基礎語法

學習目標

- 運用基本流程語法

流程控制

- 如果if或else中只有一行陳述句，則{與}可以省略
- 不過為了可讀性與可維護性而言，現在建議是就算只有一行陳述句，也要撰寫{與}明確定義範圍

簡單的if-else語法有兩種基本格式。

- 格式1：　if 運算式　區塊敘述;
- 格式2：　if 運算式　區塊敘述1;
- 　　else　　區塊敘述2;

流程控制

- **if..else**條件式

```
package ch3_1;

public class Odd {

public static void main(String[] args) {
        var input = 10;
        var remain = input % 2; //%：取餘數

        if(remain == 1) {
            System.out.printf("%d 是奇數%n", input);
        }
        else {
            System.out.printf("%d 是偶數%n", input);
        }

}

}
```

結果:

```
10 是偶數
```

流程控制

2.格式 if 條件式　語句組1;

else if 運算式　區塊敘述2;

……

else if 運算式　區塊敘述n-1;

else　區塊敘述n;

輸入Scanner

- Scanner 可以把一整行(line)的字串切成很多的token(切割單元)，預設是以空白(或tab)為基礎隔開，在透過各種next的方法，解析並轉換成我們需要的資料型態，然後回傳。

```
Scanner reader=new Scanner(System.in);
```

- reader物件呼叫下列方法（函數），讀取使用者在命令列輸入的各種資料類型。

nextByte()	找下一個token並轉換成 byte 回傳
nextDouble()	找下一個token並轉換成 double 回傳
nextFloat()	找下一個token並轉換成 float 回傳
nextInt()	找下一個token並轉換成 int 回傳
nextLine()	讀一整行的字串回傳。(包含空白，讀到換行為止)
nextLong()	找下一個token並轉換成 long 回傳
nextShort()	找下一個token並轉換成 short 回傳

流程控制

```
1. public class ch3_2 {
2.   public static void main(String[] args) {// 主程式
3.       java.util.Scanner sc = new java.util.Scanner(System.in); //
4.       System.out.print(" Please enter a grade=> ");
5.       int grade = sc.nextInt();
6.       if ( grade >= 80 ) {                          // if/else/if多選一條件敘述
7.         System.out.println(" student's result :A");   //學生成績
8.       }
9.       else if ( grade >= 70 ) {
10.        System.out.println(" student's result :B");
11.      }
12.      else {
13.        System.out.println(" student's result :C");
14.      }
15.  }
16.}
```

流程控制

```
package ch3_3;
public class ch3_3 {
public static void main(String[] args) {
        var score = 88;
        var level = '\0';
        if(score >= 90) {
            level = 'A';
        }
        else if(score >= 80 && score < 90) {
            level = 'B';
        }
        else if(score >= 70 && score < 80) {
            level = 'C';
        }
        else if(score >= 60 && score < 70) {
            level = 'D';
        }
        else {
            level = 'E';
        }
        System.out.printf("得分等級：%c%n", level);
}
}
```

結果:

```
得分等級：B
```

流程控制

- **switch**條件式格式

```
switch(運算式)  {
    case 常數運算式1:
        區塊敘述1;
        [break;]  (可省略)
    case 常數運算式2:
        區塊敘述2;
        [break;] (可省略)
    default:   (可省略)
        區塊敘述n;
}
```

- switch 語句中的運算式類型可以是： byte、short、int 或者 char。從 Java SE 7 開始，switch 支援字串 String 類型了，同時 case 標籤必須搭配運算式變更為字元、字串或數字。

流程控制

```
package ch3_4;
public class ch3_4 {
public static void main(String[] args) {
        final String warning = "（喔喔！不及格了！）";

        var score = 59;
        var quotient = score / 10;
        var level = switch(quotient) {
            case 10, 9 -> "A";
            case 8     -> "B";
            case 7     -> "C";
            case 6     -> "D";
            default    -> {
                String message = "E" + warning;
                yield message ;
            }
        };
        System.out.printf("得分等級：%s%n", level);

}
}
```

結果:

```
得分等級：E（喔喔！不及格了！）
```

流程控制

- **for**迴圈

格式:

```
for (初始條件; 終止條件; 遞增) {
    區塊敘述;          }
```

initialization：初始條件
termination：終止條件，決定何時迴圈結束
increment：遞增，每次迴圈進行後所要做的事

流程控制

- **for迴圈**

```
package ch3_5;

public class ch3_5 {

public static void main(String[] args) {
        for(var j = 1; j < 10; j++) {
            for(var i = 2; i < 10; i++) {
                System.out.printf("%d*%d=%2d ",i, j,  i * j);
            }
            System.out.println();
         }

}

}
```

結果:

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

流程控制

- **`while`**迴圈

格式:

```
while(運算式)
    區塊敘述;
```

- 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) { ... }

流程控制

- **while迴圈**

```
package ch3_6;

public class ch3_6 {

public static void main(String[] args) {
        while(true) {
            var number = (int) (Math.random() * 10);
            System.out.println(number);

            if(number == 6) {
                System.out.println("找到6");
                break;
            }
        }

}
```

結果:

```
9
3
5
I hit 5....Orz
```

流程控制

- **do..while**迴圈

格式:

```
do
    區塊敘述;
while(運算式);
```

- do...while 敘述先執行區塊內部，執行完畢後才執行括號內的運算式（*expression*），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

流程控制

- **do..while迴圈**

```
package ch3_7;

public class ch3_7 {

public static void main(String[] args) {
        var number = -1;

        do {
            number = (int) (Math.random() * 10);
            System.out.println(number);
        } while(number != 5);
        System.out.println("終於找到5");

}

}
```

結果:

```
4
3
3
7
5
終於找到5
```

流程控制

break語法

- 中斷執行，立即跳出
- 用在 switch、迴圈

流程控制

- break語法

```
package ch3_8;

public class ch3_8 {

public static void main(String[] args) {
// 外層迴圈，outer作為識別字
   outer: for (int i = 0; i < 5; i++) {
        // 內層迴圈
        for (int j = 0; j < 3; j++) {
             System.out.println("i的值為:" + i + "  j的值為:" + j);
             if (j == 1) {
                // 跳出outer標籤所標識的迴圈。
                break outer;
             }
        }
   }

}
```

結果:

```
i的值為:0   j的值為:0
i的值為:0   j的值為:1
```

流程控制

continue語法

continue的功能跳過剩餘的程式碼，直接進行下一次迴圈，並不會終止迴圈 。

流程控制

continue語法

```
package ch3_9;
public class ch3_9 {
public static void main(String[] args) {
   for (int i = 1; i <= 10; i++) {
         if (i == 5) {
            System.out.println("找到目標,繼續迴圈！");
            // 跳出本次迴圈，進入下一次迴圈
            continue;
         }
       System.out.println(i);// 列印當前的i值
   }
}
}
```

結果:

```
找到目標,繼續迴圈！
6
7
8
9
10
```

流程控制

return語法

主要有以下兩種使用格式：

- 單獨一個return關鍵字；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：關鍵字後面可以跟變數、常量或運算式，例如：return 0;

流程控制

return語法

```
package ch3_10;
public class ch3_10 {
    public static void main(String[] args) {
        // 一個簡單的for迴圈
        for (int i = 0; i < 10; i++) {
            System.out.println("i的值是" + i);
            if (i == 5) {
                //返回，結束main方法
                return;
            }
            System.out.println("return後的輸出語句");
        }
    }
}
```

結果:

```
return後的輸出語句
i的值是3
return後的輸出語句
i的值是4
return後的輸出語句
i的值是5
```