

# Java Flow Control

## 流程控制

Java Fundamental



# Content

---

- ◆ 使用者輸入
- ◆ if-else 陳述句
- ◆ switch-case 陳述句

# Content

---

- ◆ 使用者輸入
- ◆ if-else 陳述句
- ◆ switch-case 陳述句

# 使用者輸入

---

## ◆ 標準輸入 System.in

- System 類別的標準輸入成員 in (InputStream型態)
- 預設資料來源裝置是鍵盤(Keyboard)
- 提供的 read() 方法：一次取得一個位元的資料,資料以int型態傳回
- JDK1.5 新增 java.util.Scanner 類別,可以簡便的取得使用者輸入
- 語法：`java.util.Scanner sc = new java.util.Scanner (System.in);`
- 使用Scanner所提供的nextXXX()方法取得使用者輸入的各式資料

# 使用者輸入

```
01 public class ScannerDemo {
02     public static void main(String args[]) {
03         java.util.Scanner scanner = new java.util.Scanner(System.in);
04
05         System.out.print("輸入整數:");
06         int input1 = scanner.nextInt();
07
08         System.out.print("輸入浮點數:");
09         double input2 = scanner.nextDouble();
10
11         System.out.print("輸入布林值:");
12         boolean input3 = scanner.nextBoolean();
13
14         System.out.print("輸入字串:");
15         String input4 = scanner.next();
16
17         System.out.println("整數輸入:" + input1);
18         System.out.println("浮點數輸入:" + input2);
19         System.out.println("布林值輸入:" + input3);
20         System.out.println("字串輸入:" + input4);
21     }
22 }
```



系統管理員: 命令提示字元

```
C:\JavaClass>javac ScannerDemo.java
C:\JavaClass>java ScannerDemo
輸入整數:123
輸入浮點數:3.14159
輸入布林值:false
輸入字串:Hello
整數輸入:123
浮點數輸入:3.14159
布林值輸入:false
字串輸入:Hello
C:\JavaClass>
```

# Content

---

- ◆ 使用者輸入
- ◆ if-else 陳述句
- ◆ switch-case 陳述句

# Java 流程控制

---

- ◆ 流程控制是配合Java關係與條件運算式的條件來執行不同程式區塊，或重複執行指定區塊的程式碼。

- ◆ 流程控制主要分為兩種：

- 條件控制 (Branching Statement)

- 條件控制是一個選擇題，分為是否選、二選一或多選一，依照運算式的結果來決定執行哪一個程式區塊的程式碼。

- 迴圈控制 (Looping Statement)

- 迴圈控制就是重複執行程式區塊的程式碼，擁有一個結束條件可以結束迴圈的執行。

# 條件控制 – 說明

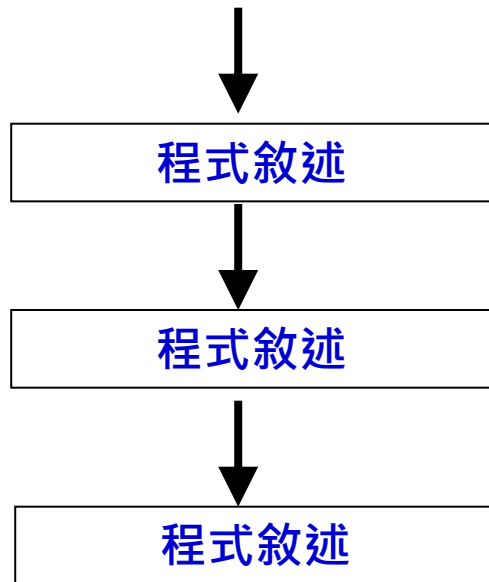
---

- ◆ **Java**條件控制敘述是使用關係和條件運算式，配合程式區塊建立的決策敘述。
- ◆ 條件控制可以分為以下幾種：
  - 是否選 ( **if** ) 。
  - 二選一 ( **if/else** ) 。
  - 多選一 ( **switch** ) 。
  - 三元運算式(條件敘述運算子) ( **?:** ) 可以建立單行程式碼的條件控制。



# Control Flow

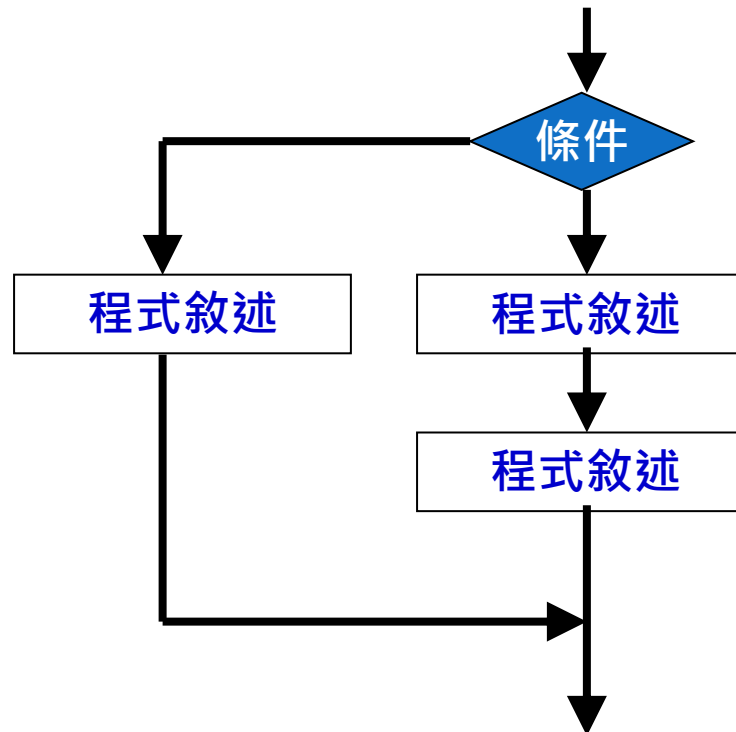
## ◆ 循序結構



## ◆ 選擇結構 (Branching Statement)

➤ if-else

➤ switch

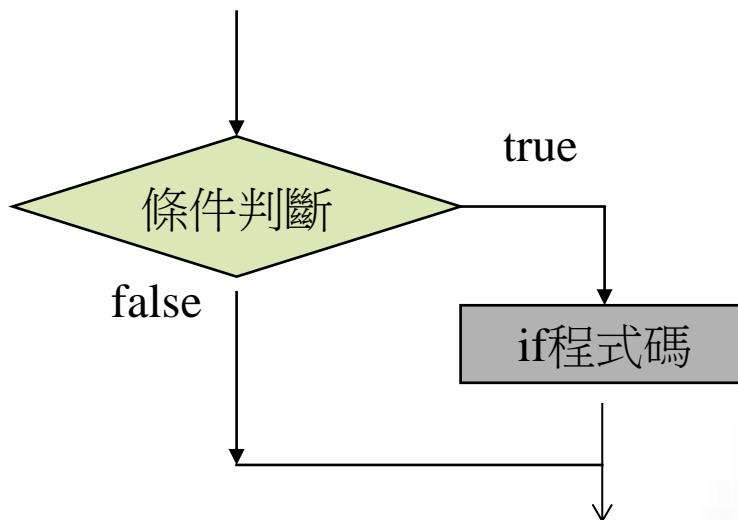


# 條件控制 – if

## ◆ if 是否選條件敘述

- if 條件敘述是一種是否執行的單選題，只是決定是否執行程式區塊的程式碼。
- 如果關係/條件運算結果為true，就執行括號之間的程式碼。
- 例如：判斷成績是否及格的if條件敘述，如下所示：

```
int grade = 80;  
  
if ( grade >= 60 ) {  
    str += "成績及格....." +  
        "\n分數: " + grade;  
}
```



# Branching Statements – if

```
if (boolean) {  
    statements;  
}
```

```
01 public class OddTest {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         System.out.print("輸入整數: ");  
07         int input = scanner.nextInt();  
08  
09         if(input % 2 == 0) //如果餘數為 0  
10             System.out.println(input + " 是偶數");  
11  
12         if(input % 2 != 0) //如果餘數不為 0  
13             System.out.println(input + " 是奇數");  
14     }  
15 }
```



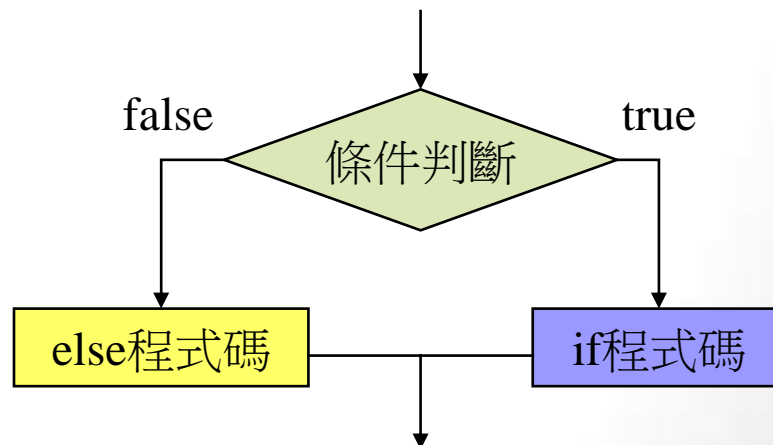
```
C:\JavaClass>java OddTest  
輸入整數: 5  
5 是奇數  
  
C:\JavaClass>java OddTest  
輸入整數: 6  
6 是偶數  
  
C:\JavaClass>
```

# 條件控制 – if/else

## ◆ if/else 二選一條件敘述

- 如果條件是排它情況，只能二選一，我們可以加上**else**。
- 如果if條件的關係/條件運算式為**true**，就執行if程式區塊；
- **false**執行**else**程式區塊。
- 例如：判斷成績是否及格且是指定課程的if/else條件敘述，如下所示：

```
int grade = 80;  
char type = 'm';  
if ( grade >= 60 && type == 'm' ) {  
    str += "課程: " + type +  
        "\n成績及格: " + grade;  
}  
else  
    str += "課程不正確或成績不及格";
```



# Branching Statements – if / else

```
if (boolean) {  
    statements;  
} else {  
    statements;  
}
```

```
01 public class OddTest2 {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         System.out.print("輸入整數: ");  
07         int input = scanner.nextInt();  
08  
09         if(input % 2 == 0) {  
10             System.out.println(input + " 是偶數");  
11         } else {  
12             System.out.println(input + " 是奇數");  
13         }  
14     }  
15 }
```



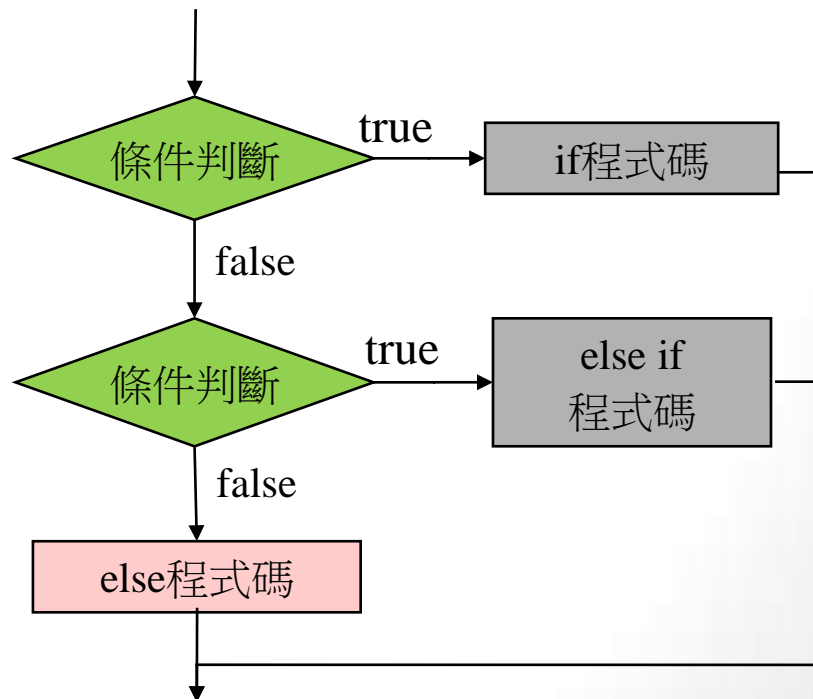
```
c:\JavaClass>java OddTest2  
輸入整數: 7  
7 是奇數  
  
c:\JavaClass>java OddTest2  
輸入整數: 8  
8 是偶數  
  
c:\JavaClass>
```

# 條件控制 – if/else/if

## ◆ if/else/if 多選一條件敘述

- 程式如果需要多選一條件敘述，也就是依照一個條件判斷來執行多個區塊之一的程式碼，只需重複使用if/else條件，就可以建立多選一條件敘述。
- 例如：判斷學生GPA成績的條件敘述，如下所示：

```
int grade = 80;  
if ( grade >= 80 )  
    str += "學生成績A";  
else  
    if ( grade >= 70 )  
        str += "學生成績B";  
    else  
        str += "學生成績C";
```



# Branching Statements– if/else if/else

```
if (boolean) {  
    statements;  
} else if (boolean) {  
    statements;  
} else {  
    statements;  
}
```

```
01 public class ScoreLevel {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         System.out.print("輸入分數: ");  
07         int score = scanner.nextInt();  
08  
09         if(score >= 90)  
10             System.out.print("得 A ");  
11         else if(score >= 80 && score < 90)  
12             System.out.print("得 B ");  
13         else if(score >= 70 && score < 80)  
14             System.out.print("得 C ");  
15         else if(score >= 60 && score < 70)  
16             System.out.print("得 D ");  
17         else  
18             System.out.print("得 E(不及格)");  
19     }  
20 }
```



```
系統管理員: 命令提示字元  
c:\JavaClass>java ScoreLevel  
輸入分數: 83  
得 B  
c:\JavaClass>java ScoreLevel  
輸入分數: 58  
得 E<不及格>  
c:\JavaClass>
```

# 三元運算子 (Ternary Operator)

## ◆ 三元運算子(Ternary Operator)

➤ 概念類似if-else 條件敘述

➤ 使用語法

$X = (\text{布林運算式}) ? \text{true-value} : \text{false-value}$

➤ 當運算式回傳值為 **true** 的時候，會進行冒號 左邊的敘述 (**true-value**) 給 X。

➤ 當運算式回傳值為 **false** 的時候，會進行冒號 右邊的敘述 (**false-value**) 給 X。

```
String s = "";  
int i = 0, j = 1;  
s = (i < j) ? "正確" : "錯誤";  
↑  
true
```

System.out.println("s=" + s);      執行結果：s=正確



# 三元運算子 (Ternary Operator)

```
01 public class OddTest3 {  
02     public static void main(String[] args) {  
03         java.util.Scanner scanner =  
04             new java.util.Scanner(System.in);  
05  
06         System.out.print("輸入整數: ");  
07         int input = scanner.nextInt();  
08  
09         int remain = input % 2;  
10  
11         System.out.println(input + " 是" +  
12             ((remain==0) ? "偶數" : "奇數"));  
13  
14     }  
15 }
```



```
C:\JavaClass>java OddTest3  
輸入整數: 9  
9 是奇數  
  
C:\JavaClass>java OddTest3  
輸入整數: 10  
10 是偶數  
  
C:\JavaClass>
```

# Exercise

---

```
01 int money = 100;  
02 if(money = 100) {  
03     System.out.println("等於 100");  
04 }  
05 else {  
06     System.out.println("不等於 100");  
07 }
```

執行結果：編譯錯誤

# Exercise

---

```
01 boolean b = false;
02 if(b = false) {
03     System.out.println("false");
04 }
05 else {
06     System.out.println("true");
07 }
```

# Exercise

---

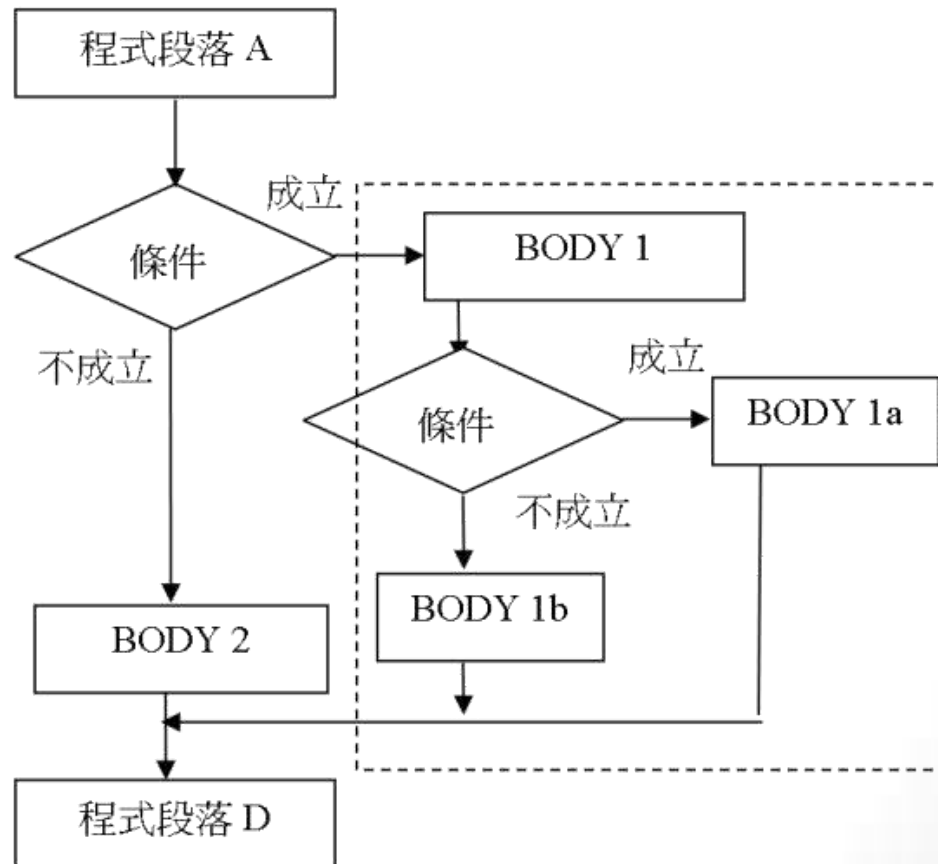
```
01 boolean b = false;  
02 if(b = false) {  
03     System.out.println("false");  
04 }  
05 else {  
06     System.out.println("true");  
07 }
```

執行結果：true

# Branching Statements – if / else

- ◆ 較複雜的情況，會使用巢狀 if-else 敘述

```
if (expression)
{
    BODY 1
    if(expression)
    {
        BODY 1a
    }
    else
    {
        BODY 1b
    }
}
else
{
    BODY 2
}
```



# 巢狀結構

◆ 較複雜的情況，會使用巢狀 if-else 敘述

```
if (...) {  
    ...  
    if (...) {  
        ...  
    } else {  
        ...  
    }  
} else {  
    ...  
}
```

```
if (...) {  
    ...  
} else {  
    ...  
    if (...) {  
        ...  
    } else {  
        ...  
    }  
}
```

# if – else 配對

---

- ◆ 當程式碼只有一行時，括號可省略
- ◆ else 配對時，應由前面的先配對
- ◆ else 先和最靠近自己的 if 配對
- ◆ 若最靠近的 if 已經配對了，則找次靠近者

```
int a = 0, i = 1, j = -1, k = 2;  
if ( i > 0 )  
    if ( j > 0 )  
        if ( k > 0 )  
            a = 100;  
        else  
            a = 200;  
    else  
        a = 300;  
System.out.println("a= " + a);
```

# Content

---

- ◆ 使用者輸入
- ◆ if-else 陳述句
- ◆ switch-case 陳述句



# 條件控制 – switch

## ◆ switch statement

- expression的數值，將會被拿來比對下面每一個case的value
- 程式將會跳至第一個比對成功(數值相等)的地方繼續執行，假如遇到break;敘述，就結束整個switch敘述
- 否則將會一直往下比對。沒有比對成功的話，則執行default:之後的程式段
- 沒遇到break時，會繼續執行下一行指令敘述

```
1. switch(expression)
   {
2.     case value1:
3.         statements...
4.         break;
5.     case value2:
6.         statements...
7.         break;
8.     ...
9.     default:
10.        statements...
11.        break;
12. }
```

# 條件控制 – switch

## ◆ 分支結構

- 針對單一變數進行，與條件判斷類似。

## ◆ 鍵值須為可自動轉換成int的型別

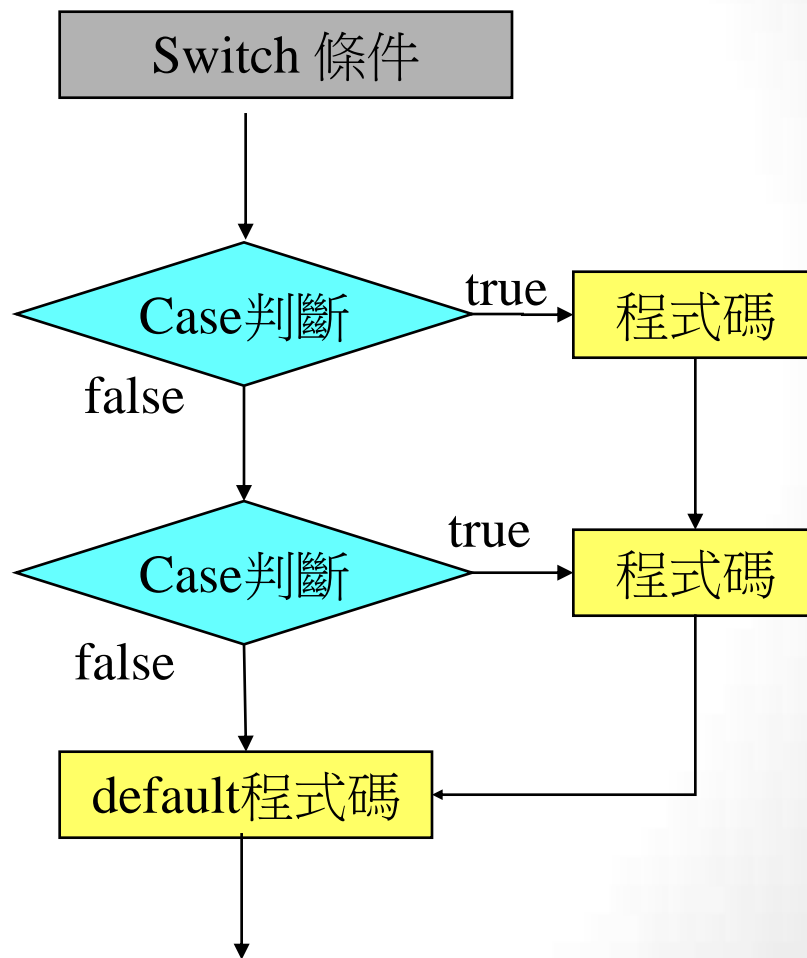
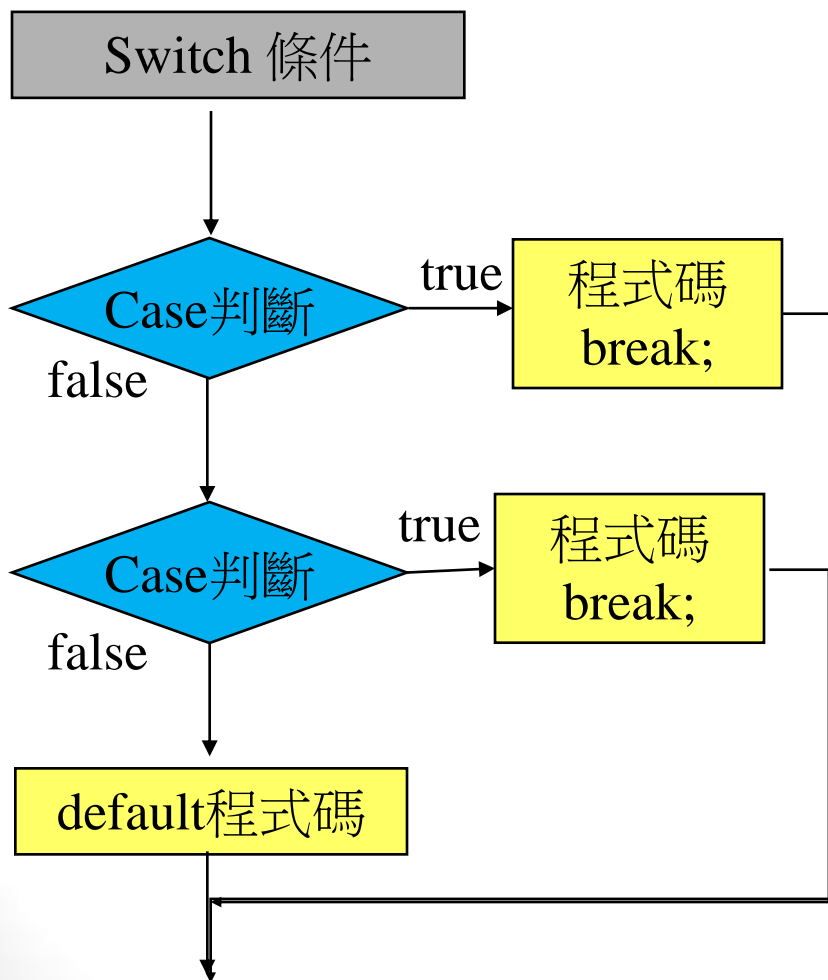
- byte, short, char, int。

## ◆ 運作特性

- case 和 default 視為標籤，其順序沒有限制。
- case 後面只能接常數或是常數的運算式，不能包含變數。
- 鍵值會和所有 case 標籤一一比對。
- 當鍵值和所有 case 標籤比對過，而且沒有符合的 case 標籤時，default 標籤以下的敘述會被執行。

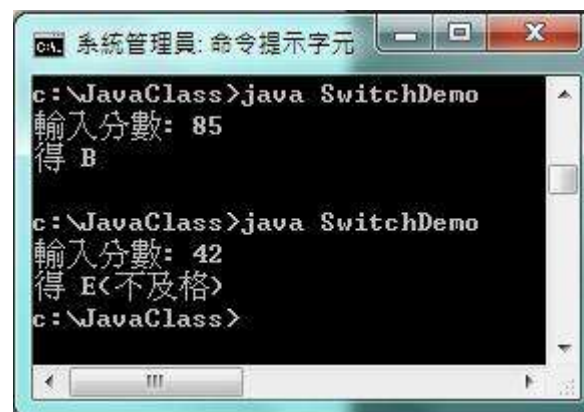
```
1. switch(expression)
   {
2.     case value1:
3.         statements...
4.         break;
5.     case value2:
6.         statements...
7.         break;
8.     ...
9.     default:
10.        statements...
11.        break;
12. }
```

# 條件控制 – switch



# 條件控制 – switch

```
01 public class SwitchDemo {
02     public static void main(String[] args) {
03         java.util.Scanner scanner =
04         new java.util.Scanner(System.in);
05         System.out.print("輸入分數: ");
06         int score = scanner.nextInt();
07         int level = (score/10);
08
09         switch(level) {
10             case 10:
11             case 9:
12                 System.out.println("得 A ");
13                 break;
14             case 8:
15                 System.out.println("得 B ");
16                 break;
17             case 7:
18                 System.out.println("得 C ");
19                 break;
20             case 6:
21                 System.out.println("得 D");
22                 break;
23             default:
24                 System.out.print("得 E(不及格)");
25         }
26     }
27 }
```



```
C:\>系統管理員: 命令提示字元
c:\JavaClass>java SwitchDemo
輸入分數: 85
得 B

c:\JavaClass>java SwitchDemo
輸入分數: 42
得 E<不及格>
c:\JavaClass>
```

# 條件控制 – 常見錯誤

---

```
01 int x = 1;
02 switch(x) {
03     case 1:
04         System.out.print("A");
05         break;
06     case 2:
07         System.out.print("B");
08 }
```

# 條件控制 – 常見錯誤

---

```
01 int x = 1;  
02 switch(x) {  
03     case 1:  
04         System.out.print("A");  
05         break;  
06     case 2:  
07         System.out.print("B");  
08 }
```

執行結果：A

# 條件控制 – 常見錯誤

---

```
01 int x = 1;
02 switch(x) {
03     case 1:
04         System.out.print("A");
05     case 2:
06         System.out.print("B");
07 }
```

# 條件控制 – 常見錯誤

---


```
01 int x = 1;  
02 switch(x) {  
03     case 1:  
04         System.out.print("A");  
05     case 2:  
06         System.out.print("B");  
07 }
```

執行結果：AB



# 條件控制 – 常見錯誤

---

```
01 char x = 'A';  
02 char valueA = 'A';  
03 switch(x) {  
04     case valueA:  編譯錯誤!  
05         System.out.print("A");  
06         break;  
07     case 'B':  
08         System.out.print("B");  
09 }
```

# 條件控制 – 常見錯誤

---

```
01 char x = 'A';  
02 final char valueA = 'A';  
03 switch(x) {  
04     case valueA:  
05         System.out.print("A");  
06         break;  
07     case 'B':  
08         System.out.print("B");  
09 }
```

# 條件控制 – 常見錯誤

---

```
01 char x = 'A';  
02 final char valueA = 'A';  
03 switch(x) {  
04     case valueA:  
05         System.out.print("A");  
06         break;  
07     case 'B':  
08         System.out.print("B");  
09 }
```

執行結果：A

Q & A