Java 使用選擇結構和相關運算子

使用關係與條件運算子

1. 關係 (relational) 運算子

情境	運算子	範例 (int x =1;)
是否 相等	==	System.out.println(x==1);
是否 不相等	!=	System.out.println(x!=1);
是否 小於	<	System.out.println(x<1);
是否 小於 等於	<=	System.out.println(x<=1);
是否 大於	>	System.out.println(x>1);
是否 大於 等於	>=	System.out.println(x>=1);

2. 條件 (conditional) 運算子

情境	運算子	範例 (int x =5, y =9);
且 (and)	&&	System.out.println((x <6) && (y >7));
或 (or)	||	System.out.println((x <6) || (y >7));
非 (not)	!	System.out.println(! (x <6));

3. 字串比較

比較兩個字串是否相同時，使用兩種方法:

1. 使用 == 比較字串是否「指向相同記憶體位置」
2. 使用 equals() 比較字串是否「相同內容」

範例:

```
public class StringCompare {
    public static void main(String[] args) {
        String s1 = "jim";                  //Java產生新的字串物件 "jim"，s1指向該物件
        String s2 = "jim";                  //字串物件都在字串池內，可以反覆使用。所以s2也指向前述物件
        String s3 = new String("jim");      //因為使用new 關鍵字，Java強制在字串池內生成新字串物件，"jim"，
                                            // s3指向該新生物件，所以 s1==s2!=s3
        System.out.println(s1==s2);
        System.out.println(s1==s3);
        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
    }
}

Output :
true
false
true
true
```

使用 if 選擇結構

1. if 選擇結構的種類

(1) 使用 if 關鍵字

語法 :

```
if (boolean_expression) {
  code_block;
}
```

(2) 使用 if 關鍵字 + else 關鍵字

語法:

```
if (boolean_expression) {
  code_block;
} else {
  code_block;
}
```

(3) 使用 if 關鍵字 + else 關鍵字 + else if 關鍵字

語法:

```
if (boolean_expression) {
   code_block;
} else if (boolean_expression) {
  code_block;
}else{
  code_block;
}
```

2. 使用三元運算子

三元(ternary)運算子使用 ? 和 : 等二個運算子將運算式切割為三個運算元

語法 :

(boolean_expression) ? value if true : value if false

範例:

```
public class TernaryOperator {
    public static void main(String[] args) {
        int a, b;
        a = 10;
        b = (a==1)? 20:30;
        System.out.println("Value of b is : " + b);
        if (a==1) {
            b = 20;
        }else {
            b = 30;
        }
        System.out.println("Value of b is : " + b);
    }
}

Output :
Value of b is : 30
Value of b is : 30
```

使用switch 選擇結構

相較於if 選擇結構， switch 選擇結構的程式碼較工整，一目了然。

switch 語法:

```
switch (variable) {
        case literal_value :
```

<code_block>

[break ;]

case another_literal_value :

<code_block>

[break ;]

[default :]

<code_block>

}

1. variable : 變數型態可以是 byte, short , char, int, 和 String
2. literal_value : 變數可能的值(字面常量)
3. default : 當case 舉例的變數值都不滿足時，則進入本程式區塊。相當於 if 結構裡的 else
4. break : 非必要。 遇到break敘述離開該case程式區塊

switch 選擇結構的執行方式:

1. 輸入 variable後，逐一比對每個case區塊的 literal_value 。若相同，程式碼進入該 literal_value所屬的case區塊
2. case 區塊內工作完成後，預期應該要有break敘述，讓程式碼跳出該case區塊。若沒有break敘述，程式碼會往下開始逐行執行所有case區塊的程式碼，直到遇到break敘述，或是switch結構結束

範例 :

```
public class Q1 {

    public static void main(String[] args) {
       String color = "Purple";
       switch(color) {
           case "Purple":
              System.out.println("Found Purple");
           case "Blue":
              System.out.println("Found Blue");
              break;
           case "White":
              System.out.println("Found White");
              break;
           default:
              System.out.println("Default");
       }

    }

}

// Output:
// Found Purple
```

```
// Found Blue
```