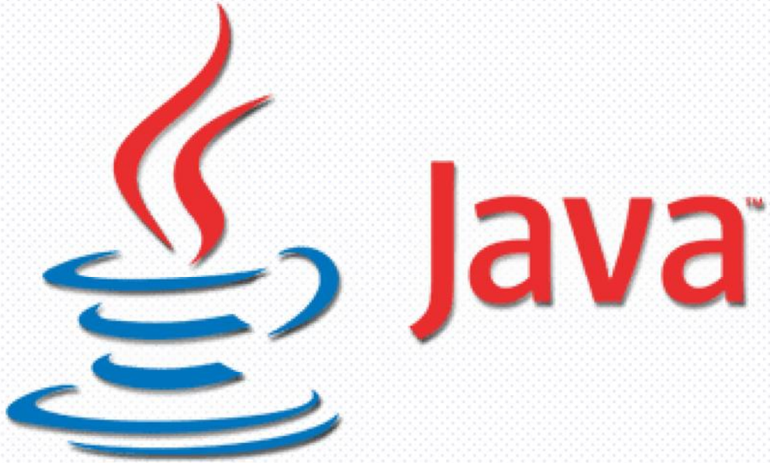
流程控制 - 條件/分支控制

Flow Control - Conditional/Branching Statement



Java Fundamental

Outline

- 使用者輸入
- if-else
- switch-case

Outline

- 使用者輸入
- if-else
- switch-case

使用者輸入

- 標準輸入 System.in
  - 預設資料來源裝置是鍵盤(Keyboard)
  - 語法： java.util.Scanner 名稱 = new java.util.Scanner (System.in);

    java.util.Scanner scanner = 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

    Ex. int input = scanner.nextInt();

使用者輸入

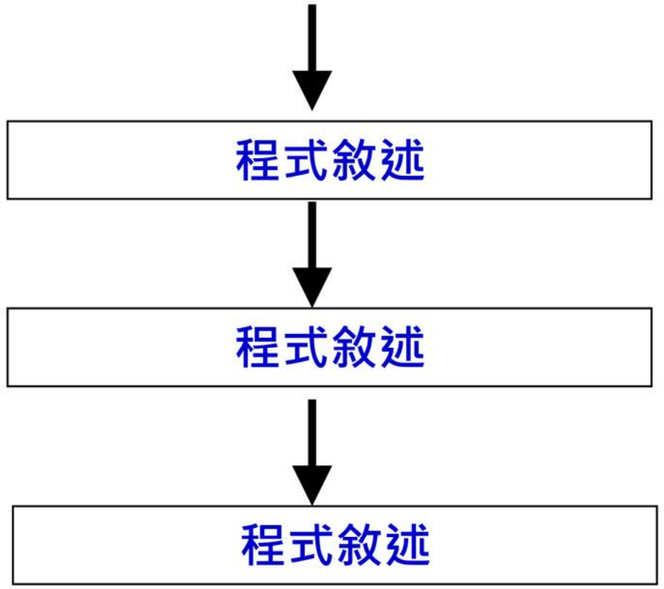
```
public class ScannerDemo {
public static void main(String args[]) {
java.util.Scanner scanner = new java.util.Scanner(System.in);

System.out.print("輸入整數:");
int input1 = scanner.nextInt();

System.out.print("輸入浮點數:");
double input2 = scanner.nextDouble();

System.out.print("輸入布林值:");
boolean input3 = scanner.nextBoolean();

System.out.print("輸入字元:");
char input4 =scanner.next().charAt(0) ;

System.out.print("輸入字串:");
String input5 =scanner.next() ; // nextLine

System.out.println("整數輸入:" + input1);
System.out.println("浮點數輸入:" + input2);
System.out.println("布林值輸入:" + input3);
System.out.println("字元輸入:" + input4);
System.out.println("字串輸入:" + input5);

scanner.close();
}
}
```

Outline

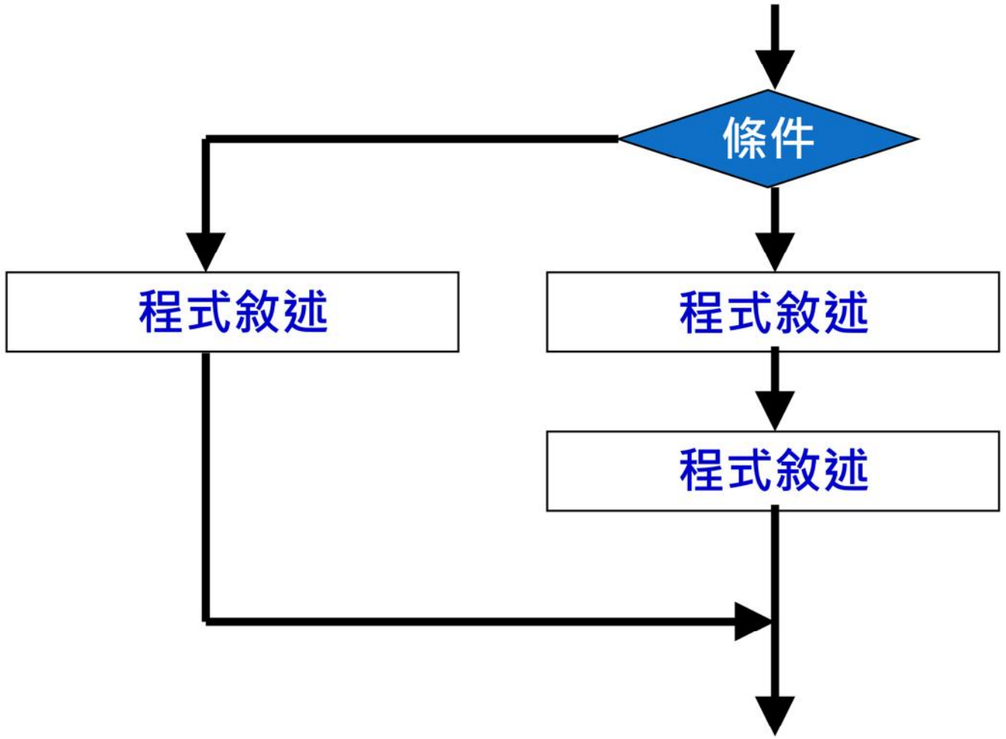
- 使用者輸入
- if-else
- switch-case

Java 流程控制 (Control Flow)

- 循序結構



- 選擇結構 (Branching Statement)
  - if-else
  - switch



Java 流程控制

- 流程控制是Java根據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。
- 流程控制分為兩種：
  - 條件控制 (Branching Statement)

    條件控制是一個選擇題，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。

Branching Statements – if

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

```
01  public class OddTest {
02      public static void main(String[] args) {
03
04        java.util.Scanner scanner =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  }
```

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 = new java.util.Scanner(System.in);
04
05          System.out.print("輸入整數: ");
06          int input = scanner.nextInt();
07
08          if(input % 2 == 0) {
09              System.out.println(input + " 是偶數");
10          } else {
11              System.out.println(input + " 是奇數");
12           }
13      }
14  }
15
```

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

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 = (布林運算式) **?** **true-value** **：** **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 **flase** 的時候，會進行冒號 右邊的敘述 **(false-value)** 給 **X**。

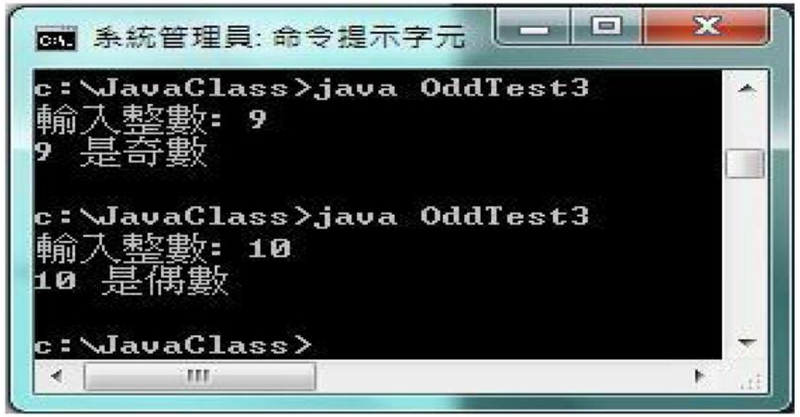
```
String s = "";
int i = 0, j = 1;
s = (i < j) ? "正確" : "錯誤";
System.out.println("s=" + s);
```

true

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
```



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**

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);
```

Outline

- 使用者輸入
- if-else
- switch-case

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。
  - 若最後都沒有比對成功的話，就會執行default:之後的程式段。
  - 沒遇到break時，會繼續執行下一行指令敘述

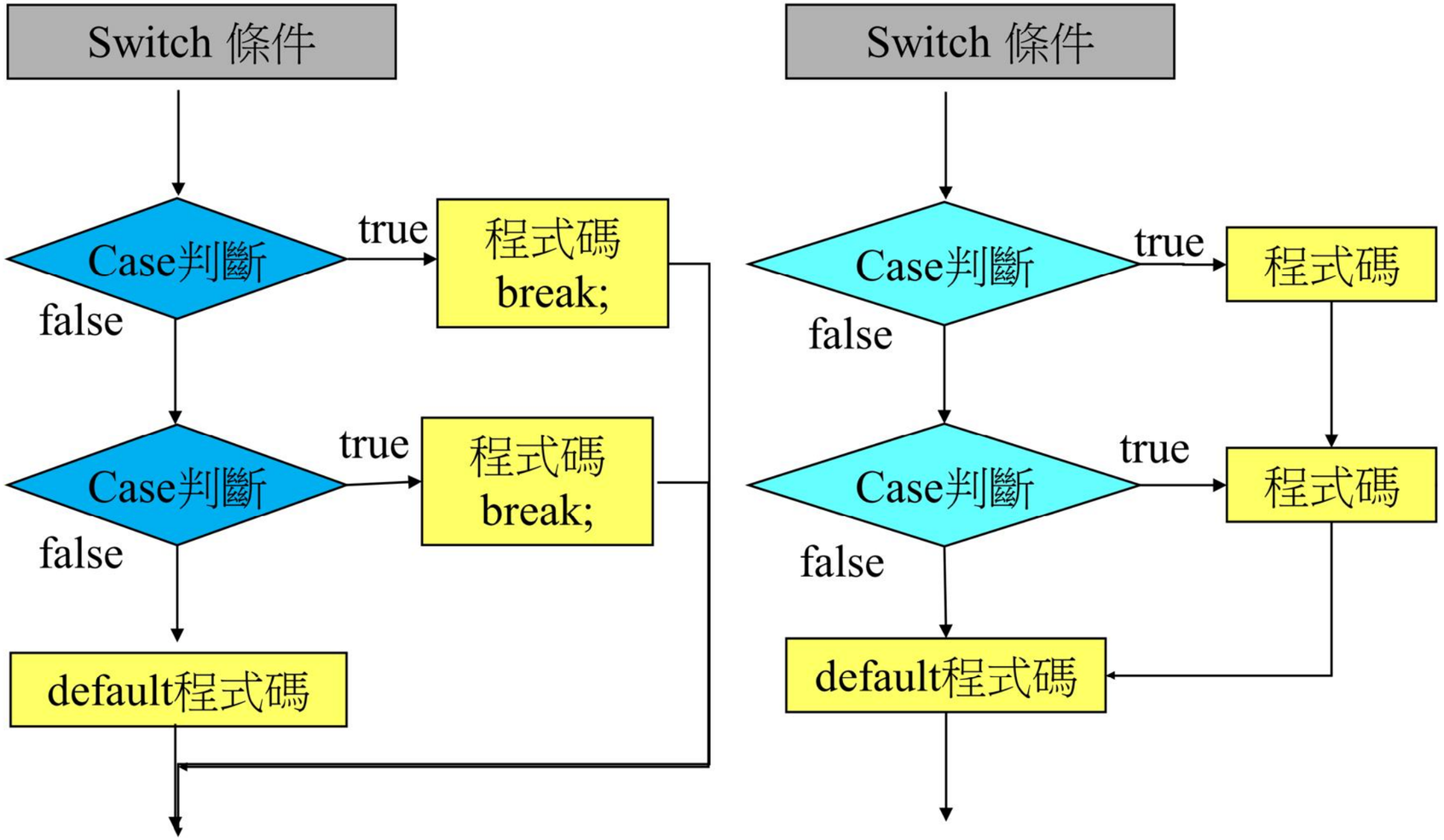
```
1. switch(expression)
   {
2.    case value1:
3.       statements…
4.       break;
5.    case value2:
6.       statements…
7.       break;
8.    …
9.    default:
10.      statements…
11.      break;
12.}
```

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的型別或是字串型別
  - byte、short、char、int、String
- 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


Switch 條件
Case判斷
true
程式碼
break;
false
Case判斷
true
程式碼
break;
false
default程式碼
Switch 條件
Case判斷
true
程式碼
false
Case判斷
true
程式碼
false
default程式碼

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  }
```

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:
05     System.out.print("A");
06     break;
07   case 'B':
08     System.out.print("B");
09 }
```

編譯錯誤！

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