第四章 流程控制

課前指引

最直覺的程式執行方式當然是一行一行的往下執行，這種結構稱之為**循序結構**。除此之外，任何一種基本程式語言都必須具有其他種類的**程式流程控制能力**，否則將無法提供各類變化的執行結果。

這些非循序結構的流程控制是由**決策**與**跳躍**組成，而決策與跳躍也可以組合成**迴圈**。在一般的結構化程式語言中，都不允許使用者自行定義跳躍，而把跳躍移到**決策**與**迴圈**之內。

章節大綱

- 4.1 Java程式的邏輯結構
- 4.2 選擇性敘述
- 4.3 迴圈敘述
- 4.4 巢狀與縮排

備註：可依進度點選小節

4.1 Java程式的邏輯結構

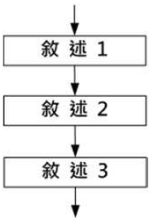
- Java的流程控制結構有以下三種：
  1. 循序結構(Sequence Structure)
  2. 選擇結構(Selection Structure)
  3. 迴圈結構(Loop Structure)
     - 這三種結構都只有一個進入點及一個出口點，這樣的特性也使得程式較容易維護與理解。
- 不允許使用Goto（無條件跳躍）指令。

3

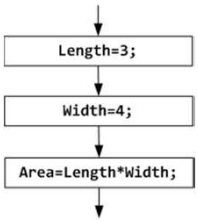
4.1.1『循序』結構

- 『循序』結構
  - 『程式碼被執行的順序為由上而下，一個敘述接著一個敘述依序執行』。
  - 在第三章之前的範例只使用了『循序』結構。

結構：



範例：



4

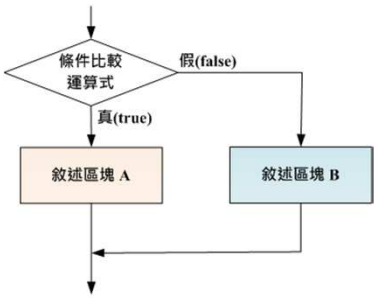
4.1.2『選擇』結構

- 『選擇』結構
  - 代表程式在執行時，會依據條件（運算式的結果）適當地改變程式執行的順序。
    - 當滿足條件時，會執行某一敘述區塊（通常是接續的敘述區塊）；
    - 若條件不滿足時，則執行另一敘述區塊。
  - 可以分為三種：
    - 單一選擇、雙向選擇、多向選擇。
    - 以雙向選擇為例，它能夠指定條件成立時要執行的敘述區塊，也能指定條件不成立時要執行的敘述區塊。敘述區塊執行完畢將會繼續執行選擇結構之後的敘述。

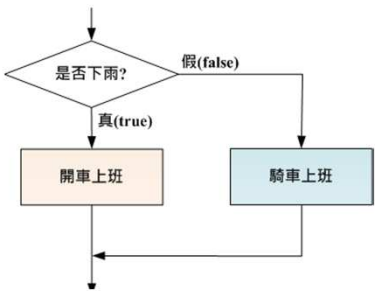
5

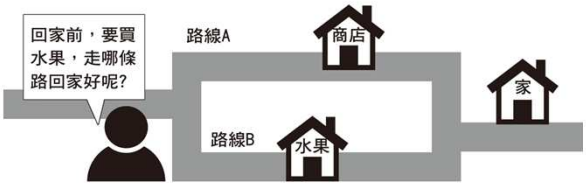
4.1.2『選擇』結構

雙向選擇結構：



範例：





6

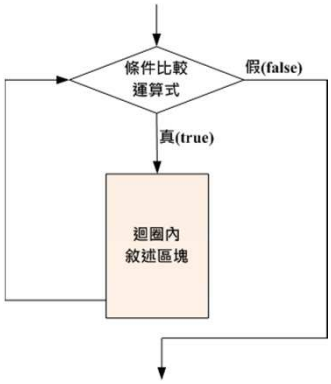
4.1.3『迴圈』結構

- 『迴圈』結構又稱為重複結構(iteration structure)
  - 代表電腦會重複執行某一段敘述區塊，直到某個條件成立或不成立時，重覆動作才會停止。
  - 可以分為三種：
    - 計數迴圈、前測式迴圈與後測式迴圈。
  - 以前測式迴圈為例
    - 它會先測試條件，若條件為真，才執行敘述區塊（即迴圈內敘述），當敘述區塊執行完畢後，會再回到測試條件重新測試條件，若條件仍舊成立，則再一次執行敘述區塊。
    - 如此反覆『測試』、『執行敘述區塊』，直到條件不成立時才會離開迴圈。
    - 因此，在執行敘述區塊時，應該要有改變測試條件成立與否的機會，否則將造成無窮迴圈（迴圈永不停止）。

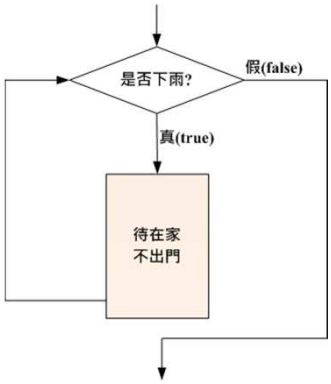
7

4.1.3『迴圈』結構

前測式迴圈：



範例：



8

4.1.4流程控制敘述

- 流程控制敘述(Control flow statements)可以分為下列兩大類：
  - 選擇性敘述(decision-making statements)：
    - 包含if, if-else, switch等種類的敘述。
  - 迴圈型敘述(looping statements)：
    - 包含for, while, do-while等種類的敘述。
- 具有分支特性的敘述(branching statements)
  - break, continue, return等敘述

9

4.2『選擇性』敘述

- Java支援選擇結構
  - 包含單一選擇、雙向選擇、多向選擇
  - 整體分類如圖4-1所示。

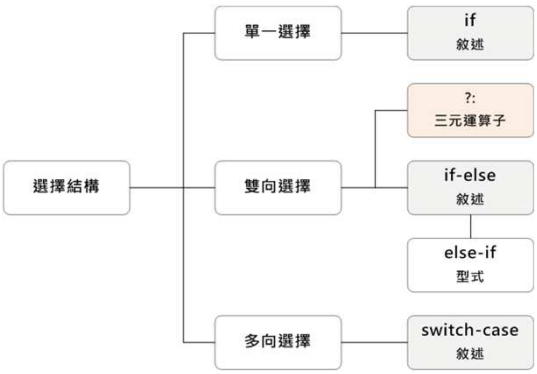


圖4-1 Java的選擇敘述種類

10

4.2.1單一選擇敘述（if敘述）

- if敘述
  - 當某個**條件運算式**成立時，就去做某件事（或某些事），當條件運算式不成立時，就不會做這些事
  - 下面是一個生活實例。

```
if(下雨)
  打傘;
```

  - 在Java中，只不過將『下雨』使用條件運算式來表達。把『打傘』用敘述來表達。
  - 例如：if(A>1)
    B=100;
    ，就是當A>1時，將B的變數值指定為100。

11

4.2.1單一選擇敘述（if敘述）

- 如果要做的事不只一件，

```
if(感冒了)
{
   多喝水;
   多休息;
}
```

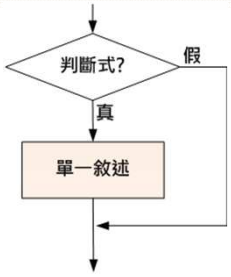
  - 從｛ 到 ｝之內的所有事情
    - 就是當if之後的條件運算式邏輯成立時所要做的事情
  - 以下是if敘述的完整語法。

12

4.2.1單一選擇敘述（if敘述）

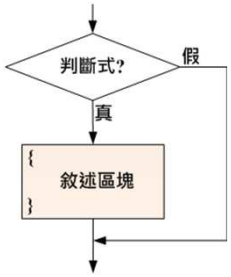
- 單一選擇敘述（if敘述）
  - if敘述語法一：

```
if(條件運算式)
    要執行的單一敘述
```



  - if敘述語法二：

```
if(條件運算式)
{
  ……敘述區塊……
}
```



13

4.2.1單一選擇敘述（if敘述）

Coding 注意事項

條件運算式代表的是「運算式的結果為布林值」。它可能是由比較運算子所構成，例如(a>b)。也可能包含邏輯運算，例如((a>b) && (a<c))，甚至可能是單純的布林變數。但它與一般的運算式不同，例如您不能撰寫為if(a=1)，因為a=1是一般運算式，它並不會產生布林值

如果想要將條件設定為a是否等於1，則應該撰寫為if(a==1)。

以上有一個例外，也就是當『=』左邊的運算元為布林變數時，則編譯器並不會產生錯誤，此時若誤用『=』與『==』則可能與預期的狀況有所不同，詳見範例4-3。

因此，為了與一般的運算式區別，本書將**條件運算式**稱為**判斷式**。

14

4.2.1單一選擇敘述（if敘述）

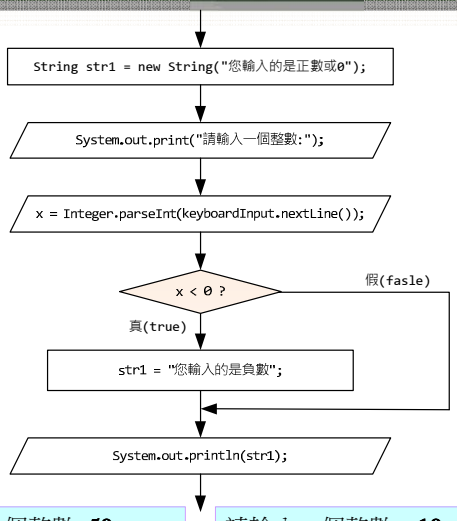
- 範例4-1：ch4_01.java(光碟 myJava\ch04\ch4_01.java)

```
1   /*  檔名:ch4_01.java      功能:if選擇性敘述範例   */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_01          //主類別
8   {
9       public static void main(String args[])
10      {
11          Scanner keyboardInput = new Scanner(System.in);
12          int x;
13          String str1 = new String("您輸入的是正數或0");
14
15          System.out.print("請輸入一個整數:");
16          x = Integer.parseInt(keyboardInput.nextLine());
17          if(x<0)
18              str1 = "您輸入的是負數";
19          System.out.println(str1);
20      }
21  }
```

15

4.2.1單一選擇敘述（if敘述）

- **對應流程圖：**



- 執行結果：

請輸入一個整數:**50** 您輸入的是正數或0	請輸入一個整數:**-10** 您輸入的是負數

16

4.2.1單一選擇敘述（if敘述）

- 範例說明：
  - (1)在第13行宣告並設定字串str1內容為『您輸入的是正數或0』。
  - (2)第16行的變數x將儲存使用者輸入的整數。其中keyboardInput.nextLine()為鍵盤輸入的字串，經過轉型後指定給變數x。
  - (3)第17~18行即為if敘述，『x<0』為條件運算式，『str1="您輸入的是負數";』為條件運算式成立時，要執行的單一敘述。
    - 因此，只有當x<0成立時，str1的字串內容才會變成『您輸入的是負數』
    - 否則，str1的內容將維持在『您輸入的是正數或0』。

17

4.2.1單一選擇敘述（if敘述）

- 【實用範例4-2】：根據購買入場卷的數量是否大於10張，決定是否打折優待。
  - **範例4-2：**ch4_02.java（隨書光碟 myJava\ch04\ch4_02.java）

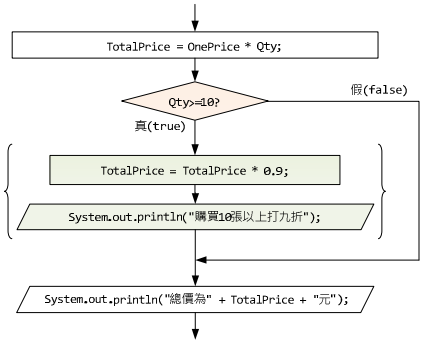
```
1   /*  檔名:ch4_02.java       功能:if選擇性敘述範例    */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_02          //主類別
8   {
9       public static void main(String args[])
10      {
11          Scanner keyboardInput = new Scanner(System.in);
12          int OnePrice = 200,Qty;
13          double TotalPrice;
14
15          System.out.println("每張入場卷價格為" + OnePrice + "元");
16          System.out.print("請輸入您要購買的張數:");
17          Qty = Integer.parseInt(keyboardInput.nextLine());
18          System.out.println("=======================");
```

18

4.2.1單一選擇敘述（if敘述）

```
19          TotalPrice = OnePrice * Qty;
20          if(Qty>=10)
21          {
22              TotalPrice = TotalPrice * 0.9;
23              System.out.println("購買10張以上打九折");
24          }
25          System.out.println("總價為" + TotalPrice + "元");
26      }
27  }
```

- **對應流程圖：**



19

4.2.1單一選擇敘述（if敘述）

- 執行結果：

```
每張入場卷價格為200元
請輸入您要購買的張數:5
============================
總價為1000.0元
```

```
每張入場卷價格為200元
請輸入您要購買的張數:15
============================
購買10張以上打九折
總價為2700.0元
```

- 範例說明：
  - (1)第20~24行是if敘述，由於當條件運算式成立時要執行的運算式敘述超過一個，因此將之組合成敘述區塊（第21~24行）。
  - (2)在執行結果中，當使用者輸入『5』時，判斷式『Qty>=10』不成立，因此第21~24行不會被執行。當使用者輸入『15』時，判斷式『Qty>=10』成立，因此第21~24行會被執行。而第25行是一定會被執行的，因為它並非if敘述的一部份。

20

4.2.1單一選擇敘述（if敘述）

- 【觀念範例4-3】：條件運算式的成立。

```
1   /*  檔名:ch4_03.java        功能:條件運算式範例    */
2
3   package myJava.ch04;
4   import java.lang.*;
5
6   public class ch4_03          //主類別
7   {
8       public static void main(String args[])
9       {
10              boolean x=false;
11              String str1 = new String("x為真");
12              if(x=false)
13                 str1 = "x為假";
14              System.out.println(str1);
15      }
16  }
```

執行結果

```
x為真
```

必定不成立，因為並非使用比較運算子，而是使用設定運算子，且x被設定為false。

21

4.2.1單一選擇敘述（if敘述）

- 範例說明：
  - (1)這一個範例的重點在於第12行if敘述的條件運算式。當我們在條件運算式（判斷式）中使用『比較運算子』時，事實上並不會真的影響變數值，而只會做運算式成立與否的檢查。
    - 在這個範例中，由於第12行執行時，x被設定為false，因此第13行不會被執行，故str1維持宣告時的內容，即"x為真"。
  - (2)本範例若希望是判定x在第11行之前是否為false，則應該要將第12行改寫為『if(x==false)』，如此，str1才會被設定為"x為假"。

22

4.2.1單一選擇敘述（if敘述）

老師的叮嚀

運算式可以由其他較小的運算式組成，條件運算式（判斷式）也同樣可以用其他較小的條件運算式組成，例如：((a>=10) && (a=<20))，只有在a介於10~20之間時，該條件運算式才會成立。

Coding 注意事項

由於呼叫函式會產生一個回傳值，如果這個回傳值的資料型態為布林型態，也可以充當條件運算式。
例如在內建類別Character中，有一個static方法isLowerCase，可以判斷字元是否為英文小寫字母，它會回傳一個布林值，若回傳true，則代表輸入的為英文小寫字母，否則回傳false。故下列程式是合法的。

```
if(Character.isLowerCase('m'))
    System.out.println("這是一個小寫英文字母");
```

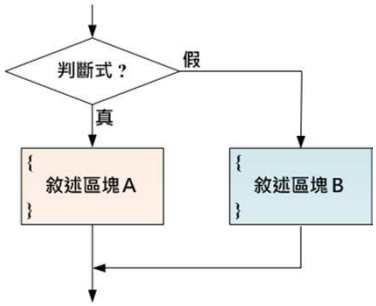
23

4.2.2雙向選擇敘述（if-else敘述）

- 使用if敘述無法在『當條件運算式不成立』時，指定要執行的動作。而if-else敘述就可以在『條件運算式不成立』的狀況下，執行某些指定的程式碼，其語法如下。
  - if-else敘述語法：

```
if(條件運算式)
{
  條件成立時要執行的敘述區塊A
}
else
{
  條件不成立時所執行的敘述區塊B
}
```

if-else敘述流程圖



24

4.2.2雙向選擇敘述（if-else敘述）

- 【語法說明】：
  - 當敘述區塊僅僅只有一個敘述時，可以省略外層的『{}』。
- 【實用範例4-4】：
  - 根據購買入場卷的數量是否大於10張，決定是否打折優待。
  - 範例4-4：ch4_04.java（隨書光碟 myJava\ch04\ch4_04.java）

25

4.2.2雙向選擇敘述（if-else敘述）

```
1   /*  檔名:ch4_04.java      功能:if-else選擇性敘述範例   */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_04         //主類別
8   {
9       public static void main(String args[])
10      {
11          Scanner keyboardInput = new Scanner(System.in);
12          int OnePrice = 200,Qty;
13          double TotalPrice;
14
15          System.out.println("每張入場卷價格為" + OnePrice + "元");
16          System.out.print("請輸入您要購買的張數:");
17          Qty = Integer.parseInt(keyboardInput.nextLine());
18          System.out.println("======================");
19          TotalPrice = OnePrice * Qty;
```

26

4.2.2雙向選擇敘述（if-else敘述）

```
20          if(Qty>=10)
21          {
22             TotalPrice = OnePrice * Qty * 0.9;
23             System.out.println("購買10張以上打九折");
24          }
25          else
26          {
27             TotalPrice = OnePrice * Qty;
28             System.out.println("您未購買10張以上的入場券,恕不打折");
29          }
30          System.out.println("總價為" + TotalPrice + "元");
31       }
32    }
```

- 執行結果：

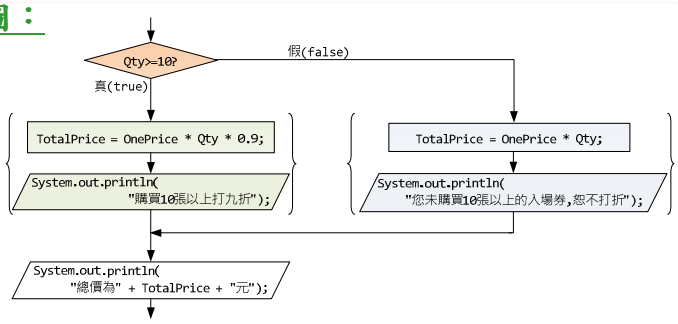
```
每張入場卷價格為200元
請輸入您要購買的張數:15
==========================
購買10張以上打九折
總價為2700.0元
```

```
每張入場卷價格為200元
請輸入您要購買的張數:5
==========================
您未購買10張以上的入場券,恕不打折
總價為1000.0元
```

27

4.2.2雙向選擇敘述（if-else敘述）

- 對應流程圖：



- 範例說明：
  - 在執行結果中，當使用者輸入『15』時，判斷式『Qty>=10』成立，因此第22~23行會被執行（第27~28行不會被執行）。
  - 當使用者輸入『5』時，判斷式『Qty>=10』不成立，因此第27~28行會被執行（第22~23行不會被執行）。
  - 而第30行是一定會被執行的，因為它並非if-else敘述的一部份。

28

4.2.3 e1？e2：e3特殊選擇運算式

- 「?:」運算符號可以用來替代簡單的if-else敘述

語法：變數=(條件運算式1) ？(運算式2)：(運算式3)；
功能：依照條件運算式1的成立與否，分別執行運算式2或運算式3，並將結果回傳給=左邊的變數。

- 【語法說明】：
  - (1)條件運算式1為true時，執行運算式2。條件運算式1為false時，執行運算式3。
  - (2)條件運算式1、運算式2、運算式3皆不含分號結尾。但整個敘述的最後必須含分號結尾。

- 【實用範例4-5】：設計一個猜數字遊戲，由使用者輸入的猜測數字，予與回覆是否為正確答案或數字太大、太小。

29

4.2.3 e1？e2：e3特殊選擇運算式

- 範例4-5：ch4_05.java（隨書光碟myJava\ch04\ch4_05.java）

```
1   /*  檔名:ch4_05.java      功能:「?:」運算子範例    */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_05          //主類別
8   {
9       public static void main(String args[])
10      {
11          Scanner keyboardInput = new Scanner(System.in);
12          String str1 = new String("恭喜您猜到了,獎品是一包乖乖.");
13          int Ans=38; /* 答案為38 */
14          int Guess;
15
16          System.out.print("請猜一個1~99的號碼:");
17          Guess=Integer.parseInt(keyboardInput.nextLine());
18          if(Guess!=Ans)
19              str1 = (Guess>Ans) ? "您猜得太大了" : "您猜得太小了" ;
20          System.out.println(str1);
21      }
22  }
```

30

4.2.3 e1？e2：e3特殊選擇運算式

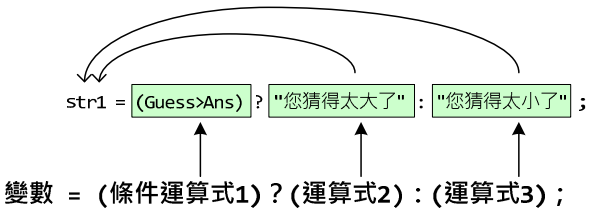
- 執行結果：

```
請猜一個1~99的號碼:38
恭喜您猜到了,獎品是一包乖乖.

請猜一個1~99的號碼:50
您猜得太大了

請猜一個1~99的號碼:12
您猜得太小了
```

31

4.2.3 e1？e2：e3特殊選擇運算式

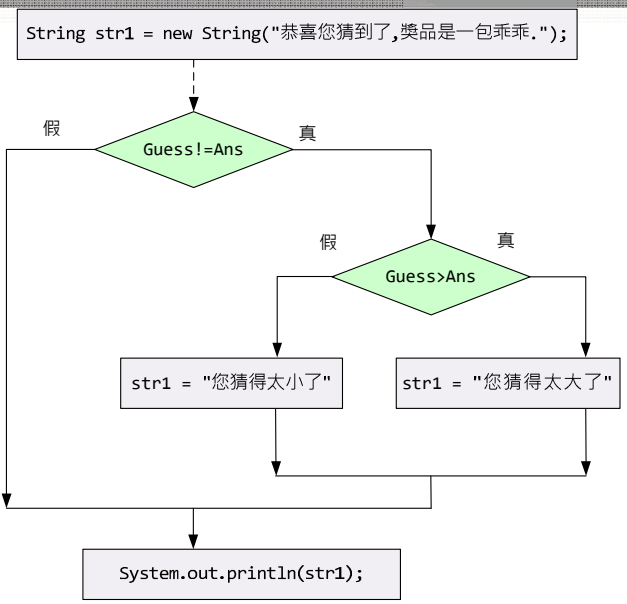
- 範例說明：
  - (1)在執行結果中，當使用者輸入『38』時，第19行不會被執行。在執行結果中，當使用者輸入的數字不是『38』時，第19行會被執行。
  - (2)第19行對應語法如下：



  - (3)第19行會依照(Guess>Ans)是否成立，決定回傳『"您猜得太大了"』（例如輸入50），還是回傳『"您猜得太小了"』（例如輸入12）給字串變數str1。所以整個流程如下。

32

4.2.3 e1？e2：e3特殊選擇運算式



33

4.2.4巢狀式選擇敘述

- 我們可以在選擇敘述的敘述區塊中再放入另一個選擇敘述，如此一來就形成了所謂的兩層式『巢狀式選擇』。
  - 還可製作更多層次及更多樣化的『巢狀式選擇』。
  - 巢狀式選擇的應用常見於『兩個以上的選擇條件』時
    - 當然我們也可以使用運算式來表達兩個以上的條件
    - 使用巢狀式選擇比設計複雜運算式來得容易理解。
  - 巢狀式選擇未規定外層的選擇敘述必須使用哪一種
    - 我們可以把內層的選擇式敘述放在單一選擇（if敘述）的敘述區塊，也可以放在雙向選擇（if-else）的敘述區塊A或else敘述區塊B中。

34

4.2.4巢狀式選擇敘述

- 圖4-2是一個巢狀式選擇的格式範例：

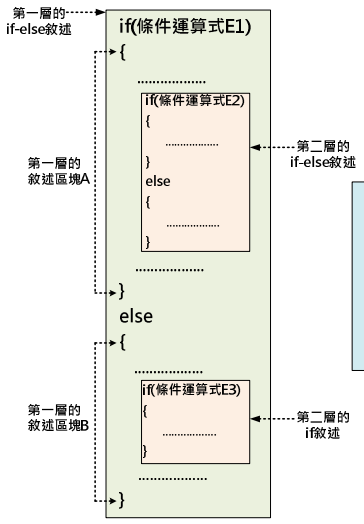


圖4-2巢狀式選擇格式範例

上述的巢狀式選擇敘述所對應的流程圖如下：

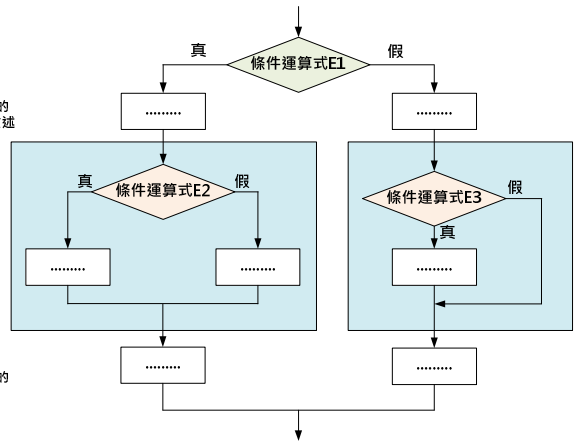


圖4-3 巢狀式選擇範例的對照流程圖

35

4.2.4巢狀式選擇敘述

- 【實用範例4-6】：根據輸入的繳款記錄、持卡年份評斷預借現金額度。其公式如下：
  - 繳款記錄：不正常 ==========>無法預借現金
  - 繳款記錄：正常
  - 持卡未滿半年 ==========>無法預借現金
  - 持卡滿半年未滿1年 ==========>預借現金額度為信用額度之1/20
  - 持卡滿1年 ==========>預借現金額度為信用額度之1/10

36

4.2.4巢狀式選擇敘述

- 範例4-6：ch4_06.java（隨書光碟 myJava\ch04\ch4_06.java）

```
1   /*  檔名:ch4_06.java      功能:巢狀選擇範例   */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_06          //主類別
8   {
9       public static void main(String args[])
10      {
11        Scanner keyboardInput = new Scanner(System.in);
12        int Credit,Status;  //信用額度,繳款狀態
13        double Year;        // 持卡年份
14
15        System.out.print("請輸入信用額度:");
16        Credit = Integer.parseInt(keyboardInput.nextLine());
17        System.out.print("繳款是否正常(1:正常,0:不正常):");
18        Status = Integer.parseInt(keyboardInput.nextLine());
```

37

4.2.4巢狀式選擇敘述

```
19        if(Status==1)
20        {
21          System.out.print("請輸入持卡年份:");
22          Year = Double.parseDouble(keyboardInput.nextLine());
23          if(Year>=0.5)
24          {
25            if(Year<1)
26            {
27              System.out.println("預借現金金額為"+(Credit*0.05)+"元");
28            }
29            else  //對應第25行的if
30            {
31              System.out.println("預借現金金額為"+(Credit*0.1)+"元");
32            }
33          }
34          else  //對應第23行的if
35          {
36            System.out.println("預借現金金額為0元");
37          }
38        }
39        else   //對應第19行的if
40        {
41          System.out.println("預借現金金額為0元");
42        }
43      }
44  }
```

38

4.2.4巢狀式選擇敘述

- 執行結果：

```
請輸入信用額度:80000
繳款是否正常(1:正常,0:不正常):0
預借現金金額為0元
```

```
請輸入信用額度:80000
繳款是否正常(1:正常,0:不正常):1
請輸入持卡年份:0.3
預借現金金額為0元
```

```
請輸入信用額度:80000
繳款是否正常(1:正常,0:不正常):1
請輸入持卡年份:0.8
預借現金金額為4000.0元
```

```
請輸入信用額度:80000
繳款是否正常(1:正常,0:不正常):1
請輸入持卡年份:2
預借現金金額為8000.0元
```

39

4.2.4巢狀式選擇敘述

- 範例說明：
  - (1)第19~42行：最外層if-else敘述，若條件(Status==1)成立，則執行第20~38行敘述區塊。若不成立則執行第40~42行。也就是符合題目之繳款不正常。
  - (2)第23~37行：第二層if-else敘述，若條件(Year>=0.5)成立，則執行第24~33行敘述區塊。若不成立則執行第35~37行。也就是符合題目之持卡未滿半年。
  - (3)第25~32行：第三層if-else敘述，若條件(Year<1)成立，則執行第26~28行敘述，也就是符合題目之持卡滿半年但未滿1年。若不成立則執行第30~32行，也就是符合題目之持卡滿1年。

40

4.2.4巢狀式選擇敘述

- 【實用範例4-7】：使用單一層的if或if-else敘述設計具有範例4-6效果的程式。
  - **範例4-7：**ch4_07.java（隨書光碟myJava\ch04\ch4_07.java）

```
1   /*  檔名:ch4_07.java        功能:條件運算式的練習    */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_07          //主類別
8   {
9       public static void main(String args[])
10      {
11        Scanner keyboardInput = new Scanner(System.in);
12        int Credit,Status;  //信用額度,繳款狀態
13        double Year=0;          // 持卡年份
14
15        System.out.print("請輸入信用額度:");
16        Credit = Integer.parseInt(keyboardInput.nextLine());
17        System.out.print("繳款是否正常(1:正常,0:不正常):");
18        Status = Integer.parseInt(keyboardInput.nextLine());
```

41

4.2.4巢狀式選擇敘述

```
19        if(Status!=1)
20        {
21           System.out.println("預借現金金額為0元");
22        }
23        else
24        {
25           System.out.print("請輸入持卡年份:");
26           Year = Double.parseDouble(keyboardInput.nextLine());
27        }
28        if((Status==1) && (Year>=0.5) && (Year<1))
29           System.out.println("預借現金金額為"+(Credit*0.05)+"元");
30        if((Status==1) && (Year>=1))
31           System.out.println("預借現金金額為"+(Credit*0.1)+"元");
32        if((Status==1) && (Year<0.5))
33           System.out.println("預借現金金額為0元");
34     }
35  }
```

- 執行結果：（同範例4-6）

42

4.2.4巢狀式選擇敘述

- 範例說明：
  - 由本程式可以看出，不使用巢狀式選擇敘述也可以解決相同的問題，不過您可能必須思考更多關於條件運算式涵蓋的範圍，才不至於出錯。

Coding 偷撇步

範例4-6、4-7解決了相同的問題，讀者可以選擇任何一種方法或者自行設計程式，但最好掌握下列原則：
1. 正確並符合題意的執行結果。（這是絕對需要遵守的條件）
2. 容易維護的設計方法（例如：易懂、適合除錯、具有擴充性）。
3. 縮短程式發展時程。
4. 考慮程式所需要的記憶體空間及執行效率。

43

4.2.5 else-if格式

- else-if並非Java語言的敘述，它只不過是將if-else敘述的某些空白重新排列而已
  - Java自由格式（也就是會去除多餘的空白）
  - 下列兩種格式對於編譯器而言，是沒有差別的。
  - 明顯地，若在下列左邊格式的else與if間換行，則事實上就是if-else敘述，而右邊的格式與左邊完全相同，只不過是將敘述區塊移到條件運算式之後，但卻更像是else-if格式：

44

4.2.5 else-if格式

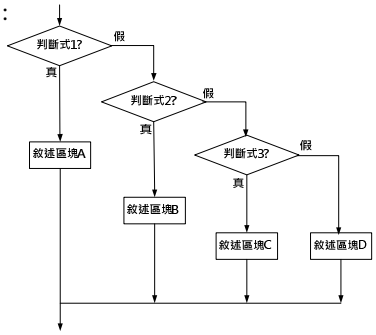
else-if格式一：

```
if(條件運算式1)
{
   敘述區塊A
}
else if(條件運算式2)
{
   敘述區塊B
}
else if(條件運算式3)
{
   敘述區塊C
}
else
{
   敘述區塊D
}
```

else-if格式二：

```
if(條件運算式1){ 敘述區塊A }
else if(條件運算式2){ 敘述區塊B }
else if(條件運算式3){ 敘述區塊C }
else { 敘述區塊D }
```

對應的流程圖：



- 【實用範例4-8】：使用else-if格式設計一個評定分數等級的程式。

45

4.2.5 else-if格式

- 範例4-8：ch4_08.java（隨書光碟 myJava\ch04\ch4_08.java）

```
1  /*  檔名:ch4_08.java    功能:else-if格式的練習   */
2
3  package myJava.ch04;
4  import java.lang.*;
5  import java.util.Scanner;
6
7  public class ch4_08          //主類別
8  {
9      public static void main(String args[])
10     {
11       Scanner keyboardInput = new Scanner(System.in);
12       int Score;
13
14       System.out.print("請輸入計概成績:");
15       Score = Integer.parseInt(keyboardInput.nextLine());
16       if(Score<60) { System.out.println("分數等級為丁等"); }
17       else if(Score<=69) { System.out.println("分數等級為丙等"); }
18       else if(Score<=79) { System.out.println("分數等級為乙等"); }
19       else if(Score<=89) { System.out.println("分數等級為甲等"); }
20       else if(Score<=99) { System.out.println("分數等級為優等"); }
21       else if(Score==100) { System.out.println("完美分數"); }
22       else { System.out.println("您輸入了不合法的分數"); }
23     }
24 }
```

執行結果

```
請輸入計概成績:99
分數等級為優等
```

46

4.2.5 else-if格式

- 範例說明：
  - else-if格式的每一個敘述區塊的執行條件彼此是互斥的（因為它是巢狀if-else的else部分）。
  - 除了使用else-if來製作上述範例之外，馬上我們也將使用switch-case敘述來製作同樣功能的程式。

47

4.2.6 浮點數比較的注意事項

- 雖然if-else的程式流程簡單易懂，但實際在設計程式時仍需注意條件運算式的使用，尤其是在進行浮點數的相等比較。
  - **【觀念範例4-9】**請設計一個出題程式，程式中預設一個台斤數（包含小數），請使用者輸入對應的公斤數，然後判別使用者是否輸入正確答案。

48

4.2.6 浮點數比較的注意事項

- **範例4-9：**ch4_09.java（隨書光碟myJava\ch04\ch4_09.java）

```
1  /*  檔名:ch4_09.java     功能:浮點數比較的陷阱    */
2
3  package myJava.ch04;
4  import java.lang.*;
5  import java.util.Scanner;
6
7  public class ch4_09          //主類別
8  {
9      public static void main(String args[])
10     {
11       Scanner keyboardInput = new Scanner(System.in);
12       float k,tk=40.5f;
13
14       System.out.println("一台斤=0.6公斤");
15       System.out.print("請問" + tk + "台斤等於幾公斤:");
16       k=Float.parseFloat(keyboardInput.nextLine());
17       if(k==tk*0.6)
18          System.out.println("答對了!");
19       else
20          System.out.println("答錯了!");
21     }
22 }
```

執行結果

```
一台斤=0.6公斤
請問40.50台斤等於幾公斤:24.3
答錯了!
```

輸入正確的數值，但卻會獲得錯誤的答案。

49

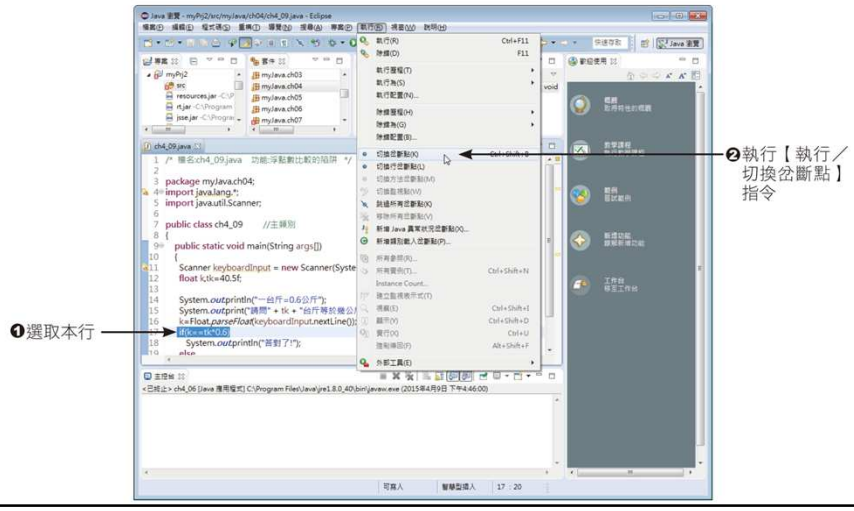
4.2.6 浮點數比較的注意事項

- 【範例說明】
  - 這個範例的執行結果將**出乎您的預料**，大部分初級的程式設計師都**找不出錯誤在哪裡**，除非透過IDE的除錯功能，採用單步執行，並且每一步都觀察變數k與tk的值才能找出錯誤所在。
  - 明顯地， 一般程式設計師會認為第16行會將k設定為24.3， 第17行的k==tk*0.6比較應該會成立，但執行結果卻告訴我們第17行是不成立的。
    - 所以如果我們能夠要求Eclipse執行到第17行時暫停，讓我們觀察各個變數的情況，就能夠完成除錯
  - 因此請如下操作，進行除錯：

50

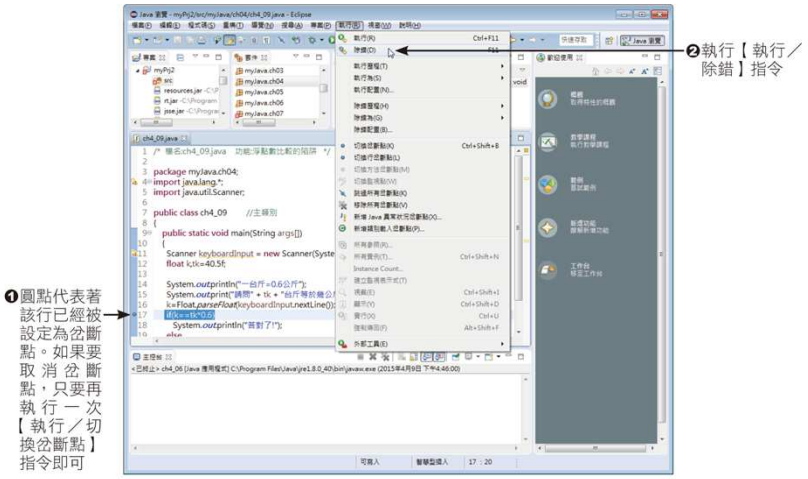
4.2.6 浮點數比較的注意事項

- STEP 1：參照附錄B.6，在Eclipse中開啟範例4-9，並且選取第17行，然後執行【執行／切換岔斷點】指令。



4.2.6 浮點數比較的注意事項

- Step2：此時會出現圓點在該行，這代表著該行已經被設定為岔斷點。執行【除錯／除錯】指令， IDE會執行到岔斷點後暫停



4.2.6 浮點數比較的注意事項

- Step3：執行過程會要求使用者輸入，請輸入24.3，接著會直行到第17行才暫停，並且會出現一個對話方塊，告知您將會開啟除錯視景，以方便除錯之用，故請按下【是】鈕。

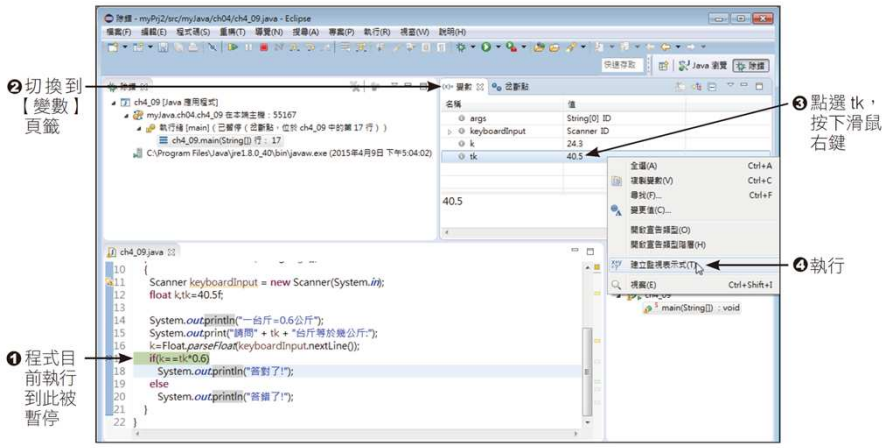
53

4.2.6 浮點數比較的注意事項

- Step4：回到IDE時，除錯視景已經被開啟，並且包含了「變數」與「岔斷點」兩個頁籤，其中變數頁籤會顯示程式中所有變數目前的變數值。
  - 我們可以發現k與tk的值並無異樣
  - 不過我們想要知道的是第17行的比較運算式的值，所以應該要新增一個監視表示式，並將之設定為tk*0.6f
    - 因為tk被宣告為float，所以乘法也以0.6f當作另一個運算元
  - 請如下操作：

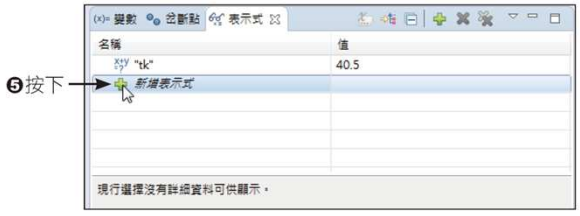
54

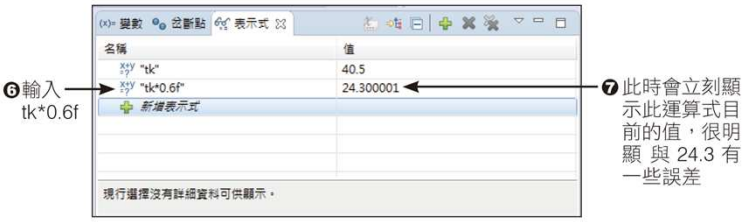
4.2.6 浮點數比較的注意事項



55

4.2.6 浮點數比較的注意事項





- 很明顯，我們已經發現問題在哪裡了，故請直接關閉 Eclipse。

56

4.2.6 浮點數比較的注意事項

- 問題到底出在哪裡呢？
  - 答案是精確度的影響，所以兩個浮點數進行相等的比較時，常常會出現問題（尤其是經過運算之後）。
  - 換句話說，我們應該要在進行數值比較時，使用整數變數，而不要使用浮點數。
    - 如果一定要使用浮點數進行比較，我們建議使用double做為資料型態比較好。

小試身手4-1
我們改寫範例4-9，使用double 來宣告k,tk，並進行相關的修改，使得執行結果是我們所預期的輸出" 答對了!"。我們將修改後的範例命名為範例4-10，程式碼請讀者參閱書附光碟。

57

4.2.6 浮點數比較的注意事項

- **範例4-10：**ch4_10.java（隨書光碟myJava\ch04\ch4_10.java）

```
1  /*  檔名:ch4_10.java     功能:浮點數比較    */
2
3  package myJava.ch04;
4  import java.lang.*;
5  import java.util.Scanner;
6
7  public class ch4_10          //主類別
8  {
9      public static void main(String args[])
10     {
11       Scanner keyboardInput = new Scanner(System.in);
12       double k,tk=40.5;
13
14       System.out.println("一台斤=0.6公斤");
15       System.out.print("請問" + tk + "台斤等於幾公斤:");
16       k = Double.parseDouble(keyboardInput.nextLine());
17       if(k==tk*0.6)
18          System.out.println("答對了!");
19       else
20          System.out.println("答錯了!");
21     }
22 }
```

執行結果

```
一台斤=0.6公斤
請問40.50台斤等於幾公斤:24.3
答對了!
```

58

4.2.7 if與else的配對

- 在一個程式中出現if關鍵字個數與else關鍵字個數不相對稱時，我們要如何得知哪一個else屬於哪一個if所有呢？
  - 其實這在Java中有明確的規定，**任何一個else都將與最接近的if配對**。
- 我們以下列範例來加以說明。
  - 【觀念範例4-11】：if與else的配對。
    - **範例4-11：**ch4_11.java（隨書光碟 myJava\ch04\ch4_11.java）

59

4.2.7 if與else的配對

```
1   /*  檔名:ch4_11.java      功能:if else的配對練習    */
2
3   package myJava.ch04;
4   import java.lang.*;
5
6   public class ch4_11          //主類別
7   {
8       public static void main(String args[])
9       {
10          int Score=75;
11          if(Score > 60)
12             if(Score > 80)
13                System.out.println("成績真不錯");
14          else
15             System.out.println("成績差強人意");
16      }
17  }
```

執行結果

```
成績差強人意
```

60

4.2.7 if與else的配對

- 範例說明：
  - (1)第14行的else看起來好像是和第11行的if配對，但實際上並非如此，它是與最接近的if（第12行的if）配對，因此執行結果為『成績差強人意』。
    - 如果我們將第12行的Score設定為50，則執行結果將不會印出任何字串。
  - (2)本範例說明了兩件事：
    - [1]Java是一種自由格式的程式語言，因此縮排與否並不會影響執行結果。
    - [2]Java的else將與最接近的if配對，因此本範例最好將之改寫如下縮排方式，以免產生錯覺。

61

4.2.7 if與else的配對

- 範例說明：

```
10        int Score=75;
11        if(Score > 60)
12           if (Score > 80)
13              System.out.println("成績真不錯");
14           else
15              System.out.println("成績差強人意");
```

- (3)如果讀者希望第15行的『System.out.println("成績差強人意");』改為在Score<=60的狀況下印出，則應該配合{}，改寫成下列格式。

```
10        int Score=75;
11        if(Score > 60)
12        {
13           if(Score > 80)
14              System.out.println("成績真不錯");
15        }
16        else
17           System.out.println("成績差強人意");
```

62

4.2.8 多向選擇敘述(switch-case敘述)

- 除了單一選擇與雙向選擇之外，Java還提供了switch-case多向選擇敘述。
  - 當狀況不只一個的時候，除了使用多個或巢狀式if-else選擇敘述之外，我們還可以使用switch-case多向選擇敘述，使得程式碼看起來更加清楚

63

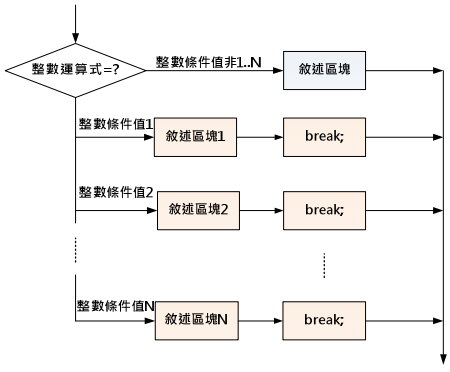
4.2.8 多向選擇敘述(switch-case敘述)

- 使用switch-case來做選擇決策，其語法包含兩類如下。

switch-case語法一：

```
switch(整數運算式)
{
  case整數條件值1：
    …敘述區塊1…
    break;
  case整數條件值2：
    …敘述區塊2…
    break;
    :
    :
  case 整數條件值N：
    …敘述區塊N…
    break;
  default：
    …敘述區塊…
    break;
}
```

switch-case流程圖一



64

4.2.8 多向選擇敘述(switch-case敘述)

- 【語法說明】：
  - (1)switch的
    - 每一個case跟隨著一個條件值，當中的敘述區塊代表當**整數運算式=整數條件值**時，所要執行的敘述區塊。
    - 而default之後不跟隨條件值，其中的敘述區塊則是代表當**整數運算式不等於任何一個整數條件值**時所要執行的敘述區塊。
    - 因此，default可有可無。
  - (2)case的敘述區塊可以是空敘述。
  - (3)整數條件值：條件值必須是資料常數（不可以是變數），例如整數或字元。
    - 並且每一個case的條件值必須不同。由於允許出現字元，因此，整數運算式也可以是字元運算式。
    - 【範例】：case 1:　　//測試值是否為1。
    - 【範例】：case 'X':　　//測試值是否為大寫字元『X』。

65

4.2.8 多向選擇敘述(switch-case敘述)

- 【語法說明】：
  - (4)break敘述：如果要符合上述的流程圖，則break敘述不可省略。
    - break的功能是作為跳離內部迴圈的分支敘述(branch statement)，例如：switch、for、while、do-while迴圈等。
  - 【註】：雖然break與後面要介紹的continue都具有跳躍功能，但它與無條件跳跳躍GOTO是有區別的，因為這兩個敘述的跳躍目標仍由程式語言所限制，而非如GOTO般隨意由程式設計師決定，因此，在Java中，容許使用這兩種敘述。

66

4.2.8 多向選擇敘述(switch-case敘述)

- (5)default的break敘述：
  - 事實上，switch-case敘述的case與default順序並不重要，隨意更動case的順序，不會影響其正確性
  - 但當您將default放在最後面時，default內的break敘述可以省略。
- (6)falling through：
  - 許多程式設計師常常會忘了在case的敘述區塊之後加入break敘述，此時將出現falling through現象
  - 也就是當程式執行完符合條件的case之後，仍舊會往下執行其他case的區塊敘述（不論是否符合case之條件），而不立即跳離switch-case敘述，請見範例4-14。

67

4.2.8 多向選擇敘述(switch-case敘述)

- 【語法說明】：
  - 語法二允許條件為字串，這是JDK7才新增的功能
  - 字串運算式是包含所有可回傳一個字串的運算式
    - 例如一個字串變數、兩個字串變數相連結、函式呼叫的返回值為String型態等等。
  - 而字串常數值則應該以雙引號包裝起來，例如"abc"。

switch-case語法二：

```
switch(字串運算式)
{
  case字串常數值1：
    …敘述區塊1…
    break;
  case字串常數值2：
    …敘述區塊2…
    break;
    :
    :
  case 字串常數值n：
    …敘述區塊n…
    break;
  default：
    …敘述區塊…
    break;
}
```

68

4.2.8 多向選擇敘述(switch-case敘述)

- 【實用範例4-12】：使用switch-case敘述設計一個評定分數等級的程式。
  - **範例4-12：**ch4_12.java（隨書光碟 myJava\ch04\ch4_12.java）

```
1   /*  檔名:ch4_12.java       功能:switch-case的練習  */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_12          //主類別
8   {
9       public static void main(String args[])
10      {
11        Scanner keyboardInput = new Scanner(System.in);
12        int Score;
13
14        System.out.print("請輸入計概成績:");
15        Score=Integer.parseInt(keyboardInput.nextLine());
```

69

4.2.8 多向選擇敘述(switch-case敘述)

```
16        if((Score>=0) && (Score<=100))
17          switch(Score / 10)
18          {
19            case 10:
20                System.out.println("完美分數");
21                break;
22            case 9:
23                System.out.println("分數等級為優等");
24                break;
25            case 8:
26                System.out.println("分數等級為甲等");
27                break;
28            case 7:
29                System.out.println("分數等級為乙等");
30                break;
31            case 6:
32                System.out.println("分數等級為丙等");
33                break;
34            default:
35                System.out.println("分數等級為丁等");
36                break;
37          }
38      }
39  }
```

70

4.2.8 多向選擇敘述(switch-case敘述)

- 執行結果：

```
請輸入計概成績:99
分數等級為優等
```

```
請輸入計概成績:75
分數等級為乙等
```

- 範例說明：
  - (1)本範例與範例4-8具有相同功能。第16行的if敘述是用來確保輸入的數字在規定的範圍0~100之內，當符合規定範圍後，將執行第17~37行的switch-case敘述。
  - (2)第17行switch的條件運算式為（Score/10），由於Score、10皆為整數，因此(Score/10)的結果也將是整數（小數部分將被捨棄）。
  - (3)根據（Score/10）的結果，決定要執行哪一個case敘述區塊，若結果非10、9、8、7、6，則執行default敘述區塊
  - (4)我們在每一個case敘述區塊末端加入了break敘述，所以眾多的case敘述區塊只會被執行其中之一（不會發生falling through現象）。至於本範例default敘述區塊的末端break敘述（第36行）則可以省略，因為它已經是位於switch-case的最後敘述。

71

4.2.8 多向選擇敘述(switch-case敘述)

- 【觀念範例4-13】：設計一個翻譯分數等級的程式，並測試switch-case敘述的字串功能。
  - **範例4-13：**ch4_13.java（隨書光碟 myJava\ch04\ch4_13.java）

```
1   /*  檔名:ch4_13.java       功能:switch-case的字串條件練習  */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_13         //主類別
8   {
9       public static void main(String args[])
10      {
11        Scanner keyboardInput = new Scanner(System.in);
12
13        System.out.print("請輸入計概等第(中文):");
```

72

4.2.8 多向選擇敘述(switch-case敘述)

```
14          switch(keyboardInput.nextLine())
15             {
16               case "優等":
17                   System.out.println("英文等第為A");
18                   break;
19               case "甲等":
20                   System.out.println("英文等第為B");
21                   break;
22               case "乙等":
23                   System.out.println("英文等第為C");
24                   break;
25               case "丙等":
26                   System.out.println("英文等第為D");
27                   break;
28               case "丁等":
29                   System.out.println("英文等第為F");
30                   break;
31               default:
32                   System.out.println("您輸入了錯誤的等第");
33                   break;
34             }
35       }
36   }
```

73

4.2.8 多向選擇敘述(switch-case敘述)

- 執行結果：

```
請輸入計概等第(中文):優等
英文等第為A
```

```
請輸入計概等第(中文):王等
您輸入了錯誤的等第
```

- **範例說明：**
  - 您可以在第13行加入
    String str = keyboardInput.nextLine();
    ，如此第14行就可以改為switch(str)。

74

4.2.8 多向選擇敘述(switch-case敘述)

- 【觀念範例4-14】：falling through現象。
  - **範例4-14：**ch4_14.java（隨書光碟 myJava\ch04\ch4_14.java）

```
1   /*  檔名:ch4_14.java      功能:falling through的示範  */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_14         //主類別
8   {
9       public static void main(String args[])
10      {
11        Scanner keyboardInput = new Scanner(System.in);
12        int Score;
13
14        System.out.print("請輸入計概成績:");
15        Score=Integer.parseInt(keyboardInput.nextLine());
16        if((Score>=0) && (Score<=100))
```

75

4.2.8 多向選擇敘述(switch-case敘述)

```
17          switch(Score / 10)
18          {
19            case 10:
20                System.out.println("完美分數");
21            case 9:
22                System.out.println("分數等級為優等");
23            case 8:
24                System.out.println("分數等級為甲等");
25            case 7:
26                System.out.println("分數等級為乙等");
27            case 6:
28                System.out.println("分數等級為丙等");
29            default:
30                System.out.println("分數等級為丁等");
31          }
32      }
33  }
```

- 執行結果：

```
請輸入計概成績:99
分數等級為優等
分數等級為甲等
分數等級為乙等
分數等級為丙等
分數等級為丁等
```

```
請輸入計概成績:75
分數等級為乙等
分數等級為丙等
分數等級為丁等
```

76

4.2.8 多向選擇敘述(switch-case敘述)

- 範例說明：
  - 本範例將範例4-12的break敘述去除，因此產生了falling through現象。從執行結果中，讀者可以發現，當某一個case被滿足後，除了該敘述區塊會被執行之外，其後所有的case與default敘述區塊也會被執行（不論是否符合該case的條件值）。

77

4.3 『迴圈』敘述

- 高階語言的迴圈結構
  - 事實上是結合了低階語言的決策與跳躍，使得程式中可以有某部分敘述區塊能夠被重複執行多次
    - （每一次重複執行，稱之為一個iteration）。
  - Java的迴圈結構又分為『計數迴圈』與『條件式迴圈』兩種
    - 而條件式迴圈又分為『前測式』與『後測式』兩種條件式迴圈。

78

4.3.1 計數迴圈（for迴圈敘述）

- **迴圈**可以重複不停的做某些動作直到某個條件成立時，動作才會停止。
  - Java提供了多種的迴圈，我們首先介紹for計數迴圈。

```
int Sum=0;
for(int count=1;count<=10;count++)
   Sum=Sum+count;
```

  - 在上面的小範例中， 迴圈一共會被執行10次，因此『Sum= Sum+count』也總共會被執行10次。
    - 最開始count 值為1、Sum值為0
    - 每次重複執行迴圈時，變數count值都會加1
    - 所以當迴圈執行完畢，Sum值就會是1~10的總和55。

79

4.3.1 計數迴圈（for迴圈敘述）

- Java可以在任意處宣告變數
  - 我們可以在迴圈內宣告count，但迴圈一旦執行完畢，count的生命週期就結束，所以在迴圈外將無法存取count。這類型的變數也被稱為迴圈變數。
  - 在上述程式中，count於迴圈結束後就不需使用，因此，可以在迴圈內宣告
  - 而Sum在迴圈外仍需使用，因此應該在迴圈之前宣告。

80

4.3.1 計數迴圈（for迴圈敘述）

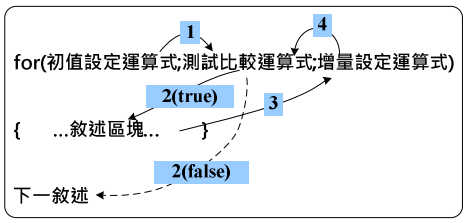
- for迴圈的語法說明
  - 我們將for迴圈語法整理如下，再來統一說明for迴圈的執行流程。
  - 【for迴圈語法】

```
for(初值設定運算式;測試比較運算式;增量設定運算式)
{                  …… 敘述區塊 ……                  }
```

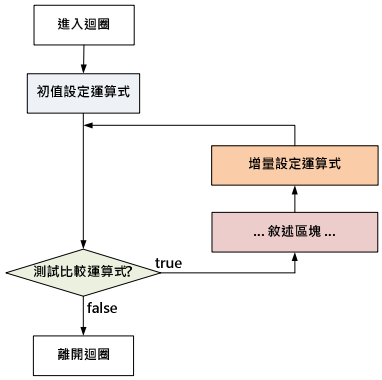
81

4.3.1 計數迴圈（for迴圈敘述）

for迴圈執行流程：



for迴圈流程圖：



- 【功能】：
  - 電腦會依照『測試比較運算式』的成立與否，決定是否重覆執行迴圈內的敘述區塊』。每次執行完畢會執行『增量設定運算式』。

82

4.3.1 計數迴圈（for迴圈敘述）

- 【語法說明】：
  - 1. 初進入迴圈時，會先執行『初值設定運算式』，然後依照『測試比較運算式』是否成立（結果是否為true），決定是否執行『迴圈內的敘述區塊』。
    - 敘述區塊執行完畢會執行『增量設定運算式』，然後再依照『測試比較運算式』是否成立決定是否繼續執行『迴圈內的敘述區塊』
    - 敘述區塊執行完畢又會再次執行『增量設定運算式』，依此順序重複執行敘述區塊，直到離開迴圈為止（『測試比較運算式』不成立時）。
  - 2. 『初值設定運算式』是進入迴圈時首先被執行的運算式，它只會被執行一次
    - 當有兩個以上的運算式時，則以『,』加以區隔。

83

4.3.1 計數迴圈（for迴圈敘述）

- 3. 『測試比較運算式』是決定是否重複進入迴圈的依據，以該運算式的結果布林值作為重複進入迴圈的條件。
- 4. 『增量設定運算式』是每次執行完迴圈敘述區塊後將被執行的運算式
  - 當有兩個以上的運算式時，則以『,』加以區隔。
- 5. 『初值設定運算式』、『測試比較運算式』、『增量設定運算式』之間並不見得存在任何關係，但我們通常會將之設定為有某種關係，以控制迴圈的重複次數。
- 6. 迴圈內的敘述區塊若僅包含單一敘述，則可以省略{}。
- 7. 迴圈內的敘述區塊也可以包含其他的迴圈，此時就會形成多重迴圈，請見範例4-17。

84

4.3.1 計數迴圈（for迴圈敘述）

- 8. 以下列範例而言，當離開迴圈之後，count變數值應該是11，而不是10（請對照for迴圈流程圖）

```
int Sum,count;
for(Sum=0,count=1;count<=10;count++)
  Sum=Sum+count;
```

- 9. 以下列範例而言，迴圈永遠不會停止（稱之為無窮迴圈），因為test=true是將test設定為true，因此每次執行test=true運算式時，都會得到一個布林值true。

```
boolean test=false;
int Sum,count;
for(Sum=0,count=1;test=true;count++)
    Sum=Sum+count;
```

85

4.3.1 計數迴圈（for迴圈敘述）

- 10. 以下列範例而言，迴圈永遠不會停止（稱之為無窮迴圈），因為Sum>=0永遠都會成立。
  - 但事實是當Sum超出int所能表達之正整數範圍後，將產生溢位，此時會被判定為負值或產生例外，因此會停止。
  - 不過結果通常不是程式設計師所要的結果，會出現下列程式片段，通常可能是設計程式時，某個地方出現邏輯上的錯誤。

```
int Sum=0,i,j;
for(i=1,j=1;Sum>=0;i++,j++)
    Sum=i+j;
```

86

4.3.1 計數迴圈（for迴圈敘述）

- 11. 在前面介紹過的break敘述，可以用來強制中途跳出迴圈。如下圖示意：

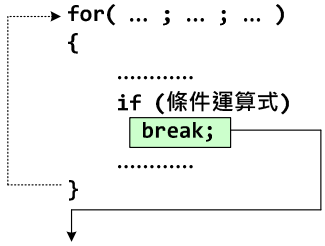


圖4-4 break敘述與迴圈示意圖

87

4.3.1 計數迴圈（for迴圈敘述）

- 12. 另外還有一個continue敘述，則是用來強制中途略過本次迴圈剩餘尚未執行的步驟，讓程式由下一次的迴圈開始。如下圖示意：

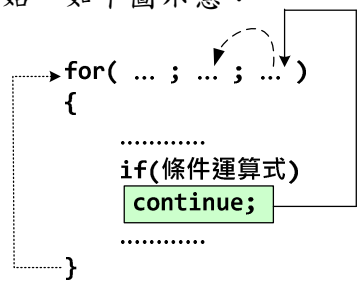


圖4-5 continue敘述與迴圈示意圖

88

4.3.1 計數迴圈（for迴圈敘述）

- 【實用範例4-15】：使用for迴圈，計算1+3+‥‥+N（N為奇數時）或1+3+‥‥+N-1（N為偶數時）的總和。
  - **範例4-15：**ch4_15.java（隨書光碟myJava\ch04\ch4_15.java）

```
1   /*  檔名:ch4_15.java       功能:for迴圈的示範  */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_15          //主類別
8   {
9       public static void main(String args[])
10      {
11          Scanner keyboardInput = new Scanner(System.in);
12          int Sum=0,n;
13
14          System.out.print("請輸入N值:");
15          n=Integer.parseInt(keyboardInput.nextLine());
16
```

89

4.3.1 計數迴圈（for迴圈敘述）

```
17          for(int i=1;i<=n;i=i+2)
18              Sum = Sum + i;
19          if((n%2)==1)
20              System.out.println("1+3+...+N=" + Sum);
21          else
22              System.out.println("1+3+...+N-1=" + Sum);
23      }
24  }
```

- 執行結果：

```
請輸入N值:7
1+3+...+N=16
```

```
請輸入N值:8
1+3+...+N-1=16
```

90

4.3.1 計數迴圈（for迴圈敘述）

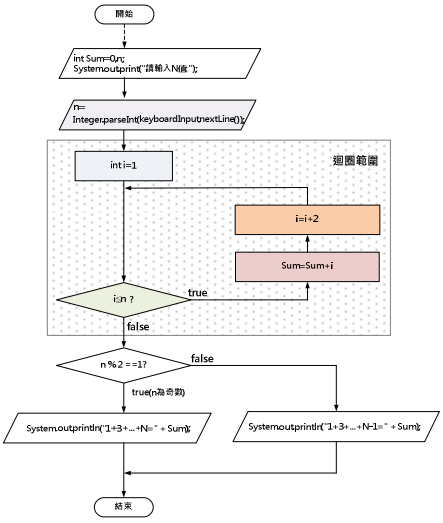
- 範例說明：
  - (1) 第17~18行是for迴圈的範圍，由於for迴圈內僅有單一敘述（Sum=Sum+i），所以不用加上{}。讀者可以將之改寫如下格式。

```
for(int i=1;i<=n;i=i+2)
{
   Sum=Sum+i;
}
```

  - (2) for迴圈的『初值設定運算式』是int i=1。『測試比較運算式』是i<=n。『增量設定運算式』是i=i+2。
  - (3)迴圈執行完畢的i值一定是奇數，如果使用者輸入的是奇數N，則i值為N+2，如果使用者輸入的是偶數N，則i值為N+1。（因為只有當i>n時，『測試比較運算式』i<=n才會不成立），不過由於我們將變數i宣告在迴圈敘述中，因此在迴圈外部無法得到i值（想在迴圈外存取i值，請將宣告移往迴圈之前）。

4.3.1 計數迴圈（for迴圈敘述）

- 範例說明：
  - (4)本範例流程圖如右：



- 【實用範例4-16】：使用for迴圈改寫範例4-15，但每次重複迴圈時，i只能增加1。
  - **範例4-16：**ch4_16.java（隨書光碟myJava\ch04\ch4_16.java）

4.3.1 計數迴圈（for迴圈敘述）

```
1  /*  檔名:ch4_16.java      功能:for迴圈的示範  */
2
3  package myJava.ch04;
4  import java.lang.*;
5  import java.util.Scanner;
6
7  public class ch4_16          //主類別
8  {
9      public static void main(String args[])
10     {
11         Scanner keyboardInput = new Scanner(System.in);
12         int Sum=0,n;
13
14         System.out.print("請輸入N值:");
15         n=Integer.parseInt(keyboardInput.nextLi
16
17         for(int i=1;((n%2)==1)?(i<=n):(i<=n-1);i++)
18             if((i%2)==1)
19                 Sum = Sum + i;
20         if((n%2)==1)
21             System.out.println("1+3+...+N=" + Sum);
22         else
23             System.out.println("1+3+...+N-1=" + Sum);
24     }
25 }
```

n為偶數時，由(i<=n-1)進行條件判斷

n為奇數時，由(i<=n)進行條件判斷

4.3.1 計數迴圈（for迴圈敘述）

- **執行結果：**（同範例4-15）
- 範例說明：
  - (1)第17~19行是for迴圈的範圍，由於for迴圈內僅有單一敘述（if敘述），所以不用加上{}。同理if內也只有單一敘述，因此也不用加上{}。
    - 讀者可以將之改寫如下格式。

```
for(int i=1;((n%2)==1)?(i<=n):(i<=n-1);i++)
{
  if((i%2)==1)
  {
    Sum=Sum+i;
  }
}
```

4.3.1 計數迴圈（for迴圈敘述）

- 範例說明：
  - (2)for迴圈的
    - 『初值設定運算式』是int i=1。
    - 『測試比較運算式』是((n%2)==1)?(i<=n):(i<=n-1)條件敘述
    - 『增量設定運算式』是i++敘述（代表遞增1）。
  - (3)測試比較運算式((n%2)==1)?(i<=n):(i<=n-1)
    - 使用的『%』是餘數運算子，當配合運算元『2』時，可以用來判斷奇數或偶數。
    - 再配合『?:』條件運算子，所以當n為奇數時，將執行(i<=n)條件運算式，當n為偶數時，則執行(i<=n-1)條件運算式。
    - 原本『?:』運算子應該設定一個變數來接收(i<=n)或(i<=n-1)的布林結果，而布林結果可以被for迴圈用來檢測迴圈是否繼續，故已被使用而可以不用接收，如果您想要接收的話也可以，只要將之改寫為boolean test=((n%2)==1)?(i<=n):(i<=n-1)即可。
  - (4)同理，不論是第18行或第20行也是使用餘數運算子『%』來測試i或n是否為奇數。

95

4.3.1 計數迴圈（for迴圈敘述）

- 【實用範例4-17】： 使用2層for迴圈設計九九乘法表。
  - 範例4-17：ch4_17.java（隨書光碟 myJava\ch04\ch4_17.java）

```
1  /*  檔名:ch4_17.java       功能:多層for迴圈的示範    */
2
3  package myJava.ch04;
4  import java.lang.*;
5
6  public class ch4_17          //主類別
7  {
8    public static void main(String args[])
9    {
10     for(int i=1;i<=9;i++)
11     {
12       for(int j=1;j<=9;j++)
13           System.out.print(i + "*" + j + "=" + i*j + "\t");
14       System.out.println();
15     }
16   }
17 }
```

內層迴圈　外層迴圈

96

4.3.1 計數迴圈（for迴圈敘述）

- 執行結果：

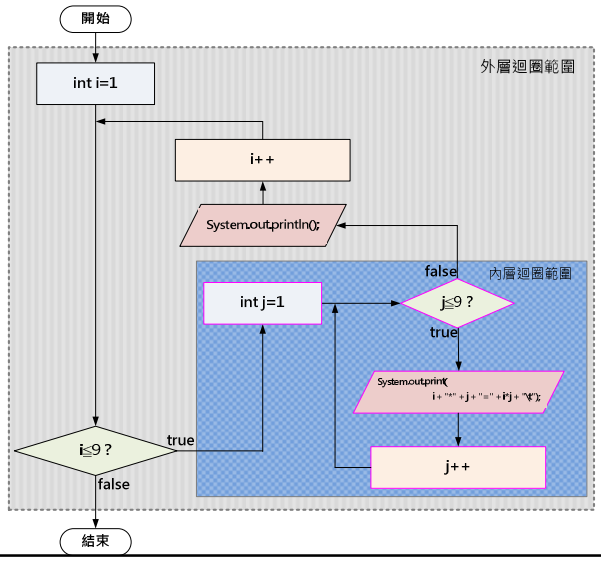
```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5   5*2=10  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
```

- 範例說明：
  - (1)第10~15行是外層迴圈，第12~13行是內層迴圈，第13行一共會被執行9*9=81次，而第14行則只會被執行9次。
  - (2)雖然i是在外層迴圈宣告，但在內層迴圈仍可存取i，因為內層迴圈屬於外層迴圈內的敘述之一；反之，由於j是在內層迴圈宣告，因此若在外層迴圈的第14行存取j，則編譯時會發生錯誤。

97

4.3.1 計數迴圈（for迴圈敘述）

- (3)本範例流程圖如下：



98

4.3.1 計數迴圈（for迴圈敘述）

- 【觀念範例4-18】：使用continue敘述，計算2+4+6+8+10的總和。
  - 範例4-18：ch4_18.java（隨書光碟 myJava\ch04\ch4_18.java）

```
1  /*  檔名:ch4_18.java      功能:continue敘述的示範   */
2
3  package myJava.ch04;
4  import java.lang.*;
5
6  public class ch4_18          //主類別
7  {
8      public static void main(String args[])
9      {
10        int Sum=0;
11        for(int i=1;i<=10;i++)
12        {
13          if((i%2)==1)
14             continue;
15          Sum=Sum+i;
16        }
17        System.out.println("2+4+6+8+10=" + Sum);
18     }
19 }
```

執行結果

```
2+4+6+8+10=30
```

強迫立即回到迴圈開頭

99

4.3.1 計數迴圈（for迴圈敘述）

- 範例說明：
  - 當i為奇數時，第13行條件成立，因此執行第14行的continue敘述，此時將略過迴圈尚未執行的第15行敘述，直接進入下一次的迴圈（從執行增量運算式i++開始）。所以Sum只會累計偶數之和。

- 【觀念範例4-19】：使用break敘述強迫中途跳出迴圈。
  - **範例4-19：**ch4_19.java（隨書光碟 myJava\ch04\ch4_19.java）

100

4.3.1 計數迴圈（for迴圈敘述）

```
1  /*  檔名:ch4_19.java      功能:break敘述的示範  */
2
3  package myJava.ch04;
4  import java.lang.*;
5  import java.util.Scanner;
6
7  public class ch4_19
8  {
9      public static void main(String args[])
10     {
11       Scanner keyboardInput = new Scanner(System.in);
12       int Sum=0,i,n;
13
14       System.out.print("求1~N的總和,請輸入N值:");
15       n=Integer.parseInt(keyboardInput.nextLine());
16       for(i=1;i<=n;i++)
17       {
18         if(Sum>Integer.MAX_VALUE-100)
19           break;        強迫立即跳離迴圈
20         Sum=Sum+i;
21       }
22       System.out.println("1~" + (i-1) + "的總和為" + Sum);
23     }
24 }
```

執行結果

```
求1~N的總和,請輸入N值:100000000
1~5004393的總和為2147483581
```

101

4.3.1 計數迴圈（for迴圈敘述）

- 範例說明：
  - 第18行條件成立時，將執行第19行的break敘述，強迫跳離迴圈（此時第20行不會被執行）
    - 所以可以避免Sum超過int的最大值2,147,483,647限制所造成的錯誤。
  - 而第18行的Integer.MAX_VALUE則是上一章所言之int範圍的最大值2,147,483,647
    - 它記錄於java.lang.Integer類別中。

102

4.3.2 前測式條件迴圈(while迴圈敘述)

- 在for迴圈中
  - 我們可以設定迴圈變數的初值、增量設定運算式與決定結束迴圈的條件比較運算式
  - 若不使用break敘述，則通常在迴圈一開始被執行的時候，我們就可以很容易地『預測』迴圈內部敘述區塊的執行次數。
    - 因此for迴圈一般用在與數字相關的運算，例如有初值及增量運算時使用。

103

4.3.2 前測式條件迴圈(while迴圈敘述)

- 但是並非所有的狀況都必須用for迴圈來撰寫。
  - 例如我們希望程式一直重複做某件事，直到某個條件成立為止，而非執行固定的次數（無固定之增量值）。
  - 這個時候，我們就可以採用條件式迴圈。
- Java提供的條件式迴圈有兩種：
  - 前測式迴圈（while迴圈）
  - 後測式迴圈（do-while迴圈）。

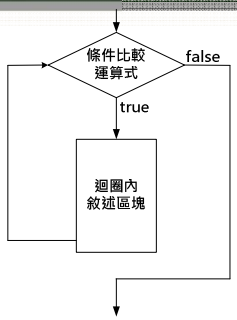
104

4.3.2 前測式條件迴圈(while迴圈敘述)

- while迴圈語法：

```
while(條件比較運算式)
{
   ……敘述區塊……
}
```

while迴圈流程圖：



- 【功能】：
  - 執行迴圈前先檢查條件比較運算式是否為true，若是則進入迴圈，否則離開迴圈。
- 【語法說明】：
  - 1. 若『條件比較運算式』結果為true，則進入迴圈內執行敘述區塊；否則不進入迴圈，直接跳往迴圈之後的下一個敘述繼續執行。

105

4.3.2 前測式條件迴圈(while迴圈敘述)

- 【語法說明】：
  - 2. 迴圈內敘述區塊執行完畢，將重新測試條件比較運算式，若條件比較運算式結果仍為true則再度執行迴圈內敘述區塊，若為false則跳離迴圈。如此週而復始，直到條件式結果為false時，才跳離迴圈。
  - 3. 若迴圈內的敘述區塊僅包含單一敘述，則可以省略{ }
  - 4. 迴圈內的敘述必須能夠改變條件比較運算式的成立狀態，或使用break敘述離開迴圈，否則將形成無窮迴圈。
    - 如果條件比較運算式永遠為true，則while迴圈就會成為無窮迴圈。常見的無窮迴圈敘述如下。

```
while(true)
{
   ……敘述區塊……
}
```

```
i=10;
while(i>7)
{
   ……無修改i的敘述……
}
```

106

4.3.2 前測式條件迴圈(while迴圈敘述)

- 【語法說明】：
  - 5.break敘述可用來強制立刻中途跳離迴圈。在無窮迴圈時常常必須配合break敘述來強制跳離迴圈。
  - 6.continue敘述是用來立刻略過該次迴圈剩餘未執行的部分，回到迴圈頂端（重新測試條件比較運算式）。
  - 7.while迴圈內可以包含其他的迴圈，成為多重迴圈。
  - 8.Java的while迴圈程式也可以改寫為for迴圈，（但如此一來就喪失了for迴圈的特點），兩種格式如下對照：

```
while(條件比較運算式)
{
   ……敘述區塊……
}
```

=

```
for(;條件比較運算式;)
{
   ……敘述區塊……
}
```

107

4.3.2 前測式條件迴圈(while迴圈敘述)

- 【實用範例4-20】：使用前測式while迴圈，撰寫一個根據輾轉相除法（歐幾里得法）求兩數最大公因數的程式。輾轉相除法範例如右：
  - **範例4-20：**ch4_20.java（隨書光碟myJava\ch04\ch4_20.java）

	792	102	
7	714	78	1
	78	24	
3	72	24	4
	6	0	

最大公因數

```
1   /*  檔名:ch4_20.java       功能:while迴圈的示範  */
2
3   package myJava.ch04;
4   import java.lang.*;
5   import java.util.Scanner;
6
7   public class ch4_20          //主類別
8   {
9       public static void main(String args[])
10      {
11        Scanner keyboardInput = new Scanner(System.in);
12        int x,y,gcd,temp;
13
```

108

4.3.2 前測式條件迴圈(while迴圈敘述)

```
14          System.out.print("輸入x:");
15          x=Integer.parseInt(keyboardInput.nextLine());
16          System.out.print("輸入y:");
17          y=Integer.parseInt(keyboardInput.nextLine());
18          System.out.print("(" + x + "," + y + ")=");
19          if(x<y)           // 將較大的數值放在x,較小的放在y
20          {
21             temp = x;   x = y;    y = temp;     // x,y數值對調
22          }
23          while(x!=0)
24          {
25             x=x%y;
26             if(x!=0)
27             {
28                temp = x;    x = y;    y = temp;     // x,y數值對調
29             }
30          }
31          gcd=y;
32          System.out.println(gcd);
33       }
34    }
```

第21行共有3個敘述『temp = x;』、『x = y;』、『y = temp;』，目的是為了將x與y的值對調。

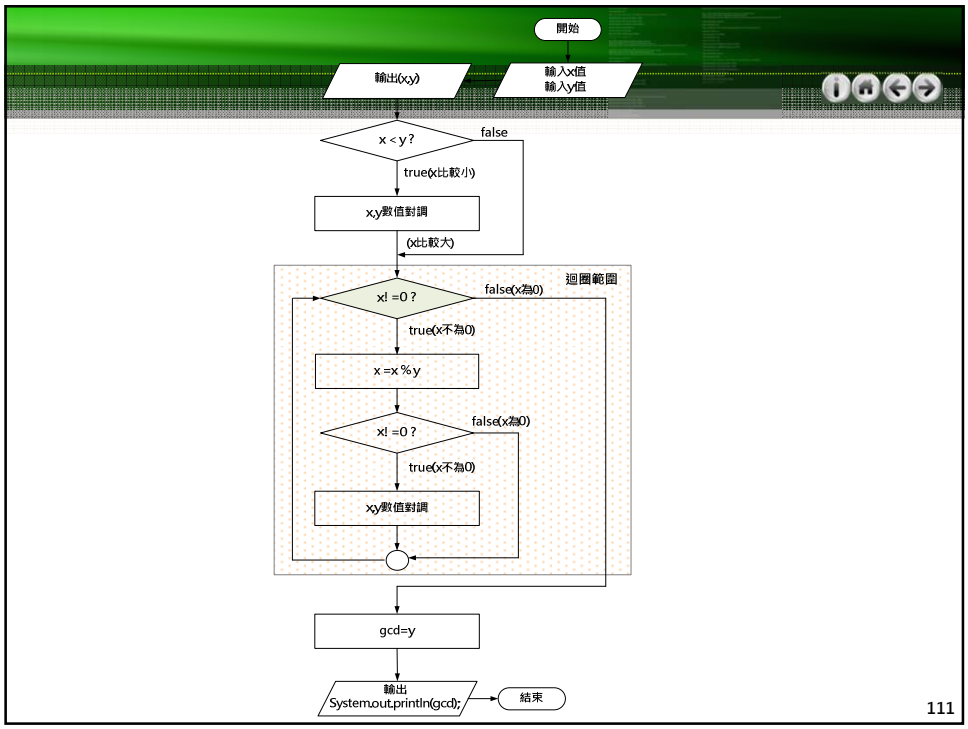
4.3.2 前測式條件迴圈(while迴圈敘述)

- 執行結果：

```
輸入x:792
輸入y:102
(792,102)=6
```

```
輸入x:117
輸入y:663
(117,663)=39
```

- 範例說明：
  - (1)第23~30行：此處出現了一個while迴圈，迴圈執行的次數完全依照x變數值的變化來決定，除非x變數值為0，否則迴圈將一直重複執行。
  - (2)本範例所使用的歐幾里得輾轉相除法求最大公因數，非常適合用while迴圈來撰寫
    - 如果強迫使用for迴圈撰寫的話，將會發現困難了許多，讀者可以自行試試看。
  - (3)本範例流程圖如下



111

4.3.3 後測式條件迴圈（do-while迴圈敘述）

- 另一種條件迴圈稱之為**後測式條件迴圈**，也就是**do-while迴圈**。
  - 此類迴圈最大的特色是，在不使用break與continue敘述的狀況下，**迴圈內的敘述區塊至少會被執行一次**。

112

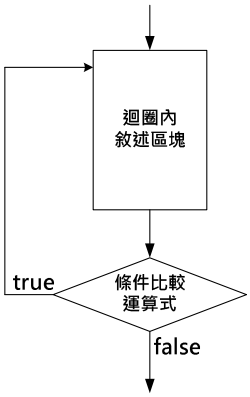
4.3.3 後測式條件迴圈（do-while迴圈敘述）

- do-while迴圈語法：

```
do
{
   ……敘述區塊……
}while(條件比較運算式);
```

- 【功能】：
  - 先進入迴圈，執行敘述區塊一次後，再判斷是否要繼續重覆執行迴圈。

do-while迴圈流程圖：



113

4.3.3 後測式條件迴圈（do-while迴圈敘述）

- 【語法說明】：
  - 1. while之後的小括號後面規定必須出現『;』。
  - 2. 先執行敘述區塊一次，然後再判斷『條件比較運算式』，若『條件比較運算式』結果為true，則再重覆進入迴圈內執行敘述區塊；否則將離開迴圈，前往迴圈之後的下一個敘述繼續執行。
  - 3. 迴圈內敘述區塊執行完畢，將重新測試『條件比較運算式』，若『條件比較運算式』結果仍為true則再執行迴圈內敘述區塊，若不成立則跳離迴圈。如此週而復始，直到『條件比較運算式』結果為false時，才跳離迴圈。

114

4.3.3 後測式條件迴圈（do-while迴圈敘述）

- 【語法說明】：
  - 4. 若迴圈內的敘述區塊僅包含單一敘述，則可以省略{ }。
  - 5. 迴圈內的敘述區塊至少會被執行一次以上。
  - 6. 迴圈內的敘述必須能夠改變『條件比較運算式』的成立狀態，或使用break敘述離開迴圈，否則將形成無窮迴圈。
  - 7. break敘述可用來強制立刻中途跳離迴圈。
  - 8. continue敘述是用來立刻略過該次迴圈剩餘未執行的部分，直接跳到測試『條件比較運算式』的步驟。
  - 9. do-while迴圈內可包含其他的迴圈，成為多重迴圈。

115

4.3.3 後測式條件迴圈（do-while迴圈敘述）

- 【實用與觀念範例4-21】：使用後測式do-while迴圈，撰寫一個遊戲結束時的『Play Again?』詢問選項。並且透過本範例，練習將字串轉為字元的方法。
  - **範例4-21：**ch4_21.java（隨書光碟myJava\ch04\ch4_21.java）

116

4.3.3 後測式條件迴圈（do-while迴圈敘述）

```
1  /*  檔名:ch4_21.java        功能:do-while迴圈的示範  */
2
3  package myJava.ch04;
4  import java.lang.*;
5  import java.util.Scanner;
6
7  public class ch4_21          //主類別
8  {
9      public static void main(String args[])
10     {
11       Scanner keyboardInput = new Scanner(System.in);
12       char inputChar;
13       String inputStr=new String();
14
15       System.out.println("Game Over...");
16       do
17       {
18          System.out.print("Play Again?(y/n)");
19          inputStr=keyboardInput.nextLine();  //讀取鍵盤輸入的一行字串
20          inputChar=inputStr.charAt(0);       //見說明
21       }while((inputChar!='y') && (inputChar!='n'));
22     }
23 }
```

執行結果

```
Game Over...
Play Again?(y/n)q
Play Again?(y/n)k
Play Again?(y/n)y
```


4.3.3 後測式條件迴圈（do-while迴圈敘述）

- 範例說明：
  - (1)第16~21行是do-while迴圈，迴圈內的敘述至少會被執行一次。
  - (2)當我們在Play Again?(y/n)後面按下任何非y或n的鍵，再按下<Enter>鍵時，將重複執行迴圈內的敘述。
    - 當按下y或n鍵時，才會離開迴圈。
  - (3)inputStr是一個字串，在第19行中它被設定為使用者輸入的字串。由於我們只需要字串的最前面一個字元，因此透過第20行將字串中的最前面一個字元取出，並設定給inputChar。
    - 其中，charAt是String類別的實體函式，也就是String物件的方法。其語法如下：

4.3.3 後測式條件迴圈（do-while迴圈敘述）

函式原型：**public char charAt(int index)**
所屬類別：java.lang.String
功能：回傳字串的第index個字元，index由0開始計數。

- 【語法說明】：
  - 若String1的內容為"TEL"
    - 則String1.charAt(0)會回傳'T'
    - String1.charAt(1)會回傳'E'
    - String1.charAt(2)會回傳'L'。

119

4.3.4 迴圈的適用狀況

- 在4.3節中，我們介紹了3種迴圈
  - 您可以在不同狀況下，選擇最方便的迴圈來實作程式需要重複執行的片段，以下是我們的建議。

迴圈	適用時機
for	有初始設定與增量運算時。
while	無初始設定與增量運算時。
do-while	至少執行迴圈內的敘述區塊一次。

120

4.3.4 迴圈的適用狀況

- 至於break與continue敘述的區別則整理如下：

迴圈內的特殊敘述	敘述類型	功能
break	分支敘述 (branch statement)	立刻強制中途跳出迴圈。
continue	分支敘述 (branch statement)	強制略過該次迴圈重複(iteration)尚未被執行的敘述，直接跳到下一個重複(iteration)的條件比較判斷式（while迴圈、do-while迴圈）或增量設定運算式（for迴圈）。

  - 【註】：break敘述也可以用來跳離switch-case敘述。

121

4.4 巢狀與縮排

- 『巢狀』和『縮排』原本是兩件不相干的事。
  - 『巢狀』可以使用在條件敘述，製作巢狀條件敘述；也可以使用在迴圈，製作多重迴圈敘述；或者將條件敘述與迴圈混合使用。
    - 因此，對於編譯器而言，『巢狀』敘述是有意義的。
  - 而『縮排』只不過是為了讓程式設計師更容易看出程式的邏輯結構所做的自發性動作
    - 對於編譯器而言，一個程式是否縮排並不影響程式的正確性。

122

4.4 巢狀與縮排

- 在前面的程式中我們使用了『縮排』來增加程式的可讀性，例如：條件敘述的敘述區塊與迴圈敘述的敘述區塊。
  - 事實上，敘述區塊是由『{}』所包圍，在敘述區塊內則是零個以上的其他敘述。為了方便，有時我們也將敘述區塊稱為區段。
  - 區段必須遵守以下三項原則：
    - 由內圈開始配對。
    - 一個 { 必須配一個 } 。
    - 不可交錯。

123

4.4 巢狀與縮排

- 舉例如圖4-6：

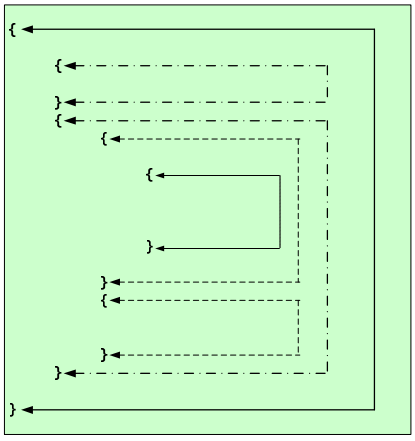


圖4-6 區段配對

124

4.4巢狀與縮排

- 圖4-6的區段縮排方式是為了讓程式設計師比較容易看出每一個區段的對應關係，進而瞭解程式邏輯。
- 但是許多書籍（尤其是Java官方網站的範例）、程式設計師、以及C語言大師Kernighan與Ritchie則有另外一種縮排方式
  - 他們將第一個｛與『條件比較運算式』寫在同一行，形成右列格式，稱之為**K&R縮排**。

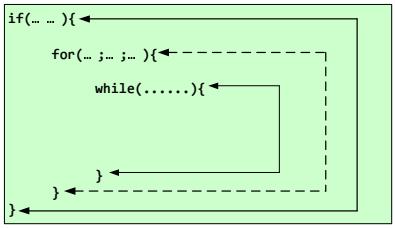


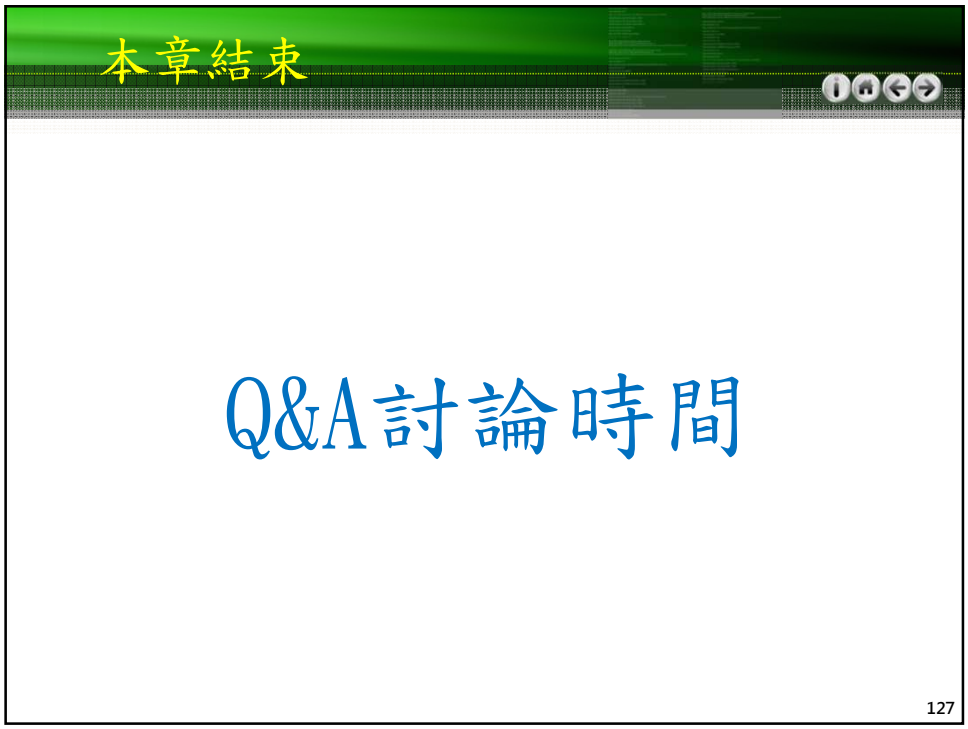
圖4-7 K&R縮排

125

4.4巢狀與縮排

- K&R縮排比較節省行數，但對於初學者而言，不一定比垂直對齊的縮排方式來得清楚。
  - 本書將盡量採用垂直對齊的縮排方式，但對於較大的程式則可能採用K&R縮排，以節省版面。
  - 由於Java官方網站的範例採用的是K&R縮排，因此建議讀者學習完本書後，採用K&R縮排來開發程式，以符合業界的習慣。

126


本章結束
Q&A討論時間
127