第 6 章

流程控制（二）：迴圈

認識迴圈

- 迴圈 (loop)是用來執行重複性的工作 (重複的執行動作)。
- 比如要計算 1 ~ 10 的平方和, 並將每次累加的結果都輸出到螢幕上
- 目前還沒介紹迴圈, 所以程式會是：

認識迴圈

程式 SquareSum.java 累加 1-10 的平方和

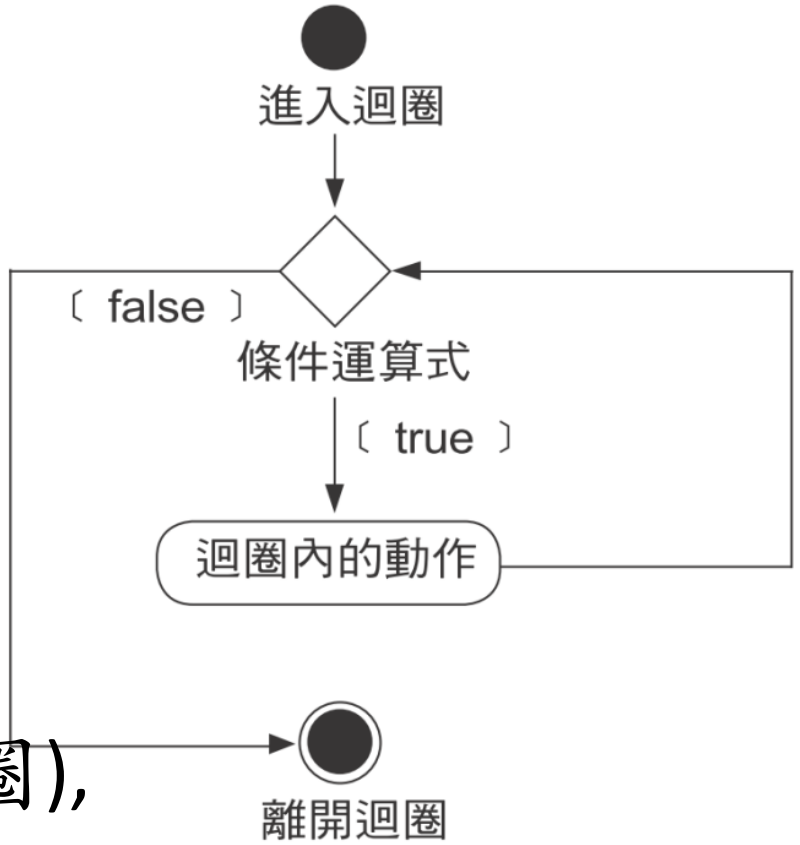
```
01 public class SquareSum {
02   public static void main(String args[]) {
03
04     int sum = 0; //儲存 1-10 的平方和累計值
05
06     sum  = 1*1;
07     System.out.println("1-1  的平方和為："+ sum);
08     sum += 2*2;
09     System.out.println("1-2  的平方和為："+ sum);
10     sum += 3*3;
11     System.out.println("1-3  的平方和為："+ sum);
.        .
.        .
.        .
24     sum += 10*10;
25     System.out.println("1-10 的平方和為："+ sum);
26   }
27 }
```

認識迴圈

- 為改進重複性程式的撰寫和執行時的效率與彈性, Java 提供了數種『迴圈』敘述, 可大幅簡化重複性程式的撰寫。

認識迴圈

- 迴圈主要是利用條件運算式的 true/false 來判斷是否要重複執行迴圈內的動作，當條件運算式為 true，程式就會執行迴圈內的動作
- 條件運算式為 false 時，就會結束迴圈 (跳出迴圈)，然後繼續往下執行。



認識迴圈

- 使用迴圈來解決上述計算平方和的問題：

程式 SquareSumLoop.java 以迴圈累加 1-10 的平方和

```
01 public class SquareSumLoop {
02   public static void main(String args[]) {
03
04     int sum = 0; //儲存 1-10 平方和累計值
05
06     for (int i=1;i<=10;i++) {   // 會重複執行區塊的內容 10 次
07       sum  += i*i;
08       System.out.println("1-" + i +" 的平方和為："+ sum);
09     }
10   }
11 }
```

執行結果

```
1-1 的平方和為：1
1-2 的平方和為：5
.
.
.
1-10 的平方和為：385
```

6-1 for 迴圈

- for 迴圈適用在需要精確控制迴圈次數的場合, 像上述計算 1 ~ 10 平方和的例子, 就是控制迴圈算到 10 便跳出迴圈。

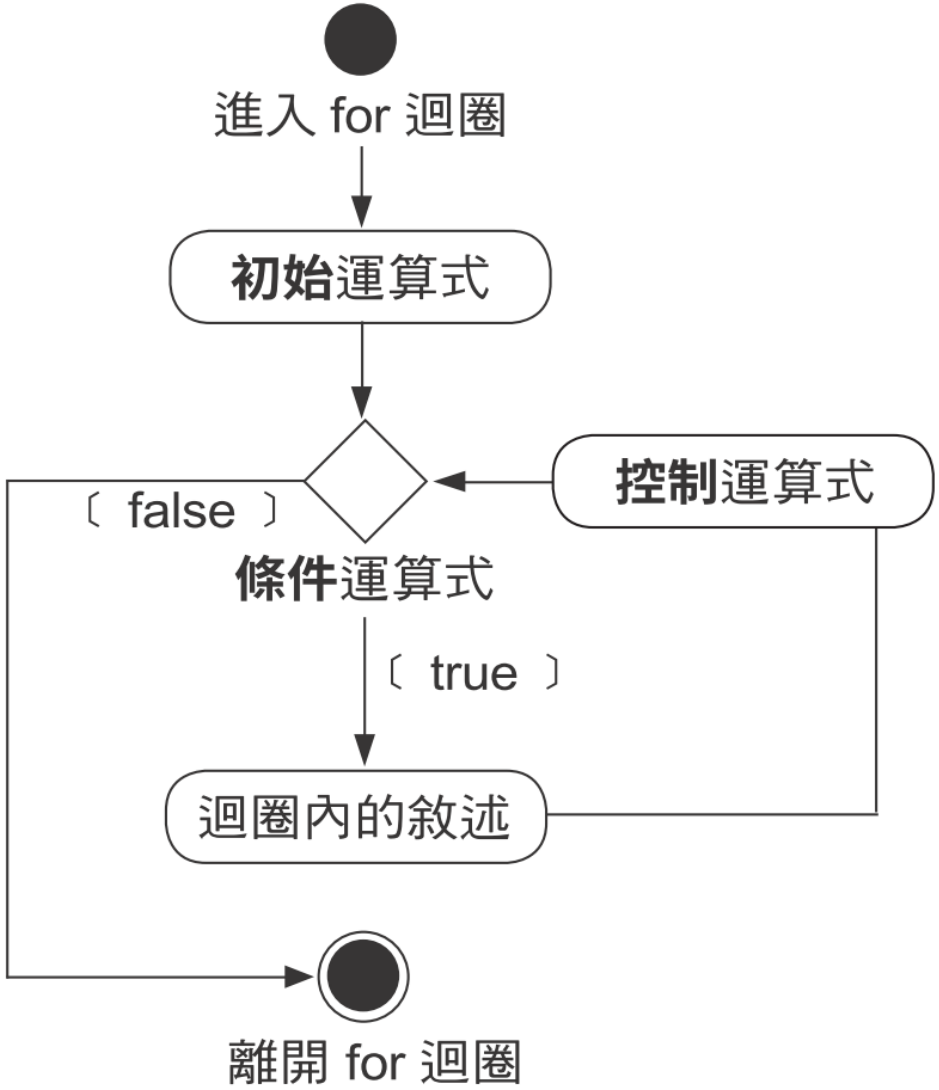
6-1-1 語法

```
for (初始運算式; 條件運算式; 控制運算式) {
    迴圈內的敘述;
}
```

6-1-1 語法

- **初始運算式：**
  條件運算式中用到的變數之初值
- **條件運算式：**
  用來判斷是否應執行迴圈中的動作敘述，傳回值需為布林值
- **控制運算式：**
  執行完for迴圈中的動作後，就會先執行此運算式
- **迴圈動作敘述：**
  利用迴圈重複執行的敘述放在此處

6-1-1 語法



6-1-2 執行流程

- for 迴圈一般都是用變數來決定執行的次數

```
第 1 次：i 值為 0 → i<3 成立  → 執行迴圈內動作 → i++ (i 變成 1)
          (i = 0)

第 2 次：i 值為 1 → i<3 成立  → 執行迴圈內動作 → i++ (i 變成 2)

第 3 次：i 值為 2 → i<3 成立  → 執行迴圈內動作 → i++ (i 變成 3)

第 4 次：i 值為 3 → i<3 不成立 → 跳出迴圈
```

- 善用迴圈的條件運算式及控制運算式，就可以控制迴圈的執行次數。

6-1-2 執行流程

程式 CountOdd.java 計算在某範圍內的所有奇數和

```
01 import java.util.*;
02
03 public class CountOdd {
04
05   public static void main(String args[]) {
06
07     // 宣告累加值 sum 及計算範圍 range
08     int sum = 0, range, i;
09
10    System.out.print("請輸入欲計算的奇數和範圍 (結尾數值)：");
11    Scanner sc = new Scanner(System.in);
12    range = sc.nextInt();
13
14    // 由 1 開始, 每次加 2 直到 i 大於 range 的 for 迴圈
15    for (i=1; i<=range; i+=2) {   // 每跑一次迴圈就將 i 值加 2
```

6-1-2 執行流程

```
16          sum += i;
17       }
18       System.out.println("1 到 "+range+" 的所有奇數和為 "+sum);
19    }
20 }
```

執行結果 1

```
請輸入欲計算的奇數和範圍(結尾數值)：15
1 到 15 的所有奇數和為 64
```

執行結果 2

```
請輸入欲計算的奇數和範圍(結尾數值)：1000
1 到 1000 的所有奇數和為 250000
```

6-1-3 for 迴圈的進階用法

- 在 for 迴圈的初始運算式中,也可以直接宣告新的變數來使用

```
int sum = 0, range; // 不用宣告迴圈的變數
...
for (int i=1; i<=range; i+=2) {   // 宣告及初始化迴圈變數 i
```

6-1-3 for 迴圈的進階用法

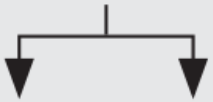
- 此種作法建立的變數 i,
  只能在 for 迴圈中存取；

```
for(int i=1; i<5; i++) {
    System.out.println(i); // 正確, i 在迴圈中可以存取
}
System.out.println(i);     // 錯誤, 離開迴圈後 i 就消失了, 因此不可存取！
```

6-1-3 for 迴圈的進階用法

- 在初始運算式及控制運算式中也可以包含多個以逗號分隔的運算式

在初始運算式中宣告 i 和 j 並指定初值



```
for(int i=1, j=2; i<5 && j>0; i++, j--)
   ...


int i, j, k;
for(i=1, j=2; i<5 && j>0; i++, j--, k=i+j)
   ...
```

6-1-3 for 迴圈的進階用法

- for 迴圈的初始、控制、及條件運算式都不是必要的，若不需要可以留白

```
for(int i; i>1; )   // 省略控制運算式
   ...

for( ;x<10; )       // 只剩條件運算式
   ...

for( ; ; )   // 全部省略，但因沒有要檢查的條件，會變成無窮迴圈，
   ...       // 必須用其他方式中斷迴圈 (中斷方式參見 6-5 節)
```

6-1-4 for-each 迴圈

- for-each 迴圈可針對陣列或集合中的每一個元素，每次取出一個來進行迴圈處理。

```
int a[] = { 1,2,3,4,5 };   // 宣告內含 5 個元素的 a 陣列

for (int e: a)   // 每次由 a 中取出一個元素存入 e, 然後執行迴圈
   System.out.print(e);
```

執行結果：

```
12345   ← 每迴圈印出一個元素，共印了 5 次
```

6-2 while 迴圈

6-2-1 語法

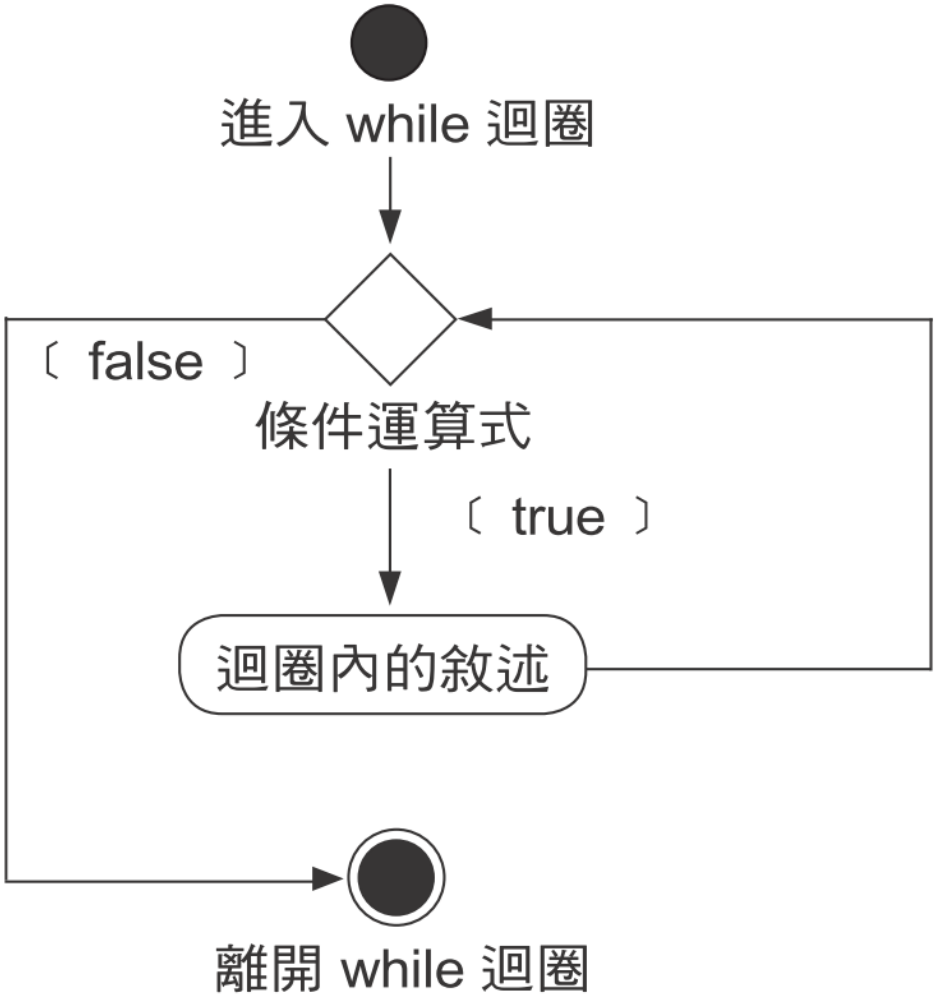
- while 迴圈
  只要條件運算式即可。

```
while (條件運算式) {
    迴圈內的敘述
}
```

6-2-1 語法

- **while：**
  根據條件式的真假，
  來決定是否執行迴圈內的動作
- **條件運算式：**
  任何結果為布林值的運算式或布林變數

6-2-2 執行流程



6-2-2 執行流程

程式 CountEven.java 利用 while 迴圈計算某範圍內的偶數和

```
01 import java.io.*;
02
03 public class CountEven {
04
05   public static void main(String args[]) throws IOException {
06
07     // 宣告累加值 sum 及計算範圍 range
08     int sum = 0, range;
09
10     System.out.print("請輸入欲計算的偶數和範圍(結尾數值)：");
11
12     BufferedReader br =
13       new BufferedReader(new InputStreamReader(System.in));
14     String str = br.readLine();
15     range = Integer.parseInt(str);
```

6-2-2 執行流程

```
16
17      int i=0;                  // 宣告迴圈變數 i
18      while (i<=range) {  // 當 i 值大於 range 即停止執行的 while 迴圈
19        sum += i;             // 每次進入迴圈時, 將 sum 的值加上 i
20        i+=2;                 // 每次都將 i 值加 2
21      }
22      System.out.println("1 到 "+range+" 的所有偶數和為 "+sum);
23    }
24  }
```

執行結果

```
請輸入欲計算的偶數和範圍(結尾數值)：2020
1 到 2020 的所有偶數和為 1021110
```

6-2-2 執行流程

- 第 20 行的 i+=2; 有點類似 for 迴圈的控制運算式, 讓 while 迴圈的條件運算式有可能產生 false 的結果。
- 如果把這行敘述拿掉, while 迴圈內的條件運算式狀況將不會改變 (永遠是 true), 稱為無窮迴圈。

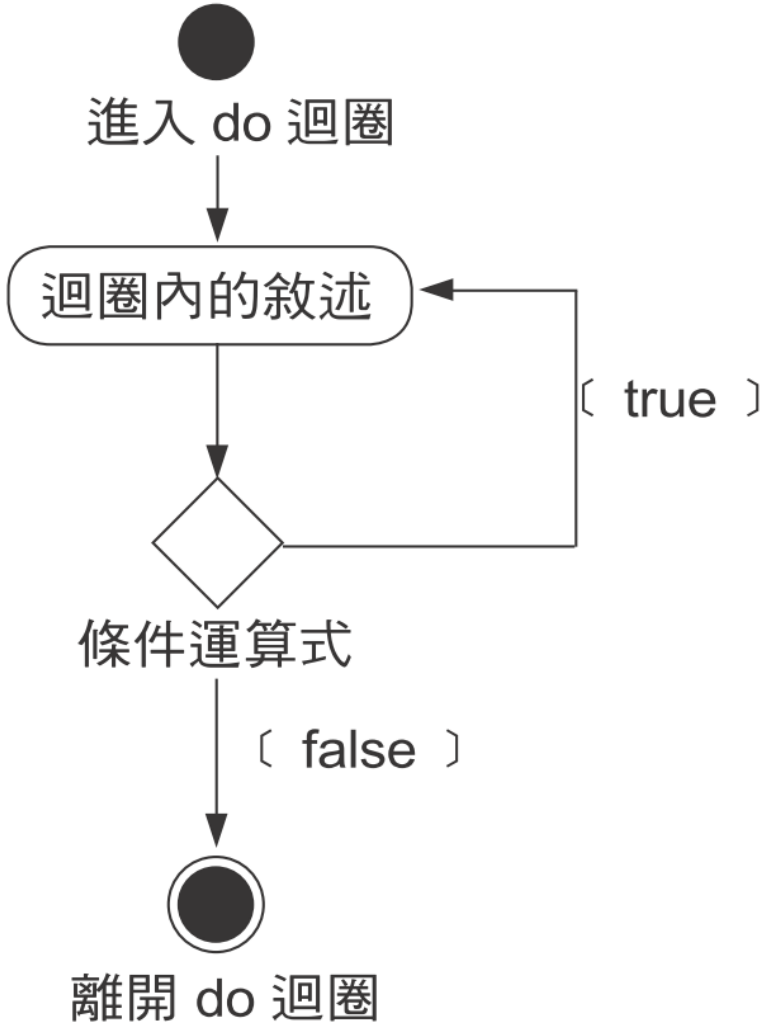
6-3 do/while 迴圈

- do/while 迴圈是 while 的一種變型。
- do/while 迴圈是先執行完迴圈內的敘述後，再檢查條件是否成立。
- do/while 迴圈的特點就是：
  不論條件式為何，
  迴圈敘述至少都會執行一次

6-3-1 語法

```
do {
        迴圈內的敘述
} while (條件運算式);    // 結尾要加分號
```

6-3-2 執行流程



6-3-2 執行流程

程式 CountWhile.java 用 while 測試迴圈執行的次數

```
01 public class CountWhile {
02
03   public static void main(String args[]) {
04
05     int i=0;              // 宣告用來記錄迴圈執行次數的變數 i
06     while (i++<3)
07       System.out.println("這是第" + i + "次執行迴圈");
08   }
09 }
```

執行結果

```
這是第1次執行迴圈
這是第2次執行迴圈
這是第3次執行迴圈
```

6-3-2 執行流程

程式 CountDowhile.java 用 do/while 測試迴圈執行的次數

```
01 public class CountDowhile {
02
03   public static void main(String args[]) {
04
05     int i=0;              // 宣告用來記錄迴圈執行次數的變數 i
06     do {
07        System.out.println("這是第" + i + "次執行迴圈");
08      } while (i++<3); // 在 while() 的結尾要記得加分號！
09   }
10 }
```

執行結果

```
這是第0次執行迴圈
這是第1次執行迴圈
這是第2次執行迴圈
這是第3次執行迴圈
```

6-4 巢狀迴圈

- 巢狀迴圈就是迴圈的大括號之中，還有其它迴圈

程式 Count9x9.java 利用巢狀迴圈輸出九九乘法表

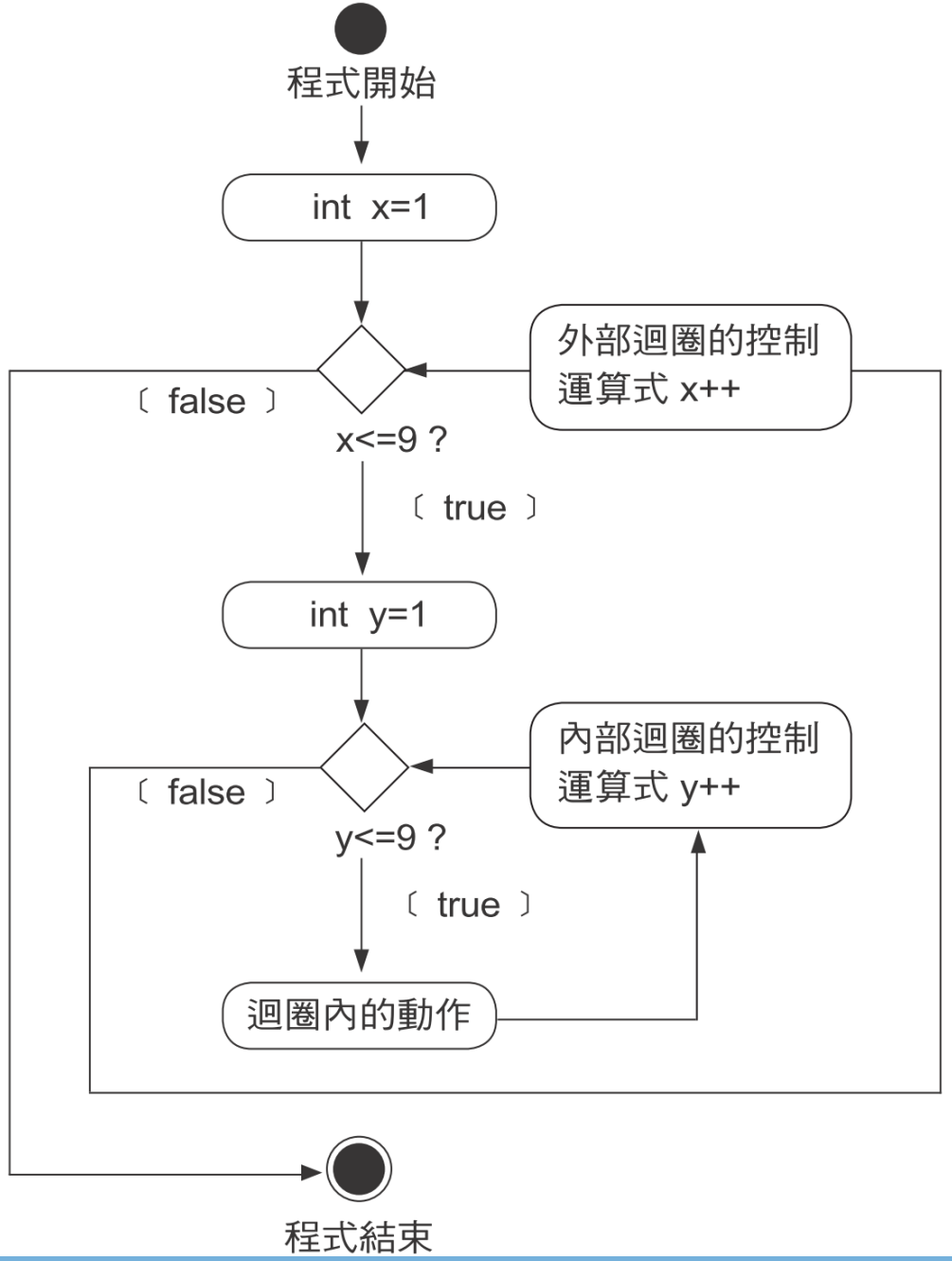
```
01 public class Count9x9 {
02
03   public static void main(String args[]) {
04
05     for (int x=1; x<=9; x++) {   // 外層迴圈從 x=1 開始
06       for (int y=1; y<=9; y++) {   // 內層迴圈從 y=1 開始
07         System.out.print( x + "*" + y + "=" + x*y + "\t");
08       }
09       System.out.println();   // 換行
10     }
11   }
12 }
```

6-4 巢狀迴圈

執行結果

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

巢狀迴圈


程式開始
int x=1
外部迴圈的控制
運算式 x++
〔 false 〕
x<=9 ?
〔 true 〕
int y=1
內部迴圈的控制
運算式 y++
〔 false 〕
y<=9 ?
〔 true 〕
迴圈內的動作
程式結束

6-4 巢狀迴圈

內迴圈控制橫向的數字增加 →

外迴圈控制縱向的數字增加 ↓

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

- 有兩個敘述：break 及 continue,
  都可以變更迴圈的執行流程,
  而跳出執行迴圈或跳到下一輪迴圈。

6-5-1 跳出迴圈的 break

- 當程式中遇到某種狀況而不要繼續執行迴圈時, 即可用 break 來跳出迴圈。

程式 UseBreak.java 使用 break 跳出無窮迴圈

```
01 public class UseBreak {
02
03   public static void main(String args[]) {
04
05     int i=1;
06
07     while (i>0) { // 無窮迴圈
08       System.out.println("無窮迴圈執行中..");
09       if (i == 5) // 當 i 為 5 時, 條件運算式成立
10         break;    // 跳出迴圈
11       i++;
```

6-5-1 跳出迴圈的 break

```
12        }
13        System.out.println("成功的跳出迴圈了！！");
14    }
15 }
```

執行結果

```
無窮迴圈執行中..
無窮迴圈執行中..
無窮迴圈執行中..
無窮迴圈執行中..
無窮迴圈執行中..
成功的跳出迴圈了！！
```

← 這行訊息僅出現 5 次，表示迴圈只執行了 5 次

6-5-2 跳到下一輪迴圈的 continue

- continue 會跳出『這一輪』迴圈，然後繼續下一輪迴圈。

程式 UseContinue.java 使用 continue 來跳到下一輪迴圈

```
13      // 由 1 開始, 每次加 1
14      for (i=1; i<=range; i++)   {
15        if(i%2==0) continue;     // 若是偶數就跳到下一輪迴圈
16        sum += i;                // 奇數才會被累加
17      }
```

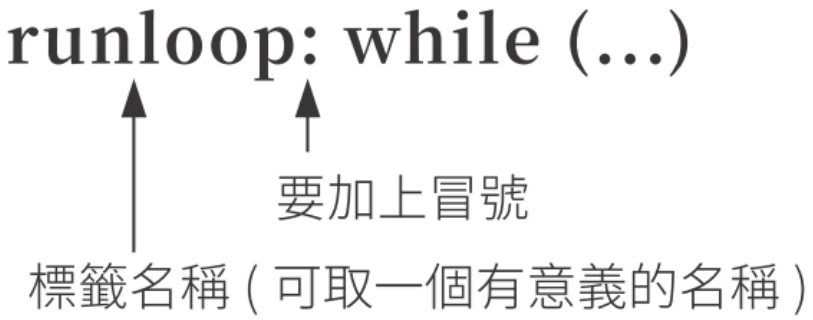
執行結果

```
請輸入欲計算的奇數和範圍 (結尾數值)：199
1 到 199 的所有奇數和為 10000
```

6-5-3 標籤與 break/continue 敘述

- 巢狀迴圈要由內層迴圈直接跳出、或跳到下一輪的外層迴圈時，必須在每一層的迴圈中，都加上 break/continue
- Java 提供在 break/continue 之後加上標籤(Label)的語法

6-5-3 標籤與 break/continue 敘述




```
runloop: while (...) {
   for (...) {
      do (...) {
         ...
         break runloop;
         ...
```

中斷最外層的迴圈

標籤與 break/continue 敘述

程式 PartOf9x9.java 只輸出部分九九乘法表內容

```
01 public class PartOf9x9 {
02
03   public static void main(String args[]) {
04
05     outloop: for (int x=1; x<=9; x++) {    // 加上標籤名稱
06       for (int y=1; y<=9; y++) {
07         if (x*y > 25) {              // 若乘積大於 25
08           System.out.println();     // 換行
09           continue outloop;         // 跳到下一輪的 outloop 迴圈
10         }
11         System.out.print( x + "*" + y + "=" + x*y + "\t");
12       }
```

6-5-3 標籤與 break/continue 敘述

```
13          System.out.println();
14        }
15      }
16  }
```

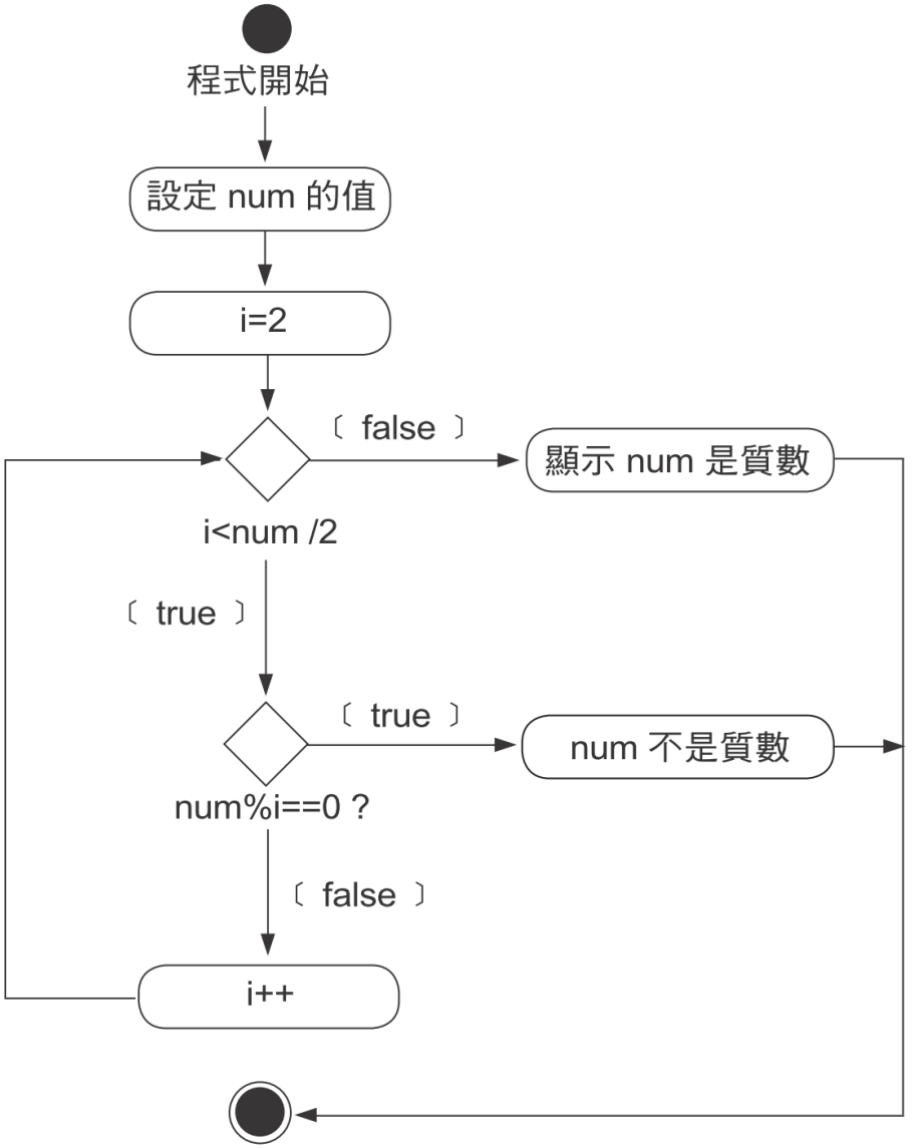
執行結果

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25
6*1=6   6*2=12  6*3=18  6*4=24
7*1=7   7*2=14  7*3=21
8*1=8   8*2=16  8*3=24
9*1=9   9*2=18
```

6-6 綜合演練

6-6-1

迴圈與 if 條件式混合應用：判斷質數



取出 1 到指定數值之間的質數

程式 IsPrime.java 判斷某數是否為質數

```
01 import java.io.*;
02
03 public class IsPrime {
04   public static void main(String args[]) throws IOException {
05
06     BufferedReader br =
07      new BufferedReader(new InputStreamReader(System.in));
08
09     while(true){                    // 讓使用者可反覆輸入新數值的迴圈
10       System.out.print("請輸入要檢查的數 (輸入 0 結束)：");
11
12       String str = br.readLine();
13       int num = Integer.parseInt(str);
14       if(num == 0) break;          // 若輸入 0 即跳出迴圈, 結束程式
15
```

取出 1 到指定數值之間的質數

```
16        boolean isPrime = true;  // 表示數值是否為質數的布林值
17        double range = num/2.0;  // 限定除數的範圍
18
19        for (int i=2; i<=range; i++) {   // 做除法運算的迴圈
20          if ((num%i) == 0) {            // 餘數為 0 表示可以整除
21            if (isPrime == true) {
22              isPrime = false;           // 非質數, 並輸出目前的除數
23              System.out.print(num +" 不是質數, 可被 "+i);
24            }
25            else {                       // 輸出目前的除數
26              System.out.print(" "+i);
27            }
28          }
29        }
30        // 檢查完畢, 依檢查結果輸出不同的訊息
```

取出 1 到指定數值之間的質數

```
31          if (isPrime) {              // 若是質數, 即輸出該數值
32            System.out.println(num +" 是質數");
33          }
34          else {
35            System.out.println(" 整除");
36          }
37        }
38     }
39  }
```

執行結果

```
請輸入要檢查的數 (輸入 0 結束):399          ←— 輸入非質數
399 不是質數, 可被 3 7 19 21 57 133 整除
請輸入要檢查的數 (輸入 0 結束):199          ←— 輸入質數
199 是質數
請輸入要檢查的數 (輸入 0 結束):0            ←— 輸入 0 可結束程式
```

6-6-2 Scanner 類別的輸入檢查

- 若使用者輸入非預期的資料，
  程式會發生例外 (Exception) 並中止執行

```
hasNextByte();                  hasNextInt();
hasNextBoolean();               hasNextLong();
hasNextDouble();                hasNextShort();
hasNextFloat();                 hasNext();    // 判斷是否有字串
```

6-6-2 Scanner 類別的輸入檢查

程式 HasNext.java 檢查輸入的內容

```
01 import java.util.*;
02
03 public class HasNext {
04   public static void main(String args[]) {
05     // 宣告累加值 sum, 計算範圍 range, 迴圈變數 i
06     int sum = 0, range, i;
07     Scanner sc = new Scanner(System.in);
08     System.out.print("請輸入欲計算的奇數和範圍 (結尾數值)：");
09
10     while(!sc.hasNextInt()) {          // 輸入非整數, 就執行迴圈
11       System.out.print("請輸入整數：");
12       sc.next();                       // 清除剛剛輸入的內容
13     }
14
15     range = sc.nextInt();              // 讀取整數值
16
```

6-6-2 Scanner 類別的輸入檢查

```
17      // 由 1 開始, 每次加 2 直到 i 值大於 range 的 for 迴圈
18      for (i=1; i<=range; i+=2) { // 每跑一次迴圈就將 i 值加 2
19        sum += i;
20      }
21
22      System.out.println("1 到 "+range+" 的所有奇數和為 "+sum);
23    }
24  }
```

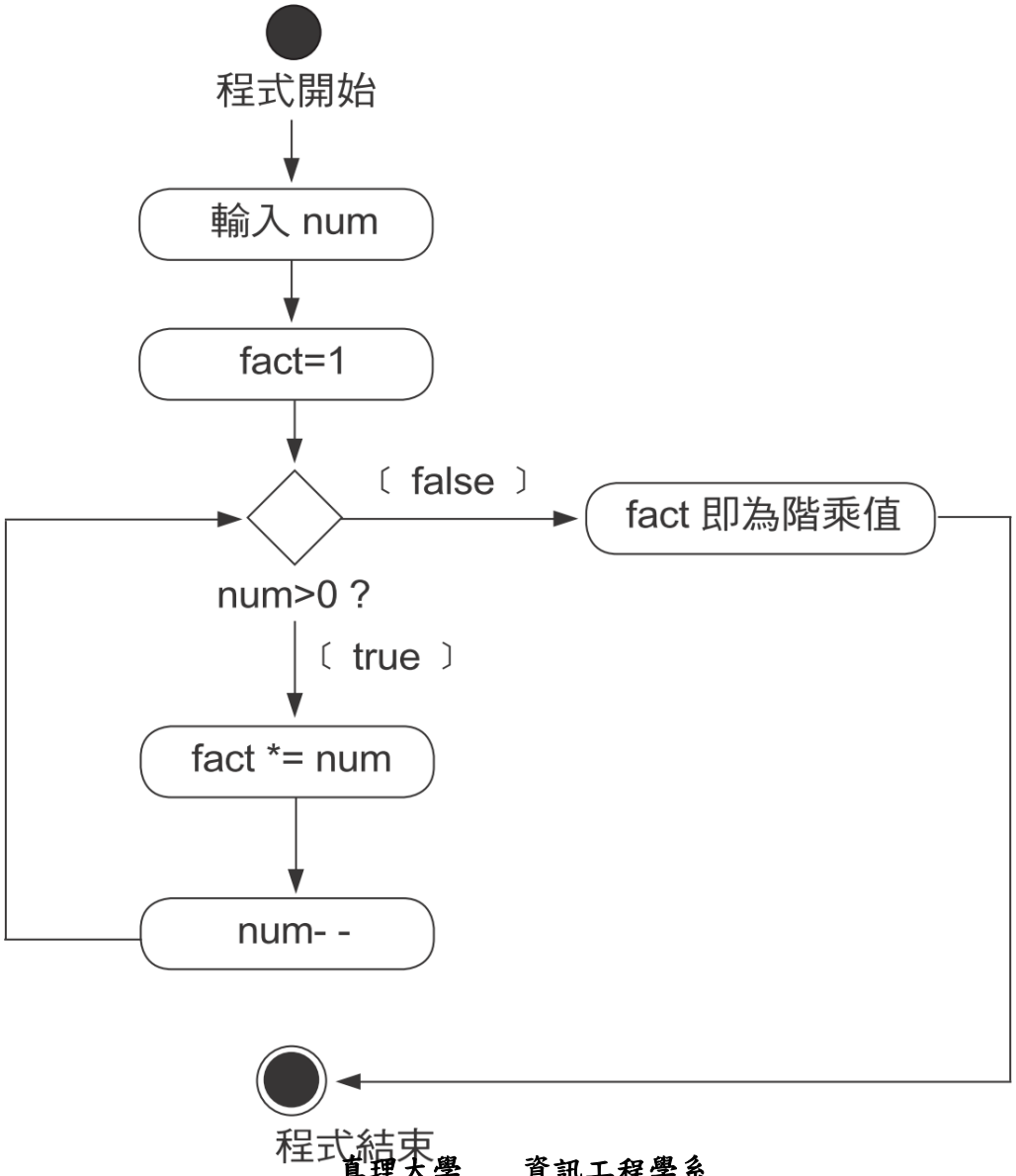
執行結果

```
請輸入欲計算的奇數和範圍 (結尾數值)：十
請輸入整數：ten
請輸入整數：10.0
請輸入整數：10
1 到 10 的所有奇數和為 25
```

輸入非整數時，程式都會要求再次輸入

6-6-3 各種迴圈的混合應用：計算階乘

```
N! = 1*2*3*...*(N-2)*(N-1)*N
```


程式開始
輸入 num
fact=1
〔 false 〕
fact 即為階乘值
num>0 ?
〔 true 〕
fact *= num
num- -
程式結束

程式 Factorial.java 計算使用者輸入數字的階乘值

```
01 import java.io.*;
02
03 public class Factorial {
04   public static void main(String args[]) throws IOException {
05
06     BufferedReader br =
07       new BufferedReader(new InputStreamReader(System.in));
08
09     while(true) {
10       System.out.println("請輸入 1-170 間的整數來計算階乘");
11       System.out.print("(輸入 0 即結束程式)：");
12       String str = br.readLine();
13       int num = Integer.parseInt(str);
```

```
14          if (num == 0)
15            break;                    // 若使用者輸入 0, 就跳出迴圈
16          else if (num>170)
17            continue;                 // 若輸入大於 170, 則重新輸入
18
19          System.out.print(num + "! 等於 ");
20
21          double fact;                // 用來儲存、計算階乘值的變數
22          for(fact=1;num>0;num--)     // 計算階乘的迴圈
23            fact *= num;              // 每輪皆將 fact 乘上目前的 num
24
25          System.out.print(fact + "\n\n"); // 輸出計算所得的階乘值
26        }
27    }
28 }
```

執行結果

```
請輸入 1-170 間的整數來計算階乘
(輸入 0 即結束程式)：199        ◄── 輸入數字大於 170 時會要求重新輸入
請輸入 1-170 間的整數來計算階乘
(輸入 0 即結束程式)：99
99! 等於 9.33262154439441E155

請輸入 1-170 間的整數來計算階乘
(輸入 0 即結束程式)：0
```