- 蔡文龍、張志成，JAVA SE 8 基礎必修課，台北市，碁峰資訊股份有限公司，2014年11月，出版。
- 陳德來，Java SE 8程式設計實例，台北市，上奇科技股份有限公司，2014年11月，初版。
- 林信良，Java SE 8 技術手冊 ，台北市，碁峰資訊股份有限公司，2014年6月，出版。
- 何嘉益、黃世陽、李篤易、張世杰、黃鳳梅，徐政棠譯，JAVA2 程式設計從零開始--適用JDK7，台北市，上奇資訊股份有限公司，2012年5月，出版。