OOP 與類別

TYIC桃高資訊社

物件導向

物件導向程式設計(Object-oriented programming，簡稱 OOP)
是指使用物件(object)的程式設計模式
而物件就是類別(class)的實例(instance)
類別定義了成員(member)和方法(method)
物件則真正擁有這些東西
且每個物件都是獨立的，互不相干

類別

類別定義方式如右，名稱建議使用大駝峰命名法
也可以在前方加上
存取修飾子(Access Modifier)中的 public
表示公開的

```java
class 類別名稱 {
    成員...
    方法...
}
```

一個檔案中
可以有多個頂級(top level)類別
但只能有一個公開頂級類別
且公開頂級類別的名稱要和檔名一致

```java
public class 類別名稱 {
    成員...
    方法...
}
```

類別

```java
class 類別名稱 {
    資料型別 成員名稱;
    資料型別 成員名稱 = 值;

    存取修飾子 final static 資料型別 成員名稱;
    存取修飾子 final static 資料型別 成員名稱 = 值;

    返回值型別 函式名稱(參數型別1 參數名稱1, 參數型別2 參數名稱2, ...) {
        陳述式...
    }

    存取修飾子 static 返回值型別 函式名稱(參數型別1 參數名稱1, 參數型別2 參數名稱2, ...) {
        陳述式...
    }


    public static void main(String[] args) {}
}
```

動態與靜態

沒有 `static` 表示是動態的，有 `static` 表示是靜態的
而兩者的區別在於：
動態的在被使用時才會分配記憶體(memory)
而靜態的則是在程式一開始就分配記憶體

動態成員和方法需要透過物件來存取
而靜態成員和方法則須透過類別來存取

靜態方法與靜態成員

要存取類別或物件的方法或成員

須使用 "." (存取運算子，access operator)

呼叫靜態方法：

```java
類別名稱.靜態方法名稱(引數1, 引數2, ...)
```

存取靜態成員：

```java
類別名稱.靜態成員名稱
```

若存取的靜態方法或靜態成員與當前屬同類別

且作用域中沒有其他的同名方法或變數，則可省略類別名稱

靜態方法與靜態成員

```java
import java.util.Scanner;

public class Main {
    static void printIsPrime(int number) {
        if (Util.isPrime(number)) {
            System.out.printf("%d is prime%n",
                    number);
            return;
        }
        System.out.printf("%d is not prime%n",
                number);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int base = scanner.nextInt();
        int power = scanner.nextInt();
        printIsPrime(base);
        printIsPrime(power);
        System.out.printf("%d ^ %d = %d", base,
                power, Util.pow(base, power));
    }
}
```

```java
class Util {
    static boolean isPrime(int number) {
        for (int i = 2; i * i <= number; i++) {
            if (number % i == 0) return false;
        }
        return true;
    }

    static int pow(int base, int power) {
        int result = 1;
        if (power >= 0) {
            for (int i = 0; i < power; i++) {
                result *= base;
            }
            return result;
        }
        for (int i = -power; i > 0; i--) {
            result *= base;
        }
        return 1 / result;
    }
}
```

```
2 4
2 is prime
4 is not prime
2 ^ 4 = 16
```
console

物件

可以使用底下的方式創建特定類別的實例(物件)

```
new 類別名稱(args)
```
java

而要存取物件的成員和方法

須使用 "."(存取運算子，access operator)

```
物件.成員或方法
```
java

```
喜歡你的第1年，我還沒有告白
喜歡你的第2年，我還是沒告白
喜歡你的第3年，我還是沒告白
喜歡你的第4年，我還是沒告白
喜歡你的第5年，我還是沒告白
喜歡你的第6年，我終於告白了
```
output

```java
public class Main {
    public static void main(String[] args) {
        int i = 1;
        final String PREFIX = "喜歡你的第";
        final String SUFFIX = new String("年，我還是沒告白");
        System.out.println(new StringBuilder().append(PREFIX).append(i++).append("年，我還沒有告白"));
        System.out.println(new StringBuilder().append(PREFIX).append(i++).append(SUFFIX));
        System.out.println(new StringBuilder().append(PREFIX).append(i++).append(SUFFIX));
        System.out.println(new StringBuilder().append(PREFIX).append(i++).append(SUFFIX));
        System.out.println(new StringBuilder().append(PREFIX).append(i++).append(SUFFIX));
        System.out.println(new StringBuilder().append(PREFIX).append(i++).append("年，我終於告白了"));
    }
}
```
java

創建物件

呼叫方法

物件

呼叫動態方法：

```java
物件.動態方法名稱(引數1, 引數2, ...)
```

存取動態成員：

```java
物件.動態成員名稱
```

若存取的動態方法或動態成員與當前屬同類別

且作用域中沒有其他的同名方法或變數，則可省略物件

```java
public class Main {
    public static void main(String[] args) {
        Person person1 = new Person();
        person1.age = 60;
        person1.name = "任嫌齊";
        System.out.println("person1 創建完成");
        person1.printInfo();
        Person person2 = new Person();
        person2.age = 65;
        person2.name = "李宗聖";
        System.out.println("person2 創建完成");
        person1.printInfo();
        person2.printInfo();
    }
}
```

```java
class Person {
    int age = 0;
    String name;

    void printInfo() {
        System.out.printf("姓名：%s 年齡：%d %n",
                name, age);
    }
}
```

```output
person1 創建完成
姓名：任嫌齊 年齡：60
person2 創建完成
姓名：任嫌齊 年齡：60
姓名：李宗聖 年齡：65
```

this

若存取的動態成員與當前屬同類別

但作用域中有其他同名變數

則必須使用以下方式來指定存取動態成員，否則會存取同名變數

this.動態成員名稱

java

```java
public class Main {
    public static void main(String[] args) {
        Person person1 = new Person();
        person1.setAge(60);
        person1.name = "任嫌齊";
        person1.printInfo();
        Person person2 = new Person();
        person2.setAge(-65);
        person2.name = "李宗聖";
        person2.printInfo();
    }
}
```

```
姓名：任嫌齊  年齡：60
姓名：李宗聖  年齡：0
```

output

```java
class Person {
    int age = 0;
    String name;

    void printInfo() {
        System.out.printf("姓名：%s 年齡：%d %n",
                name, age);
    }

    void setAge(int age) {
        if (age < 0) age = 0;
        this.age = age;
    }
}
```

java

建構子

建構子(constructor)是一種特殊的動態方法

方法名稱與類別名稱完全相同

而且不需要返回型別及返回值

會在創建物件時被呼叫

```
public class Main {
    public static void main(String[] args) {
        Person person1 = new Person(60, "任嫌齊");
        person1.printInfo();
        Person person2 = new Person(-65, "李宗聖");
        person2.printInfo();
    }
}
```

```
姓名：任嫌齊  年齡：60
姓名：李宗聖  年齡：0
```

output

```
class Person {
    int age = 0;
    String name;

    Person(int age, String name) {
        setAge(age);
        this.name = name;
    }

    void printInfo() {
        System.out.printf("姓名：%s 年齡：%d %n",
                name, age);
    }

    void setAge(int age) {
        if (age < 0) age = 0;
        this.age = age;
    }
}
```

補充：解構子

因為 Java 有垃圾回收(Garbage Collection)機制
而且 Java 不允許手動更改記憶體
所以 Java 中其實並沒有解構子(destructor)
若想在物件被銷毀(destory)後執行某些事情
須覆寫(override) "finalize" 方法

存取修飾子

存取修飾子是用來進行權限的管理