基礎程式設計（Java）　講義

方法：更深入一層

Deitel & Deitel《Java How to Program: Early Objects》11e, Ch.6

§ 1　一、名詞速查表

中文	English	一句話定義	科目	章節
方法	method	一段有名字、可重複呼叫的程式碼，完成單一工作	Java	CH6
方法宣告	method declaration	定義方法：修飾子、回傳型別、名稱、參數列、方法體	Java	CH6
回傳型別	return type	方法回傳值的型別；不回傳值用 `void`	Java	CH6
參數	parameter	方法宣告括號內的變數，接收呼叫端傳入的值	Java	CH6
引數	argument	呼叫方法時實際傳入的值	Java	CH6
static 方法	static method	屬於類別本身、不需建物件就能呼叫（`類名.方法`）	Java	CH6
呼叫堆疊	call stack	記錄方法呼叫順序的堆疊，後呼叫的先返回（LIFO）	Java	CH6
活動記錄	activation record	堆疊框；存某次方法呼叫的參數與區域變數	Java	CH6
引數提升	argument promotion	引數型別自動轉成參數要求的較大型別（如 `int`→`double`）	Java	CH6
方法多載	method overloading	同名方法、靠參數列（數量/型別/順序）不同來區分	Java	CH6
值傳遞	pass-by-value	Java 一律傳「值的拷貝」；物件傳的是參照值的拷貝	Java	CH6
範疇	scope	變數可被存取的程式區域（區域變數限於宣告的區塊）	Java	CH6
套件	package	Java API 把相關類別分組的單位（如 `java.util`）	Java	CH6
匯入宣告	import declaration	`import` 告訴編譯器到哪個套件找類別	Java	CH6
安全亂數	SecureRandom	`java.security` 的類別，產生不易預測的亂數	Java	CH6
列舉型別	enum type	用 `enum` 定義一組具名常數的型別	Java	CH6

§ 2 二、核心概念

方法把一段工作包成有名字的單位，呼叫一次就執行一次。好處是**分而治之**（把大問題拆成小方法）、**避免重複**（寫一次、用多次）、**易維護**（改一處即可）。

> 本章把 CH3 學過的方法再深入：怎麼宣告與呼叫、`static` 方法（不需物件）、方法呼叫背後的**呼叫堆疊**怎麼 push／pop、引數型別怎麼自動**提升**、Java API 的**套件**與**匯入**、用 `SecureRandom` 產生亂數（擲骰機率遊戲）、用 `enum` 定義具名常數、同名方法怎麼**多載**、傳參數時 Java 一律**值傳遞**（連物件也是傳參照的拷貝）、變數的**範疇**。

每次呼叫方法，Java 在**呼叫堆疊**頂端放一個**活動記錄**（存這次呼叫的參數與區域變數）；方法返回時這個記錄被 pop 掉，控制權回到呼叫點。最後呼叫的方法最先返回（後進先出，LIFO）。

> **遞迴預告**：「方法呼叫自己」稱為**遞迴（recursion）**，課本《Java How to Program》11e 把它放在**第 18 章**詳述，第 6 章 6.1 節僅點到。本章不展開遞迴，等第 18 章再學。

§ 3 三、主要內容

3.1 1. 方法宣告與呼叫

```
public static int square(int x) {    // 修飾子 回傳型別 名稱(參數列)
    return x * x;                    // 方法體：算出並回傳
}
// 呼叫：把引數 5 傳給參數 x，square 算出 25 回來
int result = square(5);              // result = 25
```

- **回傳型別**：方法算完要交回的值的型別；不交回值寫 `void`。
- **參數 vs 引數**：宣告時括號裡的是**參數**（`x`）；呼叫時實際給的值是**引數**（`5`）。
- `return` 把值交回呼叫點，並立刻結束方法。

3.2 2. static 方法與欄位

`static`（靜態）成員屬於**類別本身**，不需建立物件就能用，呼叫格式是 `類名.方法(...)`。

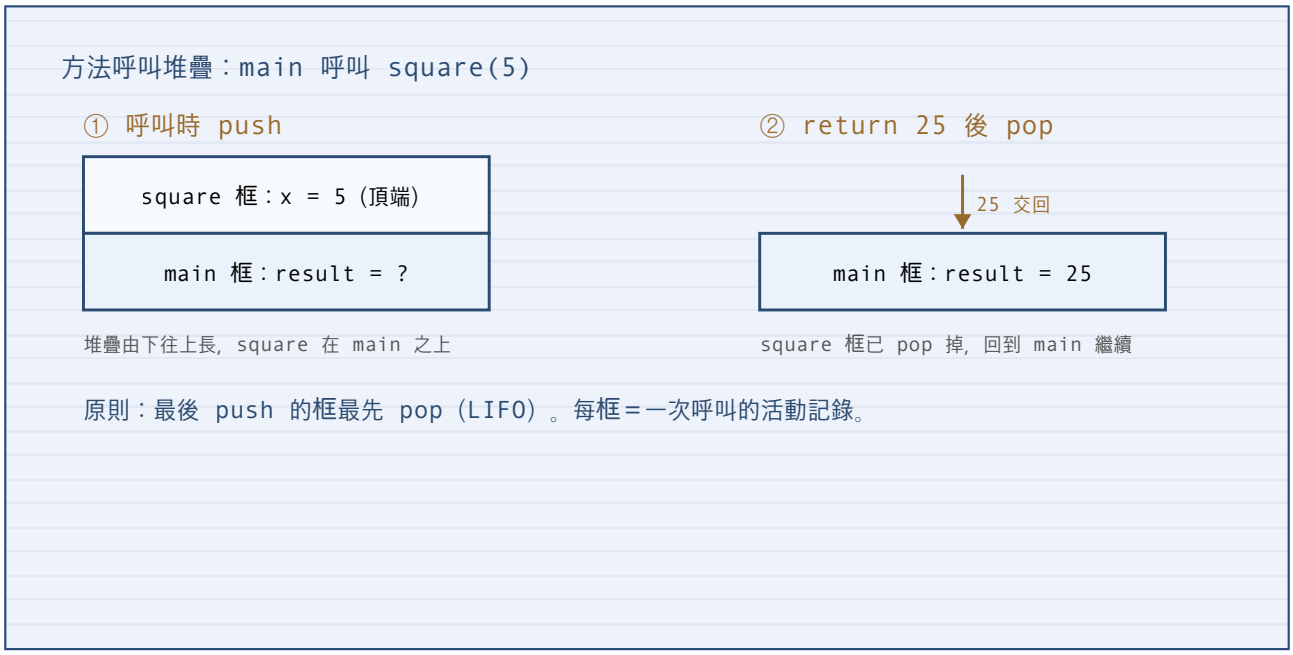
```
double r = Math.sqrt(16.0);    // Math 全是 static 方法，直接 類名.方法
int big = Math.max(3, 7);      // 不需 new Math()
```

- 同類別內呼叫自己的 `static` 方法可省略類名。
- `static` 欄位（類別變數）由**全部物件共享**一份；實例欄位則每個物件各一份。

- `main` 本身就是 `static`，所以 `main` 裡只能直接呼叫 `static` 方法，或先 `new` 一個物件再透過物件呼叫實例方法。

3.3 3. 方法如何運作：呼叫堆疊

呼叫方法時，Java 把一個**活動記錄**（含參數、區域變數、返回位址）push 到**呼叫堆疊**頂端；方法 `return` 後該記錄被 pop，控制權與回傳值交回呼叫點。最後呼叫的最先返回（**後進先出，LIFO**）。



若呼叫層疊得太深、堆疊框超過記憶體上限，會丟 `StackOverflowError`（最常見成因是無止境的呼叫鏈，第 18 章遞迴會再遇到）。

3.4 4. 引數提升

引數型別比參數要求的小時，Java 會自動**提升**成較大型別（不遺失資訊的方向）。例如把 `int` 傳給要 `double` 的參數，`int` 自動變 `double`。

```
System.out.println(Math.sqrt(25));   // 25 是 int，自動提升成 25.0 → 印 5.0
```

- 安全方向（自動）：`byte → short → int → long → float → double`。
- 反方向（大轉小，如 `double → int`）會遺失資訊，**必須明確強制轉型** `(int)`。

3.5 5. Math 類別方法

`Math` 提供常用數學運算，全是 `static`，直接 `Math.方法(...)`。

```
Math.abs(-7)         // 7       絕對值
Math.max(3, 9)       // 9       較大值（min 取較小）
Math.pow(2, 10)      // 1024.0  次方（回傳 double）
Math.sqrt(81.0)      // 9.0     平方根
Math.round(3.6)      // 4       四捨五入
Math.PI              // 3.14159… 常數（static 欄位）
```

3.6 6. Java API 套件與匯入

Java 把上千個預先寫好的類別依用途分組成**套件**（package），合稱 **Java API**。用某套件的類別前，先用 `import` 宣告告訴編譯器去哪裡找。

```
import java.util.Scanner;            // 匯入單一類別
import java.security.SecureRandom;   // 不同套件、不同類別
```

- `java.lang`（含 `Math`、`String`、`System`）**自動匯入**，不必寫 `import`。
- 常見套件：`java.util`（`Scanner`、`Arrays`）、`java.security`（`SecureRandom`）、`java.io`（輸入輸出）。
- 也可寫 `import java.util.*;` 匯入整個套件，但逐一匯入較清楚。

3.7 7. 亂數產生：SecureRandom

`SecureRandom`（`java.security`）產生**不易預測**的亂數，適合遊戲、安全用途。`nextInt(n)` 回傳 `0` 到 `n-1` 的整數。

```
import java.security.SecureRandom;
SecureRandom rand = new SecureRandom();
int face = 1 + rand.nextInt(6);   // 擲一顆骰：1~6（先得 0~5，再 +1 平移）
```

- **縮放**：`nextInt(6)` 把範圍縮成 6 個值（0~5）；**平移**：`+ 1` 把起點挪到 1。通式 `起點 + rand.nextInt(範圍寬度)`。
- `Math.random()` 也能產生亂數：回傳 `[0.0, 1.0)` 的 `double`，`(int)(Math.random() * 6)` 得 0~5。課本主推 `SecureRandom`（品質較好），`Math.random()` 為舊式作法。

3.8 8. 列舉型別 enum

`enum` 定義一組具名常數的型別，比用整數代表狀態更安全、可讀。常數依慣例全大寫。

```
private enum Status { CONTINUE, WON, LOST }   // 三個具名常數
Status gameStatus = Status.CONTINUE;          // 用「型別.常數」取值
if (gameStatus == Status.WON) { /* ... */ }
```

- 課本用擲骰子的「機率遊戲（craps）」示範：用 `Status` 取代 `1/2/3` 這種「魔術數字」，程式更易讀。
- `enum` 常數是 `static final`，比較用 `==` 即可。

3.9 9. 方法多載

同一類別中可有**多個同名方法**，只要**參數列不同**（數量、型別或順序不同）即可；編譯器依引數挑對應版本。回傳型別不同**不能**單獨用來多載。

```
public static int square(int x)       { return x * x; }       // 版本 A
public static double square(double x) { return x * x; }       // 版本 B
square(5);     // 引數是 int → 呼叫 A，得 25
square(2.5);   // 引數是 double → 呼叫 B，得 6.25
```

3.10 10. 引數傳遞：Java 一律值傳遞

Java 呼叫方法時**一律傳值的拷貝**（pass-by-value）。基本型別傳的是值的拷貝，方法內怎麼改都動不到呼叫端的變數。**物件**傳的是**參照值的拷貝**：拷貝的參照仍指向同一個物件，所以方法內可透過它改物件內容；但若把參數重新指向新物件，呼叫端的參照不受影響。

```
public static void addOne(int n) { n = n + 1; }   // 改的是拷貝
int a = 5;
addOne(a);
System.out.println(a);   // 仍是 5（基本型別：動不到呼叫端）
```

- 陣列是物件，傳進方法後改元素**會**反映到呼叫端（同一個陣列物件）。
- 「Java 是傳參照」是常見誤解；正確說法：**一律值傳遞**，物件傳的是「參照的值」。

3.11 11. 區域變數範疇

範疇＝變數能被存取的程式區域。在區塊（`{ }`）內宣告的區域變數，只在該區塊內有效，離開就消失。參數的範疇是整個方法體。

```java
public static void demo() {
    int x = 10;                  // x 的範疇：整個 demo
    if (x > 5) {
        int y = 20;              // y 的範疇：只有這個 if 區塊
        System.out.println(y);   // OK
    }
    // System.out.println(y);    // 錯：y 在這裡已超出範疇
}
```

- 不同方法可有同名區域變數，互不干擾（各自的範疇）。
- 區域變數**沒有預設值**，使用前必須先賦值（實例欄位才有預設值）。

§ 4 四、語法與 API 速查

```
修飾子 回傳型別 方法名(參數列) { 方法體; return 值; }    // 方法宣告
類名.方法(引數)                                          // static 方法呼叫
return 值;                                               // 交回值並結束方法
```

- **方法宣告**：`public static int add(int a, int b) { return a + b; }`
- **多載**：同名、參數列（數量/型別/順序）不同；回傳型別不可單獨區分
- **引數提升**：小型別自動轉大型別（`int`→`double`）；大轉小要 `(int)` 強制轉型
- **Math**：`abs / max / min / pow / sqrt / round / PI`（全 `static`）
- **套件／匯入**：`import java.util.Scanner;`；`java.lang` 自動匯入
- **亂數**：`new SecureRandom().nextInt(n)` 得 `0~n-1`；`起點 + nextInt(寬度)`
- **enum**：`enum Status { CONTINUE, WON, LOST }`；取值 `Status.WON`、比較用 `==`
- **值傳遞**：基本型別動不到呼叫端；物件傳參照拷貝，可改內容、不可換物件
- **範疇**：區域變數限宣告的區塊，沒有預設值、用前須先賦值
- **遞迴**：方法呼叫自己，課本放在**第 18 章**，本章不涵蓋

§ 5 五、常見錯誤

- **以為改參數能改到呼叫端**：基本型別值傳遞，方法內 `n = n + 1` 動不到呼叫端的變數。
- **`void` 方法寫 `return 值`**：`void` 不回傳值；要回傳就改回傳型別，不要寫 `void`。
- **多載只改回傳型別**：`int f(int)` 與 `double f(int)` 無法多載（參數列相同），編譯錯誤。
- **在 `static` 方法裡用實例成員**：`main` 是 `static`，不能直接呼叫非 `static` 方法或用實例欄位，要先 `new` 物件。
- **區域變數未初始化就用**：區域變數沒有預設值（與實例欄位不同），編譯器會報「可能尚未初始化」。

- **變數超出範疇還用**：`if` 區塊內宣告的變數，出了大括號就不存在。
- **忘了 `import`**：用 `Scanner`、`SecureRandom` 前要 `import`（`java.lang` 例外、自動匯入）。
- **亂數範圍弄錯**：`nextInt(6)` 是 `0~5` 不是 `1~6`；要 `1~6` 得 `1 + nextInt(6)`。
- **enum 常數當字串／整數比**：`enum` 是型別，取值用 `Status.WON`，不是 `"WON"` 或數字。

§ 6 六、練習題

例題 1：寫方法求兩數最大

寫一個 `static` 方法 `max(int a, int b)` 回傳較大值，在 `main` 用固定值 `(17, 42)` 測試並印出。

> 1. `public static int max(int a, int b)` 2. 體內 `if (a > b) return a; else return b;` 3. `main` 印 `max(17, 42)`

```
```

```java
public class MaxOfTwo {
    public static int max(int a, int b) {
        if (a > b) return a;
        else return b;
    }
    public static void main(String[] args) {
        System.out.println("max = " + max(17, 42));   // 42
    }
}
```

易錯：`void` 卻想回傳值；忘了 `return` 其中一條路徑。

例題 2：SecureRandom 擲骰

用 `SecureRandom` 寫方法 `roll()` 回傳一顆骰子的點數（1~6）。在 `main` 擲 5 次，逐次印出點數。需 `import java.security.SecureRandom;`。

> 1. 類別內建一個 `static SecureRandom rand = new SecureRandom();` 2. `static int roll() { return 1 + rand.nextInt(6); }`（縮放＋平移） 3. `main` 用迴圈擲 5 次，逐次印 `roll()`

```
import java.security.SecureRandom;

public class RollDice {
    private static SecureRandom rand = new SecureRandom();
    public static int roll() {
        return 1 + rand.nextInt(6);       // nextInt(6) 得 0~5，+1 平移成 1~6
    }
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            System.out.print(roll() + " ");   // 每次都是 1~6（值會變）
        }
        System.out.println();
    }
}
```

> 易錯：忘了 `import`；寫成 `nextInt(6)` 得 0~5 沒 +1；以為每次輸出相同（亂數每次不同）。

例題 3：方法多載

寫兩個同名方法 `square`：一個收 `int` 回 `int`，一個收 `double` 回 `double`。在 `main` 各呼叫一次（`square(6)`、`square(2.5)`）。

> 1. `static int square(int x) { return x * x; }` 2. `static double square(double x) { return x * x; }` 3. `main` 分別印兩個結果（編譯器依引數型別選版本）

```
public class Overload {
    public static int square(int x) { return x * x; }
    public static double square(double x) { return x * x; }
    public static void main(String[] args) {
        System.out.println(square(6));     // 36 (int 版)
        System.out.println(square(2.5));   // 6.25 (double 版)
    }
}
```

> 易錯：只改回傳型別（參數列相同）不算多載；以為 `square(6)` 會跑到 double 版。

§ 7 七、自我檢核

- [ ] 能寫出方法宣告（修飾子／回傳型別／參數列／方法體），分得清參數與引數。
- [ ] 知道 `static` 方法不需建物件、用 `類名.方法` 呼叫，`main` 裡的限制。

- [ ] 能說出方法呼叫時呼叫堆疊怎麼 push／pop（活動記錄、LIFO）。
- [ ] 知道引數提升的安全方向，大轉小要強制轉型。
- [ ] 會用 `Math` 常用方法（`abs/max/pow/sqrt`）。
- [ ] 知道 Java API 套件與 `import`，曉得 `java.lang` 自動匯入。
- [ ] 會用 `SecureRandom.nextInt(n)` 產生亂數，懂縮放與平移（`起點 + nextInt(寬度)`）。
- [ ] 知道 `enum` 定義具名常數，取值用 `型別.常數`、比較用 `==`。
- [ ] 會寫方法多載，知道回傳型別不能單獨區分。
- [ ] 能說清 Java 一律值傳遞：基本型別動不到呼叫端、物件傳參照拷貝可改內容。
- [ ] 分得清區域變數範疇，知道區域變數沒有預設值。