CHAPTER 3

學習目標

- 認識資料型態與變數
- 學習運算子基本使用
- 瞭解型態轉換細節
- 運用基本流程語法
- OOP 概論

資料型態

- Java可區分為兩大型態系統：
  - 基本型態（Primitive type）
  - 類別型態（Class type）

資料型態

- 整數
  - **byte**（長度就是一個位元組）
  - **short**（佔2個位元組）
  - **int**（佔4個位元組）
  - **long**（佔8個位元組）
- 浮點數
  - **float**（佔4個位元組）
  - **double**（佔8個位元組）

資料型態

- 字元
  - **char**型態用來儲存 'A' 、 'B' 、 '林' 等字元符號
  - 在JDK7中，Java的字元採Unicode 6.0編碼
  - JVM實作採UTF-16 Big Endian，所以每個字元型態佔兩個位元組，中文字元與英文字元在Java中同樣都是用兩個位元組儲存
- 布林
  - **boolean**型態可表示**true**與**false**

資料型態

- 測試各種型態可儲存的數值範圍

```
// byte、short、int、long 範圍
System.out.printf("%d ~ %d%n",
        Byte.MIN_VALUE, Byte.MAX_VALUE);
System.out.printf("%d ~ %d%n",
        Short.MIN_VALUE, Short.MAX_VALUE);
System.out.printf("%d ~ %d%n",
        Integer.MIN_VALUE, Integer.MAX_VALUE);
System.out.printf("%d ~ %d%n",
        Long.MIN_VALUE, Long.MAX_VALUE);
// float、double 精度範圍
System.out.printf("%d ~ %d%n",
        Float.MIN_EXPONENT, Float.MAX_EXPONENT);
System.out.printf("%d ~ %d%n",
        Double.MIN_EXPONENT, Double.MAX_EXPONENT);
// char 可表示的 Unicode 範圍
System.out.printf("%h ~ %h%n",
        Character.MIN_VALUE, Character.MAX_VALUE);
// boolean 的兩個值
System.out.printf("%b ~ %b%n",
        Boolean.TRUE, Boolean.FALSE);
```

註解

- 單行註解 //
- 多行註冊

```
/*
   Java 程式
   -version 1.0
*/
```

註解

- 以下使用多行註解方式是不對的
- 不可巢狀式

```
/*   註解文字 1……bla…bla
     /*
          註解文字 2……bla…bla
     */
*/
```

- System.out.printf()是格式化輸出方法

符號	說明
%%	因為%符號已經被用來作為控制符號前置，所以規定使用%%才能在字串中表示%。
%d	以 10 進位整數格式輸出，可用於 byte、short、int、long、Byte、Short、Integer、Long、BigInteger。
%f	以 10 進位浮點數格式輸出，可用於 float、double、Float、Double 或 BigDecimal。
%e, %E	以科學記號浮點數格式輸出，提供的數必須是 float、double、Float、Double 或 BigDecimal。%e 表示輸出格式遇到字母以小寫表示，如 2.13e+12，%E 表示遇到字母以大寫表示。
%o	以 8 進位整數格式輸出，可用於 byte、short、int、long、Byte、Short、Integer、Long、或 BigInteger。
%x, %X	以 16 進位整數格式輸出，可用於 byte、short、int、long、Byte、Short、Integer、Long、或 BigInteger。%x 表示字母輸出以小寫表示，%X 則以大寫表示。

%s, %S	字串格式符號。
%c, %C	以字元符號輸出，提供的數必須是 byte、short、char、Byte、Short、Character 或 Integer。%c 表示字母輸出以小寫表示，%C 則以大寫表示。
%b, %B	輸出 boolean 值，%b 表示輸出結果會是 true 或 false，%B 表示輸出結果會是 TRUE 或 FALSE。非 null 值輸出是 true 或 TRUE，null 值輸出是 false 或 FALSE。
%h, %H	使用 Integer.toHexString(arg.hashCode())來得到輸出結果，如果 arg 是 null，則輸出 null，也常用於想得到 16 進位格式輸出。
%n	輸出平台特定的換行符號，如果 Windows 下會置換為"\r\n"，如果是 Linux 下則會置換為'\n'，Mac OS 下會置換為'\r'。

```
System.out.printf("example:%.2f%n", 19.234);

System.out.printf("example:%6.2f%n", 19.234);
```

變數

- 程式執行時期，暫存資料用的記憶體 ...

```
System.out.println(10);
System.out.println(3.14);
System.out.println(10);
```

```
int number = 10;
double PI = 3.14;
System.out.println(number);
System.out.println(PI);
System.out.println(number);
```

變數

- 想要宣告何種型態的變數，就使用
  - byte、short、int、long、
  - float、double、
  - char、boolean等關鍵字來宣告
- 變數在命名時有一些規則，它不可以使用數字作為開頭，也不可以使用一些特殊字元，像是*、&、^、%之類的字元
- 變數名稱不可以與Java 的關鍵字（Keyword）同名，例如int、float、class等就不能作為變數名稱

變數

- 在Java領域中的變數命名慣例
  通常會以小寫字母開始，並在每個單字開始時第一個字母使用大寫，稱為駝峰式（Camel case）命名法

```
int ageOfStudent;
int ageOfTeacher;
```

變數

- 在Java中宣告一個區域變數，就會為變數配置記憶體空間，但不會給這塊空間預設值
- 不可以宣告區域變數後未指定任何值給它之前就使用變數

```
double score;
System.out.println(score);
```

variable score might not have been initialized

(Alt-Enter shows hints)

變數

- 如果在指定變數值之後，就不想再改變變數值，可以在宣告變數時加上**final**限定
- 如果後續撰寫程式時，自己或別人不經意想修改final變數，就會出現編譯錯誤

```
final double PI = 3.141596;
```

cannot assign a value to final variable PI

(Alt-Enter shows hints)

```
PI = 3.14;
```

變數

- 各種資料表達式（Literal constant）

```
int number1 = 12;     // 10 進位表示
int number2 = 0xC;    // 16 進位表示，以 0x 開頭
int number3 = 014; // 8 進位表示，以 0 開頭

double number1 = 0.00123;
double number2 = 1.23e-3;

char size = 'S';
char lastName = '林';

char symbol = '\'';

boolean flag = true;
boolean condition = false;
```

變數

忽略符號	說明
\\	反斜線\。
\'	單引號'。
\"	雙引號"。
\uxxxx	以 16 進位數指定 Unicode 字元輸出，x 表示數字。
\xxx	以 8 進位數指定 Unicode 字元輸出，x 表示數字。
\b	倒退一個字元。
\f	換頁。
\n	換行。
\r	游標移至行首。
\t	跳格(按下 Tab 鍵的字元)。

```
System.out.println("\u0048\u0065\u006C\u006C\u006F");
```

變數

- Java 特殊表示法

```
int number1 = 1234_5678;
double number2 = 3.141_592_653;

int mask = 0b101010101010;   // 用二進位表示 10 進位整數 2730

int mask = 0b1010_1010_1010;   // 用二進位表示 10 進位整數 2730
```

算術運算子

- 以下程式碼片段會在文字模式下顯示7

```
System.out.println(1 + 2 * 3);
```

- 以下程式碼會顯示的是6：

```
System.out.println(2 + 2 + 8 / 4);
```

- 以下程式碼顯示的是3：

```
System.out.println((2 + 2 + 8) / 4);
```

算術運算子

- 運算子計算的結果是除法後的餘數

n = 10 / 2　　(n is 5)

n = 10 % 2　　(n is 0)

+　-　*　/　%(餘除)

運算子

- 遞增、遞減運算

```
int i = 0;
i = i + 1;
System.out.println(i);
i = i - 1;
System.out.println(i);
```

```
int i = 0;
i++;
System.out.println(i);
i--;
System.out.println(i);
```

```
int i = 0;
System.out.println(++i);
System.out.println(--i);
```

運算子

```
int i = 0;
int number = 0;
number = ++i;    // 結果相當於 i = i + 1; number = i;
System.out.println(number);
number = --i;    // 結果相當於 i = i - 1; number = i;
System.out.println(number);

int i = 0;
int number = 0;
number = i++;    // 相當於 number = i; i = i + 1;
System.out.println(number);
number = i--;    // 相當於 number = i; i = i - 1;
System.out.println(number);
```

複合指定運算子

指定運算子	範例	結果
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b
<<=	a <<= b	a = a << b
>>=	a >>= b	a = a >> b

運算子

- 條件運算子（也稱為三元運算子）

```
System.out.printf("該生是否及格?%c%n", score >= 60 ? '是' : '否');

System.out.printf("是否為偶數?%c%n", (number % 2 == 0) ? '是' : '否');


if(number % 2 == 0) {
    System.out.println("是否為偶數?是");
}
else {
    System.out.println("是否為偶數?否");
}
```

型態轉換

- 這個片段編譯時沒有問題...

```
double PI = 3.14;
```

- 如果你寫了個程式片段 ...

```
float PI = 3.14;
```

possible loss of precision
  required: float
  found: double

(Alt-Enter shows hints)

型態轉換

- 在程式中寫下一個浮點數時，編譯器預設會使用**double**型態
- 編譯器會告知想將double長度的資料指定給float型態變數，會因為8個位元組資料要放到4個位元組空間，而遺失4個位元組的資料

型態轉換

- 兩種方式可以避免這個錯誤...

```
float PI = 3.14F;

float PI = (float) 3.14;
```

- 使用(float)語法告訴編譯器，你就是要將double型態的3.14指定給float變數，別再囉嗦了
- 後果自負...遺失精度而發生程式錯誤了，那絕不是編譯器的問題

型態轉換

- 這沒有問題 ...

```
int number = 10;
```

- 但...

integer number too large: 2147483648

(Alt-Enter shows hints)

```
int number =2147483648;
```

integer number too large: 2147483648

(Alt-Enter shows hints)

```
long number =2147483648;
```

型態轉換

- 程式中寫下一個整數時，預設是使用不超過int型態長度
- 2147483648超出了int型態的長度
- 直接告訴編譯器，用long來配置整數的長度，也就是在數字後加上個**L**

```
long number = 2147483648L;
```

型態轉換

- 程式中寫下一個整數時，預設是使用不超過int型態長度

```
byte number = 10;
```

- 不過這樣不行：

```
byte number = 128;
```

型態轉換

- 如果運算元都是不大於**int**的整數，則自動全部提昇為**int**型態進行運算

```
short a = 1;
short b = 2;
short c = a + b;
```

possible loss of precision
 required: short
 found: int

(Alt-Enter shows hints)

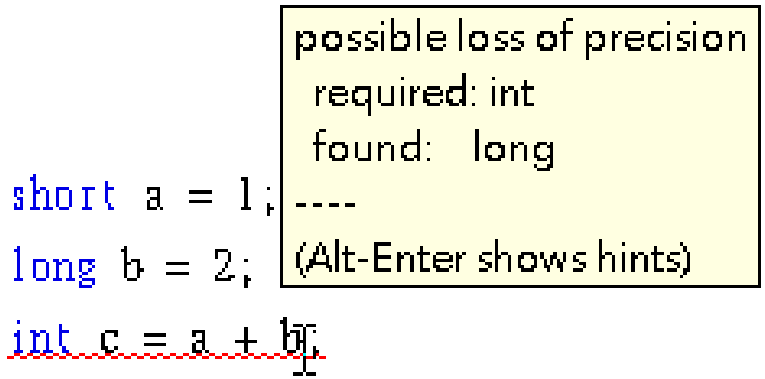
- int的運算結果要放到short，編譯器就又會囉嗦遺失精度的問題

型態轉換

- 你要告訴編譯器，就是要將int的運算結果丟到short，請它住嘴：

```
short a = 1;
short b = 2;
short c = (short) (a + b);
```

- 這次怎麼又遺失精度？



型態轉換

- b是long型態，於是a也被提至long空間中作運算，long的運算結果要放到int變數c，自然就會被編譯器囉嗦精度遺失了
- 如果這真的是你想要的，那就叫編譯器住嘴吧！

```
short a = 1;
long b = 2;
int c = (int) (a + b);
```

型態轉換

- 以下你覺得會顯示多少？

```
System.out.println(10 / 3);
```

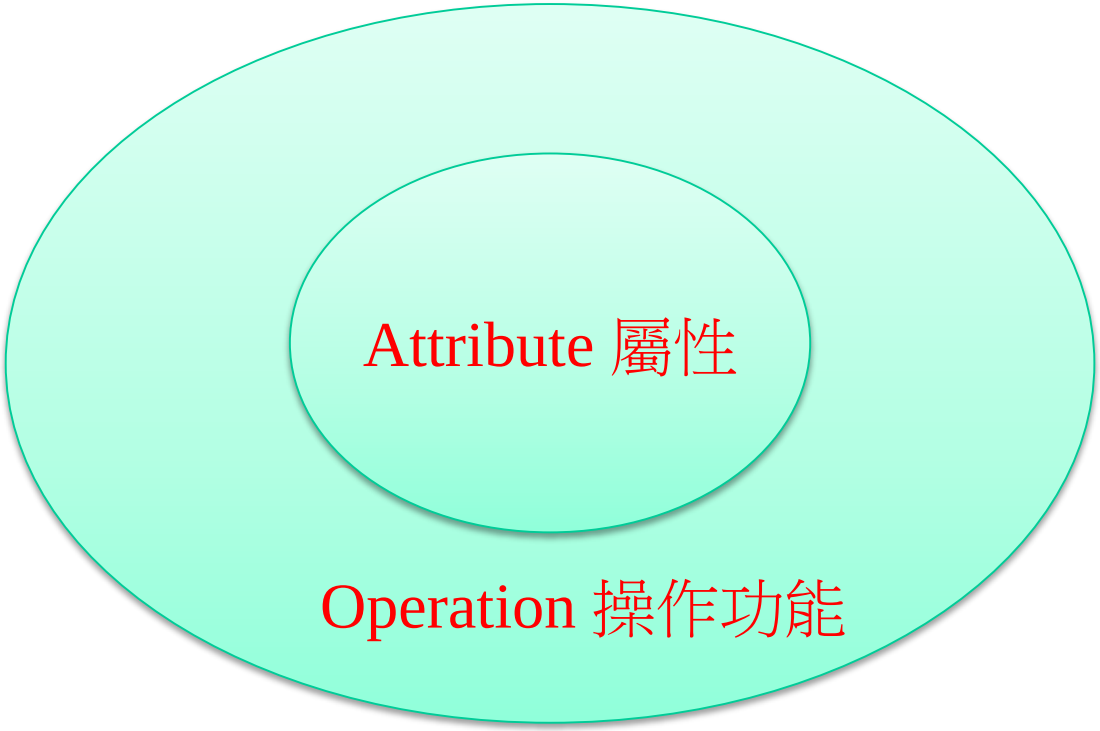
- 答案是3，而不是3.333333....，因為10與3會在int長度的空間中作運算

```
System.out.println(10.0 / 3);
```

Thinking in Object

- 「物件導向程式設計」簡單地說，就是一種抽象且擬人化的程式設計，與以往我們所熟悉的「程序式程式設計」大不相同。
- 程式設計師所設計的不再是一個個函式，而是一個個將程式抽象化且各自獨立的物件。
- 而每一個**物件**就是一個即是類別的**實例**

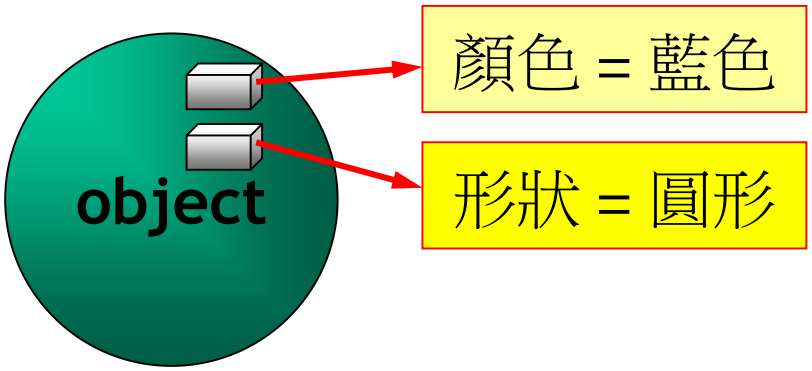
從物件分析與思考



- Objects have attributes (characteristics) such as size, name, shape, and so on.
- Objects have operations (they can do) such as setting a value, displaying a screen, or increasing speed.

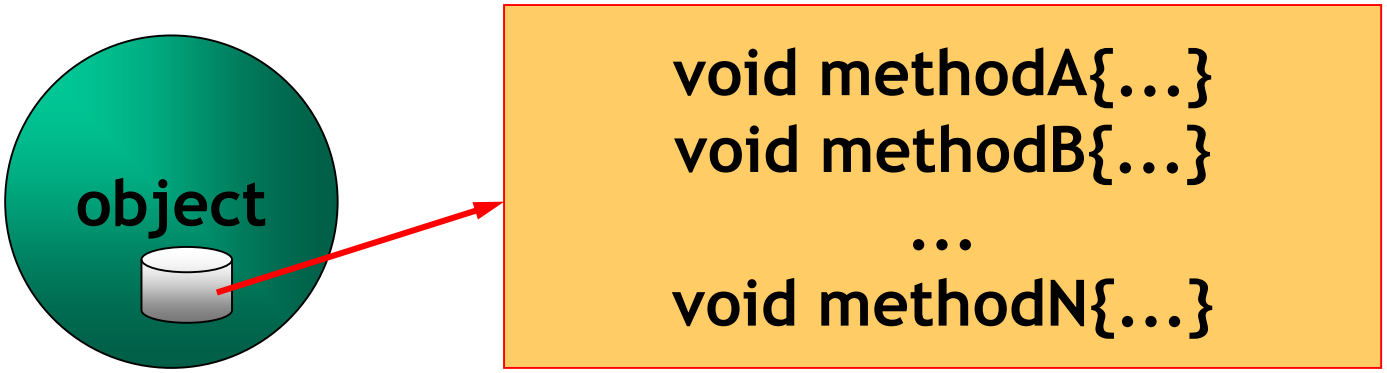
物件導向基礎論述

- 定義在物件內用來描述資料的稱作：
  - 屬性(Attribute)
  - 也是一種 ”變數” 或稱 “資料欄位”



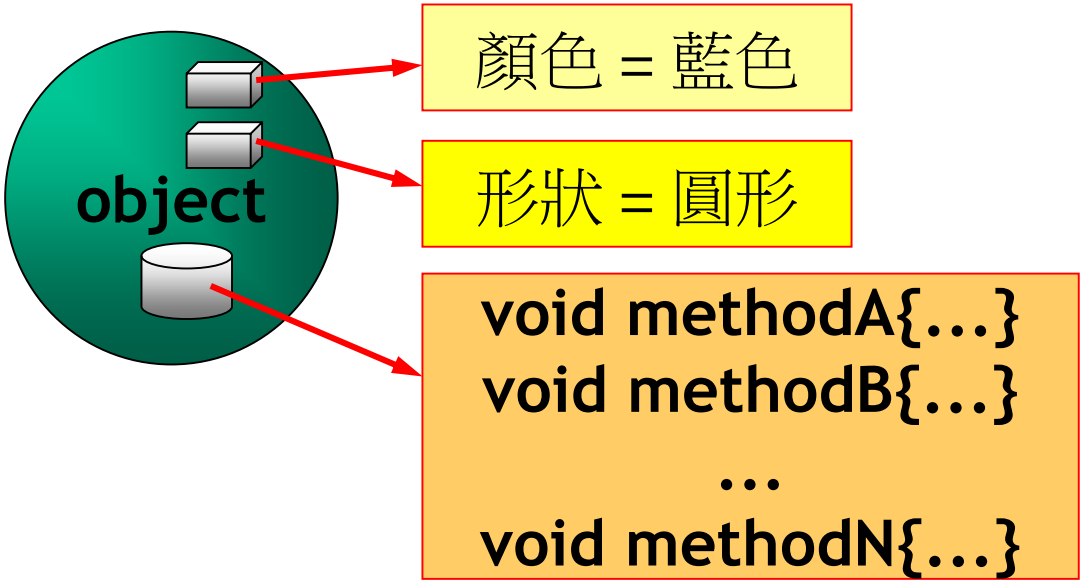
物件導向基礎論述

- 用來描述物件內部的行為，或一連串運算動作的集合稱作：
  - 方法(Method)



物件導向基礎論述

- 因此物件本身就是一群相關屬性(成員資料)與方法(成員函式)的組合。



物件導向基礎論述

- 物件導向程式(OOP)中，物件(Object)就是某個類別(Class)的實體(Instance)。
- 在Java語言中，產生一個物件實體可用new這個關鍵字。
- 在Java中，類別或物件可透過繼承的關係相互呼叫、引用與傳遞訊息，形成一種彼此繫結且有層次的結構。

物件導向基礎論述

- Java物件導向程式語言的 3 個特徵：
  - 封裝(Encapsulation)。
  - 繼承(Inheritance)。
  - 多型(Polymorphism)。

物件導向基礎論述

- 何謂類別？
  - 類別是物件的藍圖，也是物件實體的設計基礎。
  - 類別內部定義了資料欄位(屬性)與方法(操作功能)
  - 在 Java 程式中類別的宣告中引用了class 這個關鍵字
  - Java類別定義的語法：

```
存取修飾字 修飾語 class 類別名稱 {
   //定義 field 以及 method 敘述
}
```

物件導向基礎論述

- 類別 class：
  就是將相關的資料欄位(field) 與方法(method) 集合設計在同一個模組中。
- 主類別　(在類別中定義一個主方法)
- 一般類別 (主要用途作為建立物件的樣版)

物件導向基礎論述

- 一個專案的設計單元就是類別，也就是說專案是由1-N個各式的類別所組成。
- 類別來源
  - JDK Library (官方提供的類別 API)
  - 因專案需求設計者自行定義建構
  - 第三方商業元件

物件導向基礎論述

- 範例：

```
public class MyClass {
    private int money; // 屬性
    private int getMoney() { ... } // 方法
}
```

- 類別的存取修飾字若為 public，則類別名稱必須與實體類別檔名相同。
  - 以上述範例為例：MyClass.java

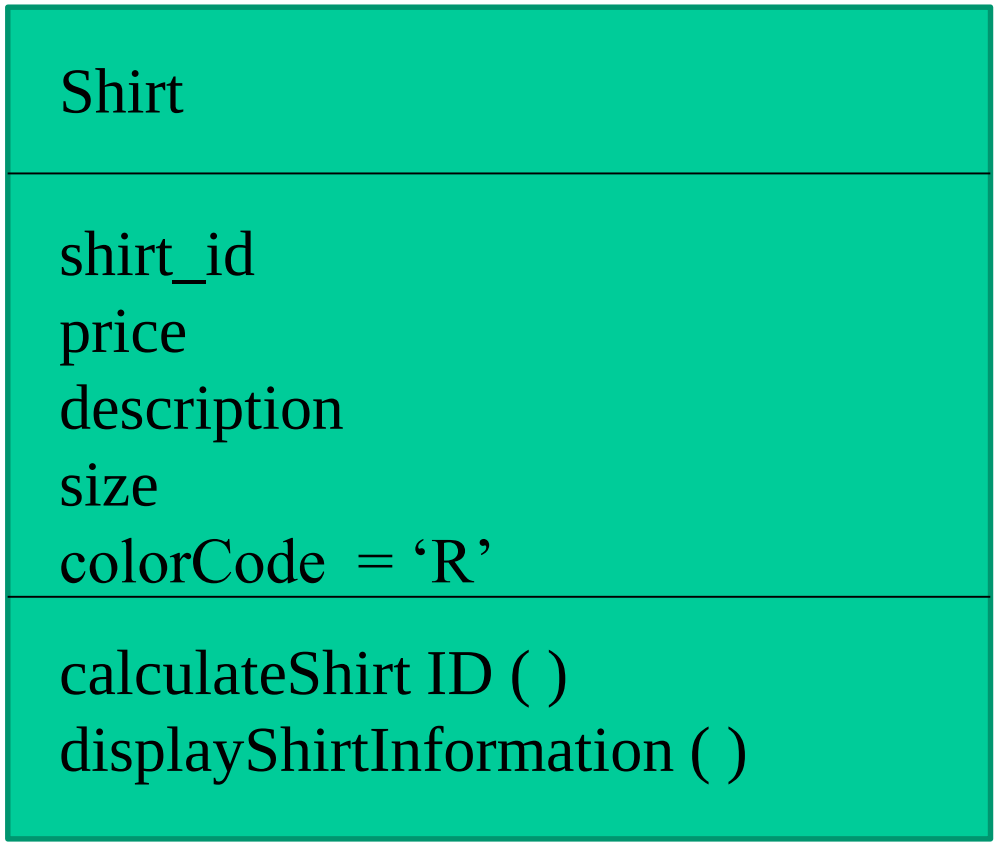
UML 類別設計圖

- UML: Universal Modeling Language

Class name

field

method



類別與物件

Shirt
shirt_id price description size colorCode = 'R'
calculateShirt ID ( ) displayShirtInformation ( )

先有類別，後有物件

Class

Object

資料欄位與物件屬性

- 類別宣告 field
  - 屬性是物件的資料，也是定義類別宣告的物件變數
  - 宣告物件變數(field)時，變數必須宣告在 class 內，例如：

```
public class MyClass {
    int price = 100;              // 物件變數(field)
    char  colorCode = 'G';
}
```

物件屬性

- 物件屬性-特性
  - 物件變數由物件各自獨立維護，彼此不受干擾，每一個物件都有自己的一份屬性。

物件方法

- 物件方法
  - 物件的方法是一種描述類別內部的行為，也是外部存取物件內部資料的方法。

```
public class MyClass {
    int momey = 100;    // 物件變數 field
    void aMethod() {    // 物件方法 (不加 static 修飾關鍵字)
        // block of code !
    }
}
```

物件方法

- 物件方法
  - 提供該物件具備的操作能力、運算處理等。

```
public class MyClass {
    private int money;
    public void getMoney() {
        方法實作區塊
    }
    public void setMoney(int money) {
        方法實作區塊
    }
}
```

建立物件實體

- 如何建立物件實體
  - 透過 new 關鍵字：
    1. 類別名稱　物件(變數)名稱；
       物件(變數)名稱　=　**new**　類別名稱()；
    2. 類別名稱　物件(變數)名稱　=　**new**　類別名稱()；

```
MyClass obj = new MyClass()
```

建立物件實體

- new 關鍵字會傳回一個參考值，以便指定給相對應之物件變數。
- new 關鍵字會配置一個實體的記憶體空間給一個物件，並將其參考值指向所宣告的物件變數。

MyClass obj = new MyClass()