

Looping

迴圈

Java Fundamental



Content

- ◆ 迴圈概念
- ◆ 迴圈分類
- ◆ 迴圈進階

Content

◆ 迴圈概念

◆ 迴圈分類

◆ 迴圈進階

迴圈概念 (Looping)

◆ 所謂迴圈 (Loop) 就是在達到某個目的前，不斷地做同樣的事，直到這個目的達成為止。

◆ Java主要有三種迴圈：

- for迴圈。
- while迴圈。
- do~while迴圈。

◆ 迴圈進階控制

- 巢狀結構。
- 無窮迴圈。
- break & continue。
- Label 標籤。

Content

◆ 迴圈概念

◆ 迴圈分類

◆ 迴圈進階

迴圈分類

◆ 預先知道重複執行的次數

➤ 使用 For 迴圈

- 執行 n 次

◆ 知道停止執行的條件，不確定執行次數

➤ 使用 While 迴圈

- 前測型迴圈
- 執行 $0 \sim n$ 次

➤ 使用 Do-While 迴圈

- 後測型迴圈
- 執行 $1 \sim n$ 次

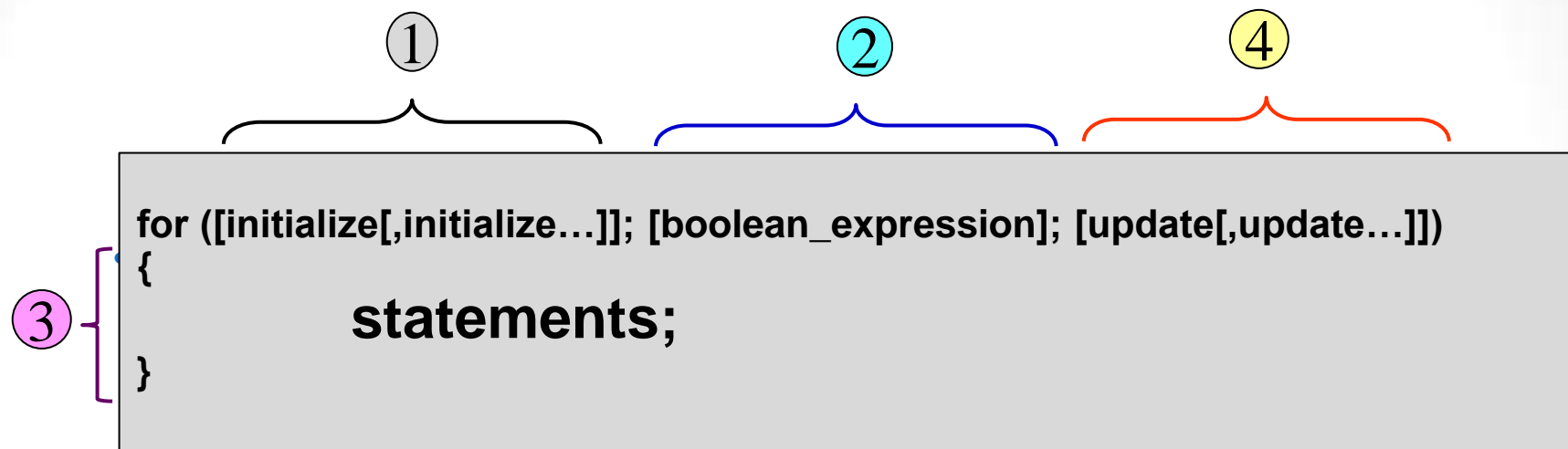
For 迴圈

◆ For迴圈有3個控制式，以跑操場的角度來看這3個控制式：

1. 初始值(起跑點)：設定for迴圈控制變數的初始值。
2. 條件式(終點)：若終點是1000公尺，則條件式可寫成「 $i < 1000$ 」；
當條件式為true時會繼續執行迴圈內容，直到條件式為false才會終止迴圈。
3. 變量(間距)：如果每跑1步就累加1公尺，變量可以寫成「 $i++$ 」；
而「 $i+=2$ 」就代表一次累加2。

```
for ([initialize[,initialize...]]; [boolean_expression]; [update[,update...]])  
{  
    statements;  
}
```

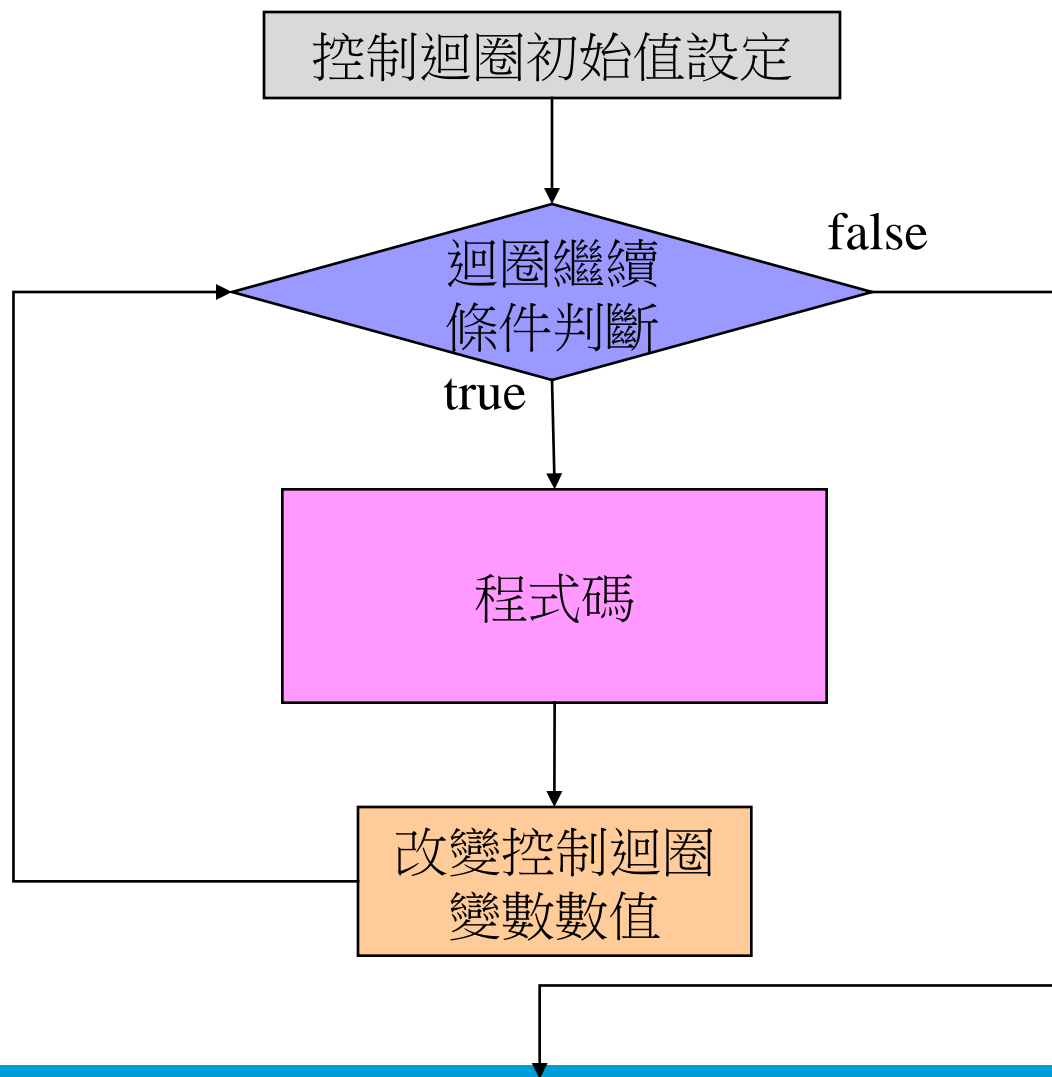
For 迴圈



◆ for迴圈執行時:

- 變數宣告：只執行一次
- 布林判斷式：每次迴圈開始之前
- 執行 `code_blocks`
- 更新狀態：執行`code_blocks` 之後，布林判斷式 之前

For 迴圈



For 迴圈

◆ 名稱習慣用 i, j, k 來命名，迴圈內使用，可降低程式錯誤的機率

◆ 在一個迴圈內，可以同時有多個控制變數

◆ 例：`for(int i=1, j=i+1; i<5; i++, j=i*2)`

{

...

}

◆ 巢狀迴圈

- `for` 迴圈中還有 `for` 迴圈
- 在複雜的情況下使用

For 迴圈

◆ for statement

- 語法

```
for( initialization; termination; increment )  
{  
    statements...  
}
```

- initialization：初始條件

- 在此宣告之變數其有效範圍僅存在於此迴圈內

- 例：`for(int x=0; x<10; x++)`

```
{  
    ...  
}
```

- termination：終止條件

- 決定何時迴圈結束

- increment：增量運算

- 每次迴圈進行後所要做的事

For 迴圈

```
01 public class ForDemo {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         System.out.print("輸入執行次數: ");  
07         int input = scanner.nextInt();  
08  
09         for(int i = input; i > 0 ; i--){  
10             System.out.println("Hello World!");  
11         }  
12     }  
13 }
```

```
for(int i = 0; i < input; i++){  
    System.out.println("Hello World!");  
}
```



```
C:\JavaClass>java ForDemo  
輸入執行次數: 5  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!
```

For 迴圈

◆ Example: [ForExample.java](#)

```
/**  
 * 印出 1~16, 最後在印出最後的 i 值  
 */  
  
int i;  
for (i=1; i <=16; i++)  
    System.out.println ("i = " + i);  
System.out.println ("After loop, i = " + i);
```

For 迴圈

```
public class ForExample
{
    public static void main(String[] args)
    {
        int maxX = 12;
        int maxY = 18;
        for (int x = 0, y = 0; (x < maxX) && (y < maxY); x++, y = x * 2)
        {
            System.out.println(" x < 12 : " + x + ", y < 18 : " + y);
        }
    }
}
```

While 迴圈

- ◆ While 迴圈只有一個條件式用來判斷是否繼續執行迴圈，而沒有初始值和變量值。
- ◆ For迴圈可改寫成while迴圈。

While.java

```
1. class While{
2.     public static void main(String[] args){
3.         int i=2;
4.         System.out.print("等差數列：");
5.         while(i<=10){
6.             System.out.print(i + " ");
7.             i+=2;
8.         }
9.     }
10. }
11.
12. }
```

→ 相當於 **for** 迴圈的初始值

要進入區塊前會先做條件式判斷；執行到
區塊結束大括號時也會做條件式判斷，以
決定是否繼續執行迴圈內容

→ 相當於 **for** 迴圈的變量，如果想要看看
什麼是無限迴圈，可以把這行程式移除

-----輸出-----

等差數列：2 4 6 8 10

While 迴圈

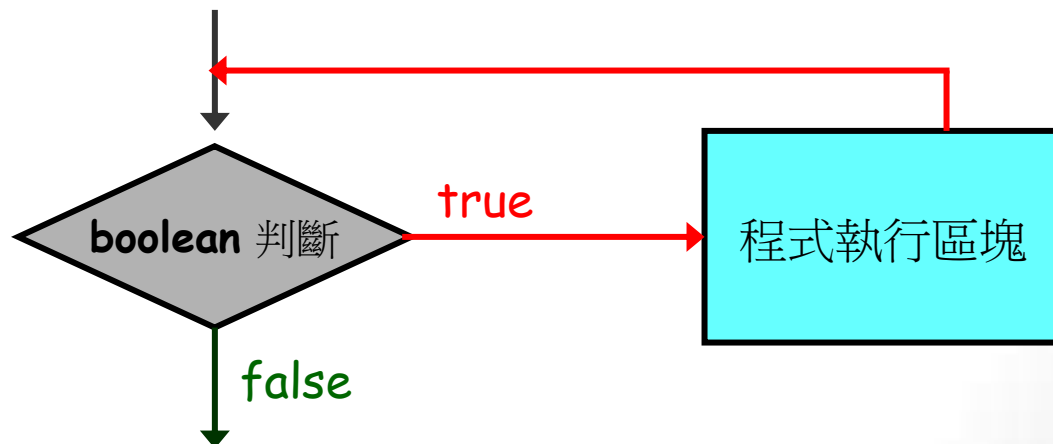
◆ 語法

```
while (expression)  
{  
    statements...  
}
```

- while 會先執行括號內的運算式(*expression*)，此運算式必回傳 true或false
- 若運算式結果為 true 則執行區塊內部敘述，否則跳離
- 適用情況：不知道幾次迴圈才會停止
- 無窮迴圈
 - while (true) { ... }

While 迴圈

```
while (boolean) {  
    statements;  
}
```



While 迴圈

```
01 public class WhileDemo {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         System.out.print("輸入執行次數: ");  
07         int input = scanner.nextInt();  
08  
09         int count = 0; //計數用  
10  
11         while(count < input){ //判斷次數  
12             System.out.println("Hello World!");  
13             count++; //每次執行後遞增  
14         }  
15     }  
16 }
```



The screenshot shows a Windows Command Prompt window titled "系統管理員: 命令提示字元". The command prompt shows the execution of the Java program. The user enters "5" for the number of iterations. The program outputs "Hello World!" five times, once for each iteration. The command prompt shows the following text:

```
c:\JavaClass>java WhileDemo  
輸入執行次數: 5  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
c:\JavaClass>
```

While 迴圈

```
public class WhileExample
{
    public static void main(String[] args)
    {
        double r = 0;
        while(r < 0.99d) {
            r = Math.random(); // Math 類別中提供產生亂數的類別方法
            System.out.println(r);
            } // end of while
        }
    }
```

do while 迴圈

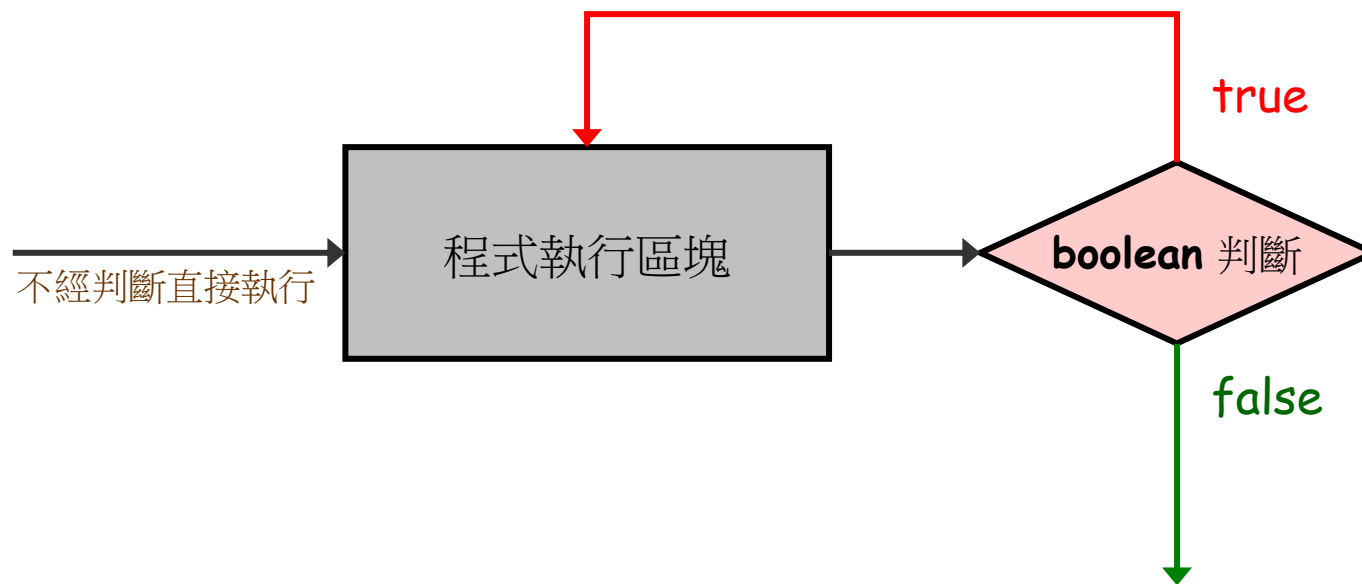
➤ do-while statement

➤ 語法

```
do  
{  
    statements...  
} while (expression);
```

- do...while 先執行區塊內部，完畢後才執行括號內的運算式，回傳 true 或 false
- 若運算式結果為 true，則再執行區塊敘述，否則跳離
- do...while 敘述必至少執行一次區塊內部敘述

do while 迴圈



do while 迴圈

- ◆ do~while 迴圈會在第一次時，先執行區塊內程式碼，之後再做條件式判斷。
- ◆ 在便利商店挑一罐飲料試喝，覺得好喝，才會繼續買相同的飲料。這種觀念，就與 do~while 相同。

DoWhile.java

```
1. class DoWhile{
2.     public static void main(String args[]){
3.         int i=7;
4.         do{
5.             System.out.print(i + " ");
6.             i--;
7.         } while(i>=8);
8.     }
9. }
10. }
```

第一次會直接進入區塊內執行，並不會做條件式判斷

第一次執行完後才會做條件式判斷

-----輸出-----
7

迴圈控制 – do/while

計算所有平方值小於 100 的數字並印出

```
public class DoWhileExample
{
    public static void main (String[] args)
    {
        int i=0, j;
        do {
            i++;
            j = i * i;
            System.out.println ( i + " * " + i + " = " + j);
        } while (j < 100);
    }
}
```

迴圈控制 – do/while

```
01 public class DoWhileDemo {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         int input = 0;  
07         int anser = 30;  
08  
09         do{  
10             System.out.print("輸入數字: ");  
11             input = scanner.nextInt();  
12         } while ( input != anser );  
13  
14         System.out.println("恭喜您!猜對了!");  
15     }  
16 }
```



```
c:\JavaClass>java DoWhileDemo  
輸入數字: 18  
輸入數字: 26  
輸入數字: 30  
恭喜您!猜對了!  
c:\JavaClass>
```


Content

◆ 迴圈概念

◆ 迴圈分類

◆ 迴圈進階

迴圈進階

- ◆ 巢狀迴圈。
- ◆ 無窮迴圈。
- ◆ break & continue。
- ◆ Label 標籤。

巢狀迴圈

◆ 巢狀迴圈

- 巢狀迴圈是在迴圈內擁有其他迴圈，例如：在for迴圈擁有for、while和do/while迴圈，同樣的，在while迴圈內也可以有for、while和do/while迴圈。
- Java巢狀迴圈可以有很多層，二、三、四層都可以，例如：一個二層的巢狀迴圈，在for迴圈內擁有while迴圈，如下所示：

```
for ( i = 1; i <= 9; i++ ) {  
    .....  
    j = 1;  
    while ( j <= 9 ) {  
        .....  
        j++;  
    }  
}
```

巢狀迴圈

◆ 迴圈控制 - 巢狀式for迴圈

- 如果要輸出一列有規則性的資料，可以利用一個for迴圈來達成。
- 但若要輸出的資料是由多列組成，也就是所謂的二維資料，就必須使用巢狀式迴圈(Nested loop)才能辦到。
- 舉例來說，平面座標就可以使用巢狀式for迴圈來表示。

For_Nested.java

```
1. class For_Nested{
2.     public static void main(String args[]){
3.         for(int x=0; x<3; x++){
4.             for(int y=0; y<3; y++){
5.                 System.out.print("(" + x + ", " + y + ") ");
6.
7.                 System.out.println();
8.             }
9.         }
10. }
```

當 x=0 時，y 會從 0 遞增到 2；當 x=1 時，y 也會從 0 遞增到 2。換句話說，x 的值變換 1 次，y 的值就會跑 3 次

換行

-----輸出-----

(0,0) (0,1) (0,2)
(1,0) (1,1) (1,2)
(2,0) (2,1) (2,2)

巢狀迴圈

```
for(int i=1; i<=5; i+=2){  
    for(int j=2; j<=6; j+=2){  
        System.out.print("(" + i + "," + j + ")\t");  
    }  
    System.out.print("\n");  
}
```

執行方向→

外層for第1次	內層for第1次(1,2)	內層for第2次(1,4)	內層for第3次(1,6)	換行
外層for第2次	內層for第4次(3,2)	內層for第5次(3,4)	內層for第6次(3,6)	換行
外層for第3次	內層for第7次(5,2)	內層for第8次(5,4)	內層for第9次(5,6)	換行


巢狀迴圈

◆ Example: `ForExample.java`

```
/**
 * 巢狀迴圈的使用方法
 */
for (int i=1; i <=8; i++)
{
    for (int j=1; j <=7; j++)
    {
        for (int k=1; k <=6; k++)
        {
            System.out.println("i = " + i + " j = " + j + " k = " + k);
        } // end of for k
    } // end of for j
} // end of for i
```

巢狀迴圈

```
01 public class NestedForDemo {
02     public static void main(String[] args) {
03         java.util.Scanner scanner =
04             new java.util.Scanner(System.in);
05
06         System.out.print("輸入長: ");
07         int length = scanner.nextInt();
08         System.out.print("輸入寬: ");
09         int width = scanner.nextInt();
10
11         for (int i = width; i > 0; i-- ) {
12             for(int j = length; j > 0; j--){
13                 System.out.print("@");
14             } //內層 for
15
16             System.out.println(); //換行
17         } //外層 for
18     }
19 }
```



```
C:\JavaClass>java NestedForDemo
輸入長: 4
輸入寬: 3
@
@
@

C:\JavaClass>java NestedForDemo
輸入長: 8
輸入寬: 4
@@@@
@@@@
@@@@
@@@@

C:\JavaClass>
```

巢狀迴圈

```
01 public class NestedWhileDemo {
02     public static void main(String[] args) {
03         java.util.Scanner scanner =
04             new java.util.Scanner(System.in);
05
06         System.out.print("輸入長: ");
07         int length = scanner.nextInt();
08         System.out.print("輸入寬: ");
09         int width = scanner.nextInt();
10
11         int widthCount = 0;
12         while(widthCount < width){
13             int lengthCount = 0;
14             while(lengthCount < length){
15                 System.out.print("@");
16                 lengthCount++;
17             } //內層 while
18
19             System.out.println(); //換行
20             widthCount++;
21         } //外層 while
22     }
23 }
```



```
C:\JavaClass>java NestedWhileDemo
輸入長: 5
輸入寬: 3
#####
#####
#####

C:\JavaClass>java NestedWhileDemo
輸入長: 10
輸入寬: 2
#####
#####

C:\JavaClass>
```


無窮迴圈

```
for(;;)  
{  
    // 區塊內敘述  
}
```

```
while(true)  
{  
    // 區塊內敘述  
}
```

指令迴圈 break & continue

◆ break

- 語法：break;
- 中斷執行，立即跳出
- 用在 switch、迴圈

◆ continue

- 語法：continue;
- 跳過剩餘的程式碼，直接進行下一次迴圈
- 用在迴圈中

◆ return

- 程式會跳出目前所在的 method，回到原先呼叫的 method 的地方
- 離開方法並回傳值：return count;

指令中斷迴圈(break)

◆ break 指令中斷迴圈

- 可以強制跳出所在的迴圈區塊。假設學生每天都要到學校上8堂課，如果學生生病了，就沒有辦法繼續接下來的課程；這種情形就需要使用break跳出迴圈。

```
1. class Loop_Break{
2.     public static void main(String[] args){
3.         boolean isSick = true; //是否生病
4.
5.         for(int i=1; i<=8; i++){
6.             if(isSick){
7.                 System.out.println("停止上課，回家休息！");
8.                 break;
9.             }
10.            System.out.println("開始上第" + i + "堂課");
11.        }
12.    }
13. }
```

-----輸出-----
停止上課，回家休息！

isSick 為 true 就代表生病了

直接跳出迴圈區塊

指令中斷迴圈(break)

```
01 public class BreakDemo {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         int score = 0;  
07         int sum = 0;  
08         int count = -1;  
09  
10         while(true){  
11             count++;  
12             sum += score;  
13             System.out.print("輸入分數(輸入-1結束):");  
14             score = scanner.nextInt();  
15  
16             if(score == -1)  
17                 break;  
18         }  
19         System.out.println("平均分數:" + (double)sum/count);  
20     }  
21 }
```



指令繼續迴圈(continue)

◆ continue 指令繼續迴圈

- 可以強制跳過這一次的迴圈執行，但不會影響之後的迴圈執行。適用在下列情況：如果學生不喜歡上某一堂課，那麼該堂課可能就會蹺課，但還是會回來上之後的課程。

```
1. class Loop_Continue{
2.     public static void main(String[] args){
3.         int musicClass = 5; //第 5 堂課是音樂課
4.
5.         for(int i=1; i<=8; i++){
6.             if(i==musicClass){
7.                 System.out.println("音樂課蹺課出去玩!");
8.                 continue;
9.             }
10.            System.out.println("開始上第" + i + "堂課");
11.        }
12.    }
13. }
```

判斷是否為音樂課

跳過這次迴圈執行

如果執行 **continue**，就會跳過這行程式碼，不予執行

-----輸出-----

開始上第1堂課
開始上第2堂課
開始上第3堂課
開始上第4堂課
音樂課蹺課出去玩！
開始上第6堂課
開始上第7堂課
開始上第8堂課

迴圈控制 – continue

```
01 public class ContinueDemo {  
02     public static void main(String[] args) {  
03         for (int i = 0; i < 10; i++) {  
04             if(i==5)  
05                 continue;  
06             System.out.println("i = " + i);  
07         }  
08     }  
09 }
```



系統管理員: 命令提示字元

```
c:\JavaClass>java ContinueDemo  
i = 0  
i = 1  
i = 2  
i = 3  
i = 4  
i = 6  
i = 7  
i = 8  
i = 9
```

迴圈控制 – break & continue

◆ break

- 離開正在執行的迴圈區塊 (while/do-while/for)

◆ continue

- 略過後面敘述,繼續迴圈循環 (while/do-while/for)

```
01 for(int i=1 ; i <= 12 ; i++) {  
02     if (i == 10) {  
03         break;  
04     }  
05     if (i%3 == 0) {  
06         continue;  
08     System.out.print(i);  
09 }
```

直接執行下一個 stepping

強制跳離 for-loop

執行結果：1 2 4 5 7 8

迴圈控制 – continue & break

```
for(int i = 0; i < 100; i++)  
{  
    if(i == 80) break; // Out of for loop  
    if(i % 9 != 0) continue; // Next iteration  
    System.out.println(i);  
}
```


迴圈控制 – Labeling

◆ 標籤 Label

- `label`本身並不是一行指令，代表的是下一行程式開始執行的地方
- 將程式區塊加上標籤(Label Name :)
- 與`break`和`continue`搭配,可指定要跳出或繼續的區段

◆ `break`、`continue` 加上 Label

- `break`搭配`label`的使用，可以用來終止更外層的迴圈
- `continue`搭配`label`的使用，會跳過以`label`標示的外層迴圈當輪的執行，直接進行該外層迴圈的下一輪

迴圈控制 – Labeling

◆ 標籤功能(label)

- **continue**和**break**作用範圍限制在所在的迴圈，如果使用巢狀式迴圈，是無法作用到更外層的迴圈；這個時候可以使用標籤將迴圈命名，便可指定**continue**或**break**要作用到哪一層迴圈。
- 假設體育課要測考1000公尺，如果跑到一半昏倒了，那接下來的課也不用上了，直接送去醫院。這時要搭配標籤功能來指定欲結束的迴圈。

迴圈控制 – Labeling

- ◆ `continue` 和 `break` 作用範圍限制在所在的迴圈。如果使用巢狀式迴圈，是無法作用到更外層的迴圈。
- ◆ 這個時候可以使用標籤將迴圈命名，便可指定`continue`或`break`要作用到哪一層迴圈。
- ◆ 假設體育課要測考1000公尺，如果跑到一半昏倒了，那接下來的課也不用上了，直接送去醫院。這時要搭配標籤功能來指定欲結束的迴圈。

```
1. class Loop_Label{
2.     public static void main(String[] args){
3.         int athClass = 5;
4.         boolean isFaint = true;
5.
6.         labelA: → 標籤與其代表的迴圈之間不可有其它程式碼
7.         for(int i=1; i<=8; i++){ → 代表一天的 8 堂課
8.             System.out.println("開始上第" + i + "堂課");
9.             if(i==athClass){
10.                 for(int d=0; d<=1000; d++){ → 代表體育課測考 1000 公尺
11.                     if(isFaint){
12.                         System.out.println("昏倒了，送醫院!");
13.                         break labelA; → 終止 labelA 所代表的迴圈
14.                     }
15.                     System.out.println("跑了" + d + "公尺");
16.                 }
17.             }
18.         }
19.     }
20. }
21. }
```

-----輸出-----

開始上第1堂課
開始上第2堂課
開始上第3堂課
開始上第4堂課
開始上第5堂課
昏倒了，送醫院！

迴圈控制 – Labeling

```
01  OuterLoop :    // 標籤名稱 (迴圈的名字)
02  for( ; ; ) {   // 外層迴圈
03      InnerLoop :
04      for( ; ; ) { // 內層迴圈
05          break InnerLoop;
06          // 程式區塊...
07          continue InnerLoop;
08          // 程式區塊...
09          break OuterLoop;
10          // 程式區塊...
11          continue OuterLoop;
12          // 程式區塊...
13      }
14  }
```

The diagram illustrates the execution flow of the code. A dotted line originates from the `break OuterLoop;` statement on line 09 and points to the `OuterLoop :` label on line 01, indicating a jump to the start of the outer loop. Another dotted line originates from the `break InnerLoop;` statement on line 05 and points to the `InnerLoop :` label on line 03, indicating a jump to the start of the inner loop. The code is enclosed in a black border, and the line numbers are listed on the left side of each line of code.

迴圈控制 - Label + break

```
01 public class BreakLabel {
02     public static void main(String[] args) {
03         label1 : {
04             for (int i = 0; i < 10; i++) {
05                 if(i==9){
06                     System.out.println("break label1");
07                     break label1;
08                 }
09             }
10             System.out.println("for迴圈之後執行");
11         } //label1區塊結尾
12     }
13 }
```

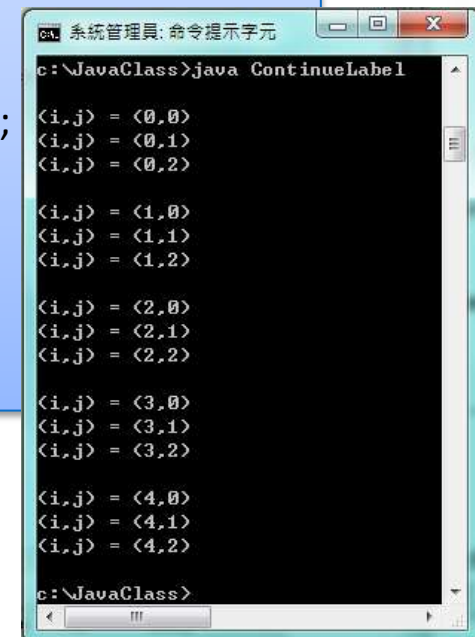


```
c:\JavaClass>
c:\JavaClass>java BreakLabel
break label1

c:\JavaClass>
```

迴圈控制 - Label + continue

```
01 public class ContinueLabel {  
02     public static void main(String[] args) {  
03         label1:  
04         for (int i = 0; i < 5; i++) {  
05             System.out.println();  
06             label2:  
07             for (int j = 0; j < 5; j++) {  
08                 if(j==3)  
09                     continue label1;  
10                 System.out.println("(i,j) = (" + i + ", " + j + ")");  
11             }  
12         }  
13     }  
14 }  
15 }
```



```
系統管理員: 命令提示字元  
c:\JavaClass>java ContinueLabel  
(i,j) = (0,0)  
(i,j) = (0,1)  
(i,j) = (0,2)  
  
(i,j) = (1,0)  
(i,j) = (1,1)  
(i,j) = (1,2)  
  
(i,j) = (2,0)  
(i,j) = (2,1)  
(i,j) = (2,2)  
  
(i,j) = (3,0)  
(i,j) = (3,1)  
(i,j) = (3,2)  
  
(i,j) = (4,0)  
(i,j) = (4,1)  
(i,j) = (4,2)  
  
c:\JavaClass>
```

Q & A