列舉

Enum(列舉) 在 C 語言時代就是賦予常數值可讀意義的簡便方法，C# 也是一開始就提供 Enum 型別 ，Java 則遲到 5.0 才提供

基於 Java 語言的特性，Enum 應用在方法的參數傳遞與回傳值上，具有提高表達能力以及強制內容檢查的兩種好處

在你直接以數值代表參數的某些意義時，你可能要提防方法調用者傳一些預期以外的參數值，你使用 Enum 之後，除了賦予參數值可讀的意義之外，亦強制調用者只能傳你列舉的參數值進來，你不再需要關心非預期的參數值，同理，當你使用 Enum 處理回傳值時，方法調用者也更易懂也能更簡單地處理你的回傳值

列舉本身就是一種型別，在 Java 語言中則完完全全地是個類別，從 getClass() 中即可看出來

```java
//沒有使用列舉
public static int where(int type) {
    if (type == 1) {
        //something
    } else if (type == 2) {
        //something
    } else {
        //3,4,5,... 沒意義的值，錯誤的值。
        return 1; //??
    }
    return 0; //什麼意思？
}
```

```java
//使用列舉
public enum WhereType {

    Asc, Desc
}

public enum ReturnCode {

    Ok, Error
}

public ReturnCode where(WhereType type) {
    if (type == WhereType.Desc) {
        //something
    } else if (type == WhereType.Asc) {
        //something
    }       //不用再處理 3,4,5 這些錯誤的值。
    return ReturnCode.Ok;
}
```

```java
public class EnumeratedTypes1 {

    public enum Month {

        JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
    };

    public static void main(String[] args) {
        for (Month month : Month.values()) {
            System.out.print(month + " ");
        }
        System.out.println();
        Month month = Month.JAN;
        System.out.println("Month.JAN==>" + month);
        System.out.println("month.getClass()==>" + month.getClass());
    }
}
```

```
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC
Month.JAN==>JAN
month.getClass()==>class java18_列舉.EnumeratedTypes1$Month
```

上例中的輸出結果可看出，Month 是一個類別，可稱之為列舉類別

命名空間的問題

在 Java 中，因為完完全全以類別來處理，所以我們在設定時必須要加上類別名稱，就像是在存取類別中靜態常數值一樣，從而避免了命名衝突的情況

```
public class EnumeratedTypes2 {

    enum Color {

        RED, GREEN, BLUE, ORANGE
    };

    enum Fruit {

        BANANA, ORANGE, PEACH
    };

    public static void main(String[] args) {
      // Color color = ORANGE; // 無法通過編譯
        Color color = Color.ORANGE; // 必須指定類別才能通過編譯
        System.out.println("Color.ORANGE==>" + color);
        Fruit fruit = Fruit.ORANGE;
        System.out.println("Fruit.ORANGE==>" + fruit);
    }
}
```

```
Color.ORANGE==>ORANGE
Fruit.ORANGE==>ORANGE
```

列舉範例

```
enum Color1 {

    Red, Green, Blue, Black;
}
```

```
public static void 列舉1() {
    Color1 x;
    x = Color1.Red;
    System.out.println("x=" + x);
    System.out.println("Color1.Red=" + Color1.Red);
}
```

```
x=Red
Color1.Red=Red
```

```
public class EnumDemo1 {

    enum Color2 {

        Red, Green, Blue;
    }
}
```

```
public static void 列舉2() {

    EnumDemo1.Color2 x;
    x = EnumDemo1.Color2.Red;
    System.out.println("x=" + x);
    System.out.println("EnumDemo1.Color2.Red=" + EnumDemo1.Color2.Red);
}
```

```
x=Red
EnumDemo1.Color2.Red=Red
```

```java
//列舉不能被定義在方法內
class EnumDemo2 {

    void abc() {
//        enum Color4 {
//
//            Red, Green, Blue;
//        }
    }
}
```

```java
enum CoffeeSize1 {

    BIG, HUGE, OVERWHELMING
}

class Coffee1 {

    CoffeeSize1 size;
}
```

```java
public static void 列舉3() {
    Coffee1 drink = new Coffee1();
    drink.size = CoffeeSize1.HUGE;
    System.out.println("drink.size=" + drink.size);
}
```

```
drink.size=HUGE
```

```java
class Coffee2 {

    enum CoffeeSize2 {

        BIG, HUGE, OVERWHELMING
    }
    CoffeeSize2 size;
}
```

```java
public static void 列舉4() {
    Coffee2 drink = new Coffee2();
    drink.size = Coffee2.CoffeeSize2.BIG;
    System.out.println("drink.size=" + drink.size);
}
```

```
drink.size=BIG
```

列舉與 switch

```java
public static void 列舉的值可使用在switch中() {
    Color1 x = Color1.Red;

    switch (x) {
        case Blue:
            System.out.println("Blue");
            break;
        case Red:
            System.out.println("Red");
            break;
        case Black:
            System.out.println("Black");
    }
}
```

```
Red
```

列舉與集合

```java
enum Example {

    ONE, TWO, THREE
}
```

```java
public static void 列舉與Map() {
    //列舉可以放在 Map 裡當 values
    Map<Integer, Example> m = new HashMap<Integer, Example>();
    m.put(1, Example.ONE);
    m.put(2, Example.TWO);
    m.put(3, Example.THREE);

    System.out.println(m);
}
```

```
{1=ONE, 2=TWO, 3=THREE}
```

```java
public static void 列舉與TreeSet() {
    //列舉可以放在 TreeSet 並且可排序
    Set<Example> s = new TreeSet<Example>();
    s.add(Example.TWO);
    s.add(Example.THREE);
    s.add(Example.ONE);
    s.add(Example.Four);
    System.out.println(s);
}
```

```
[ONE, TWO, THREE]
```

```java
public enum Count {
    加, 減, 乘, 除;
}
```

```java
public static void 呼叫四則運算() {
    int x = 10;
    int y = 5;

    System.out.println(四則運算(x, y, Count.加));
    System.out.println(四則運算(x, y, Count.減));
    System.out.println(四則運算(x, y, Count.乘));
    System.out.println(四則運算(x, y, Count.除));
}

public static int 四則運算(int x, int y, Count p) {

    switch (p) {

        case 加:
            return x + y;
        case 減:
            return x - y;
        case 乘:
            return x * y;
        case 除:
            return x / y;
        default:
            return 0;
    }
}
```

```
15
5
50
2
```