基礎程式設計（Java）　講義

控制敘述（下）：for、do-while、switch 與邏輯運算子

Deitel & Deitel《Java How to Program: Early Objects》11e, Ch.5

§ 1　一、名詞速查表

中文	English	一句話定義	科目	章節
for 迴圈	for loop	把初始化、條件、更新寫在同一行的計數器迴圈	Java	CH5
初始化	initialization	for 第一格，迴圈開始前只做一次的設定	Java	CH5
迴圈條件	loop-continuation condition	for 第二格，為真才繼續、每輪先檢查	Java	CH5
更新	increment / update	for 第三格，每輪迴圈體跑完後執行	Java	CH5
do-while 迴圈	do-while loop	先做一次再檢查條件，至少執行一次	Java	CH5
switch 多選	switch statement	拿一個值逐一比對 `case`，相符就執行	Java	CH5
case 標籤	case label	switch 內的比對值，相符從此處開始執行	Java	CH5
貫穿	fall-through	case 沒寫 `break` 會接著往下個 case 執行	Java	CH5
break	break	立刻跳出最內層 `switch` 或迴圈	Java	CH5
continue	continue	跳過本輪剩餘迴圈體，直接進下一輪	Java	CH5
邏輯且	logical AND `&&`	兩邊都真才真；左為假就短路不看右邊	Java	CH5
邏輯或	logical OR `||`	任一邊真就真；左為真就短路不看右邊	Java	CH5
邏輯非	logical NOT `!`	把布林值反轉，真變假、假變真	Java	CH5
短路	short-circuit	結果已確定就不再計算右運算元	Java	CH5
不短路且／或	boolean `&` / `|`	一定算兩邊（含副作用）的且／或	Java	CH5
互斥或	XOR `^`	兩邊布林值不同才真、相同為假	Java	CH5

§ 2 二、核心概念

CH4 學會 `while`；本章補上其餘重複與選擇結構，並學會用**邏輯運算子**把多個條件組合成一個 `boolean`。

> `for` 是計數器迴圈的緊湊寫法：把「初始化、條件、更新」三件事擠進一行，最適合**已知次數**。`do-while` 把條件檢查移到迴圈尾，所以迴圈體**至少執行一次**。`switch` 在「拿一個值比對多個固定選項」時，比一長串 `if-else if` 清楚。

選擇與重複都靠 `boolean` 條件決定流向。當條件不只一個（例如「分數 ≥ 60 **而且** ≤ 100」），用 `&&`、`||`、`!` 把它們組合起來；`&&`、`||` 會**短路**（左邊已能決定結果就不算右邊），這既省事、也能避免像「先檢查不為 null 再取用」這類錯誤。

§ 3 三、主要內容

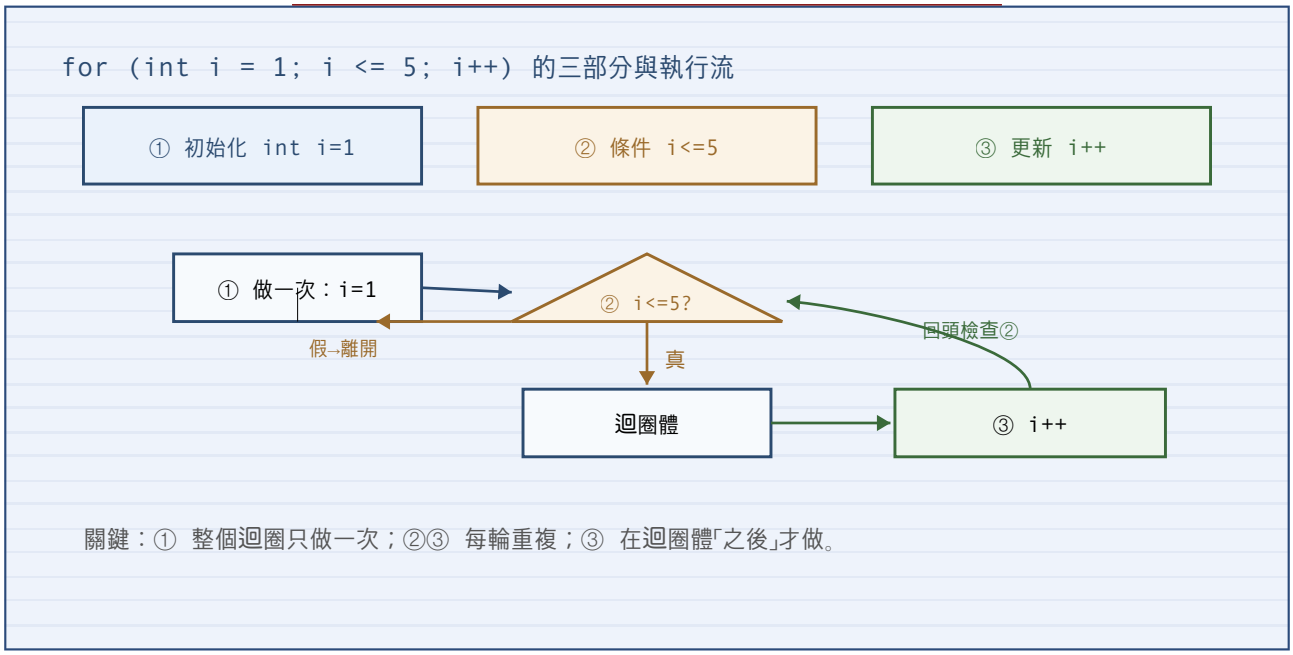
3.1 1. for 迴圈：計數器三件事擠一行

`while` 寫計數器要分三處（初值、條件、`i++`）；`for` 把這三件事放在同一行，一眼看清迴圈次數。

```
for (int i = 1; i <= 10; i++)          // 初始化; 條件; 更新
    System.out.println(i);             // 迴圈體（單句免大括號）

for (int i = 1; i <= 10; i++) {        // 多句要用大括號
    System.out.printf("%d ", i);
    sum += i;
}
```

執行順序：**初始化做一次** → 檢查條件 → （真）做迴圈體 → 做更新 → 回頭檢查條件……（假）離開。下圖拆解這三格與執行流。



`for` 的三格都可省略或變化：更新可用 `i += 2`（每次加 2）、`i--`（倒數）；條件留空等於永真（要靠 `break` 跳）。`for` 內宣告的 `i`（`for (int i...`）作用域只到迴圈結束，迴圈外不可用。

3.2 2. do-while：先做再檢查

`while` 是「先檢查再做」，條件一開始就假則一次都不做；`do-while` 把條件移到尾巴，**迴圈體至少跑一次**，再決定要不要繼續。

```
int i = 1;
do {
    System.out.printf("%d ", i);    // 先做
    i++;
} while (i <= 5);                   // 再檢查（注意結尾分號）
```

適用「先給一次、再看要不要重複」的場景（如選單：先顯示一次再問要不要重選）。

3.3 3. switch 多選：拿一個值比對多個選項

一個整數／字元／字串要對應多個固定選項時，`switch` 比一串 `if-else if` 清楚。把成績等第轉文字：

```
char grade = 'B';
switch (grade) {
    case 'A':
        System.out.println("優"); break;
    case 'B':
        System.out.println("甲"); break;      // 命中這裡
    case 'C':
        System.out.println("乙"); break;
    default:
        System.out.println("其他");            // 都不符時
}
```

- 每個 `case` 末尾要 `break`，否則**貫穿（fall-through）**：執行完會接著做下一個 case，常是 bug。
- 想讓多個值走同一段，可故意不寫 `break` 讓它們堆疊：`case 'A': case 'B':`（A、B 都做下面那段）。
- `default` 處理「都不符」，可省略但建議保留。
- switch 比對的型別限 `byte/short/int/char`、`String`、`enum`（Java 8 範圍；非浮點數）。

3.4 4. break 與 continue

- `break`：立刻跳出**最內層**的 `switch` 或迴圈，後面的迴圈體與剩餘輪次都不做。
- `continue`：跳過本輪剩餘迴圈體，**直接進下一輪**（`for` 會先做更新、`while` 直接回去檢查條件）。

```
for (int i = 1; i <= 10; i++) {
    if (i == 5) continue;      // i=5 這輪跳過 println，但迴圈繼續
    if (i == 8) break;         // i=8 直接結束整個迴圈
    System.out.print(i + " ");
}                              // 輸出：1 2 3 4 6 7
```

3.5 5. 邏輯運算子：把多個條件組起來

把多個 `boolean` 條件組合：`&&`（且）、`||`（或）、`!`（非）。下表為三者的真值表（`T`=true、`F`=false）。

邏輯運算子真值表（T=true F=false）

a && b（且）

a	b	a&&b
T	T	T
T	F	F
F	T	F
F	F	F

a || b（或）

a	b	a||b
T	T	T
T	F	T
F	T	T
F	F	F

!a（非）

a	!a
T	F
F	T

短路：&& 左為 F 就不看右（結果必 F）；|| 左為 T 就不看右（結果必 T）。

不短路 & |：一定算兩邊；^（XOR）：兩邊不同才 T。

優先序：! → 關係/相等 → & → ^ → | → && → ||；混用時加括號最保險。

```
if (score >= 60 && score <= 100)        // 兩條件都成立才及格且合理
    System.out.println("Pass");
if (day == 0 || day == 6)               // 週日或週六
    System.out.println("Weekend");
if (!found)                             // found 為 false 時
    System.out.println("Not found");
```

短路（short-circuit）：`&&` 左邊為假時，整體必為假，右邊**不算**；`||` 左邊為真時，整體必為真，右邊**不算**。常用來保護：

```
if (s != null && s.length() > 0)    // 左假則右不算，避開 NullPointerException
    System.out.println(s);
```

`&`、`|`（不短路）會**強制算兩邊**（少用、通常用在需要兩邊副作用時）；`^`（XOR、互斥或）在兩邊布林值**不同**時為真。

3.6 6. 巢狀迴圈：迴圈中的迴圈

把一個迴圈放進另一個迴圈，常用於表格、二維輸出。外迴圈跑一輪，內迴圈整個跑完。印 9×9 乘法表片段：

```
for (int i = 1; i <= 9; i++) {          // 外：第幾列
    for (int j = 1; j <= 9; j++) {      // 內：每列 9 欄
        System.out.printf("%d*%d=%-3d", i, j, i * j);
    }
    System.out.println();               // 每列結束換行
}
```

內層敘述總共執行「外圈數 × 內圈數」次（此例 9×9=81 次）。

§ 4 四、語法與 API 速查

```
for (初始化; 條件; 更新) { 迴圈體 }       // 初始化做一次；每輪：檢查→體→更新
do { 迴圈體 } while (條件);                 // 至少執行一次，結尾分號別漏
switch (值) {                              // 值限 int/char/String/enum…
    case A: …; break;                      // 漏 break 會貫穿
    case B: case C: …; break;              // 多值共用一段
    default: …;                            // 都不符
}
break;      // 跳出最內層迴圈或 switch
continue;   // 跳過本輪剩餘、進下一輪
```

- **邏輯運算子**：`&&`（短路且）、`||`（短路或）、`!`（非）、`&` `|`（不短路）、`^`（XOR）
- **優先序**：`!`（單元）> 關係（`< <= > >=`）> 相等（`== !=`）> `&` > `^` > `|` > `&&` > `||` > `?:` > 指派；混用時加括號最保險
- **範圍 1..n 求和**：`for (int i = 1; i <= n; i++) sum += i;`
- **倒數**：`for (int i = n; i >= 1; i--) …`
- **每隔 2**：`for (int i = 0; i <= 20; i += 2) …`（偶數）

§ 5 五、常見錯誤

- **for 三格分號錯**：`for (i = 1, i <= 5, i++)` 用逗號錯，三格要用**分號** `;` 分隔。
- **switch 漏 break**：忘了 `break` → **貫穿**到下個 case，多印出不該印的；除非刻意共用。
- **do-while 漏結尾分號**：`} while (條件)` 後面一定要 `;`。
- **`&&` 寫成 `&`**：`&` 不短路，會照算右邊；若右邊有 `s.length()` 而 `s` 為 null 就爆 `NullPointerException`。短路保護一定用 `&&`。
- **`||` 寫成 `|`**：同理，`|` 不短路、失去短路保護。
- **off-by-one**：`for (int i = 0; i <= n; i++)` 跑 n+1 次、`i < n` 跑 n 次；邊界要算清楚。
- **for 計數器作用域**：`for (int i...)` 的 `i` 出迴圈即消失，迴圈外要用就得在外面宣告。
- **巢狀內外迴圈變數同名**：內外都叫 `i` 會編譯錯（內層遮蔽）；用不同名 `i`、`j`。
- **switch 比 double**：`switch` 不接受浮點數與 `boolean`；只能 `byte/short/int/char/String/enum`。

§ 6 六、練習題

例題 1：for 求 1..100 的和

用 `for` 迴圈計算 1 + 2 + … + 100，印出 `Sum = 總和`。

> 1. `int sum = 0;` 2. `for (int i = 1; i <= 100; i++) sum += i;` 3. 迴圈外印 sum

```
public class ForSum {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i <= 100; i++) {
            sum += i;
        }
        System.out.println("Sum = " + sum);    // 5050
    }
}
```

> 易錯：三格用逗號分隔（應分號）；`i <= 100` 寫成 `i < 100` 少加 100。

例題 2：switch 成績轉等第

給一個分數 `score`（用固定值 85），用 `switch` 依 `score / 10` 印出等第：90 以上 `A`、80–89 `B`、70–79 `C`、其餘 `F`。

> 1. `int level = score / 10;`（85→8） 2. `switch (level)`：`case 10:` 與 `case 9:` 共用印 A；`case 8:` 印 B… 3. 用 `default` 處理其餘印 F

```
public class GradeSwitch {
    public static void main(String[] args) {
        int score = 85;
        int level = score / 10;       // 85 / 10 = 8 (整數除法)
        switch (level) {
            case 10:
            case 9:
                System.out.println("A"); break;
            case 8:
                System.out.println("B"); break;   // 命中
            case 7:
                System.out.println("C"); break;
            default:
                System.out.println("F");
        }
    }
}
```

> 易錯：漏 `break` 造成貫穿；忘了 90 與 100 都該是 A（`case 10` 與 `case 9` 共用）。

例題 3：巢狀 for 印 9×9 乘法表

用**巢狀** `for` 印出完整 9×9 乘法表，每格 `i*j=積`，每列 9 格、列末換行。

> 1. 外迴圈 `i` 1→9（第幾列） 2. 內迴圈 `j` 1→9：`printf("%d*%d=%-3d", i, j, i*j)` 3. 內迴圈結束後 `println()` 換行

```
public class MultTable {
    public static void main(String[] args) {
        for (int i = 1; i <= 9; i++) {
            for (int j = 1; j <= 9; j++) {
                System.out.printf("%d*%d=%-3d", i, j, i * j);
            }
            System.out.println();          // 每列換行
        }
    }
}
```

> 易錯：內外迴圈變數同名 `i`；`println` 放在內迴圈裡導致每格換行；忘了列末 `println()`。

§ 7 七、自我檢核

· [ ] 能把一個計數器 `while` 改寫成等價的 `for`，說得出三格各做什麼、各執行幾次。

- [ ] 知道 `do-while` 與 `while` 差在「至少執行一次」，會寫並記得結尾分號。
- [ ] 會用 `switch` 做多選，知道漏 `break` 會貫穿、何時刻意共用 case。
- [ ] 分得清 `break`（跳出迴圈）與 `continue`（跳過本輪）。
- [ ] 會用 `&&`、`||`、`!` 組合條件，能默寫三者真值表。
- [ ] 知道 `&&`、`||` 會短路，並會用 `s != null && …` 避開 NullPointerException。
- [ ] 會寫巢狀迴圈，算得出內層敘述總執行次數。