

測試环境：

1) java

2) IDEA

電子三甲

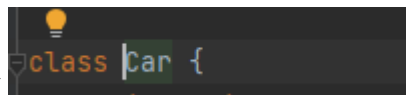
裘翀皓

108360150

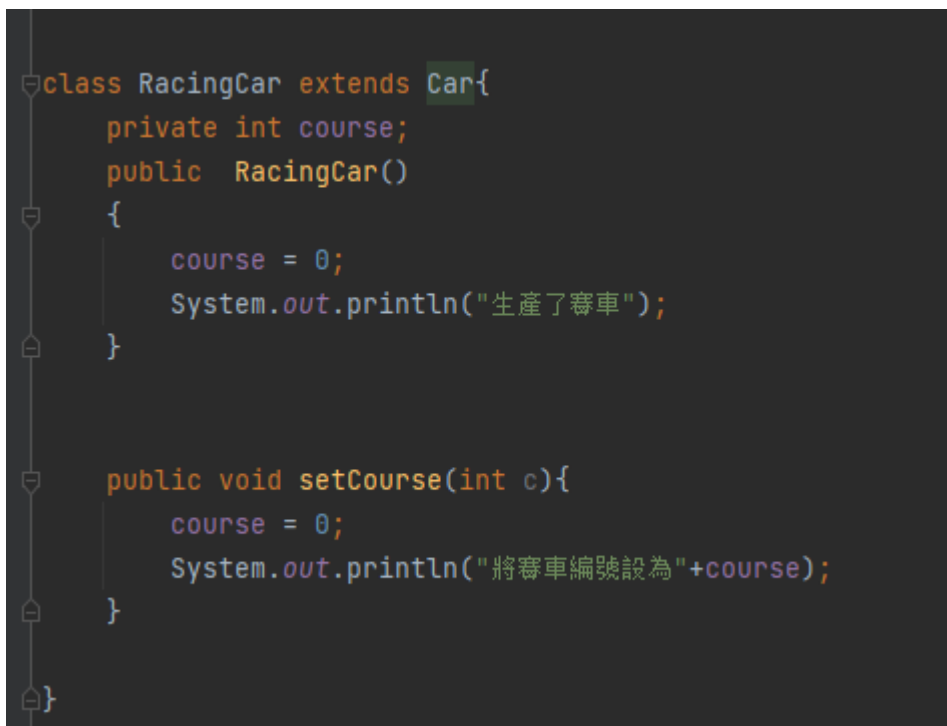
心得：

繼上次學習了Class的各種操作，這次我們學習了Java中繼承（extend）的用法。

當我們構建了



之後，



```
class RacingCar extends Car{
    private int course;
    public RacingCar()
    {
        course = 0;
        System.out.println("生產了賽車");
    }

    public void setCourse(int c){
        course = 0;
        System.out.println("將賽車編號設為"+course);
    }
}
```

可以使用extends去繼承這個類別。在這次講義中，Car就是父類別，而RacingCar作為現實中Car的一個種類，在java中我們把他定義成Car類別的子類別。

```
rccar1.setCar( n: 1234, g: 20.5);  
rccar1.setCourse(5);
```

子類別可以使用父類別的方法

若無特別設定，子類別在建立前會先呼叫父類別無參數的建構式來幫助父類別做初始化，看輸出結果可以發現父類別建構子是最先執行的：

```
public Car() {  
    num = 0;  
    gas = 0.0;  
    System.out.println("生產了車子");  
}
```

```
public RacingCar()  
{  
    course = 0;  
    System.out.println("生產了賽車");  
}
```

```
↓ 生產了車子  
  生產了賽車
```

從子類別的建構式，呼叫父類別中特定的建構式，必須使用「`super()`」關鍵字：

```
RacingCar rccar1 = new RacingCar( n: 1234, g: 20.5, c: 5);
```

```
public Car(int n ,double g) {  
    num = n;  
    gas = g;  
    System.out.println("將車號設為"+ num +"汽油里設為"+gas);  
}
```

```
public RacingCar(int n,double g,int c)  
{  
    super(n,g);  
    course = c ;  
    System.out.println("生產了編號為"+course +"的賽車");  
}
```

```
將車號設為1234汽油里設為20.5  
生產了編號為5的賽車
```

對於子類別來說，父類別的成員有兩種。一種是private，另一種是protected。
private類別不能被子類別讀取，protected可以被子類別讀取。

Overriding：

我們可以在子類別中建構與父類別名稱一樣的方法，這樣的話如果我們呼叫這個方法，只會做子類別中的那個方法，被複寫掉的方法不會執行，但是這兩個方法的「參數」、「方法名稱」、「傳回值型態」都必須與父類別的方法相同。

呼叫哪個方法與類別本身變數無關，與new的物件有關
因為我new的是賽車，所以show用的是賽車的方法

```
public static void main (String[] args)
{
    Car car1 = new RacingCar();
    car1.setCar( n: 1234, g: 20.5);
    car1.show();
}
```

```
public void show()
{
    System.out.println("賽車的車號是" + num);
    System.out.println("汽油量是" + gas);
    System.out.println("賽車編號是" + course);
}
```

但是我們car是Car類別，所以不能使用RacingCar的setCourse，如下圖紅色字所示。

```
public static void main (String[] args)
{
    Car car1 = new RacingCar();
    car1.setCar( n: 1234, g: 20.5);
    car1.setCourse(5);
    car1.show();
}
```

toString()方法的主要是：把「物件」轉成「字串」並將結果傳回原呼叫程式。

```
public String toString()//此處複寫
{
    String str = "車號:" + num + ", 汽車量:" + gas;
    return str;
}
```

車號:1234, 汽車量:20.5

如果我們這裡不覆寫的，傳回會是：

```
/*public String toString()//此處複寫
{
    String str = "車號:" + num + ", 汽車量:" + gas;
    return str;
}*/
```

p24.Car@4f3f5b24

前面的Car是此物件的類別名稱，後面的XXX是此物件的十六進位雜湊碼

這是抽象類別和抽象方法，

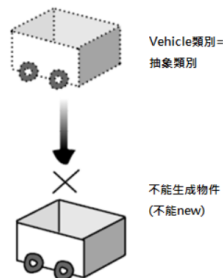
```
abstract class Vehicle
{
    protected int speed;
    public void setSpeed(int s)
    {
        speed = s;
        System.out.println("將速度設為" + speed + "了");
    }
    abstract void show();
}
```

📖 抽象類別(1/6)

當定義類別時，可以僅宣告方法名稱而不實作當中的邏輯，這樣的方法稱之為**抽象方法(abstract method)**，如果一個類別中包括了抽象方法，則該類別稱之為**抽象類別(abstract class)**，抽象類別是個未定義完全的類別，所以它不能被用來生成物件(new)，它只能被擴充，並於擴充後完成未完成的抽象方法定義。

語法

```
abstract class 類別名稱
{
    欄位的宣告;
    abstract 傳回值的型態 方法名稱;
    ...
}
```



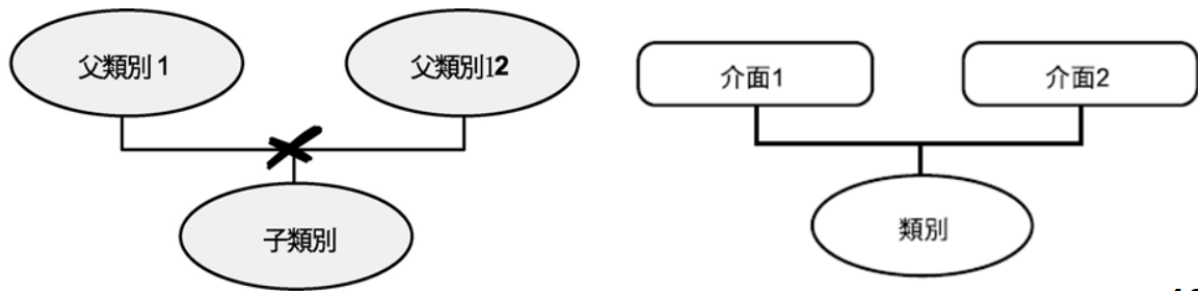
介面和使用介面構建類別：

```
interface iVehicle
{
    int weight = 1000;
    void show();
}
```

```
class Plane implements iVehicle
{
    private int flight;
    public Plane(int f)
    {
        flight = f;
        System.out.println("生產了"+flight + "班次的飛機");
    }

    public void show() { System.out.println("飛機的班次是" + flight); }
}
```

介面和類別很像，但是介面不能像類別一樣用**new XXXX**來新建物件。
使用介面可以達成多重繼承：



```
interface iVehicle
{
    void vShow();
}

interface iMaterial
{
    void mShow();
}

class Car implements iVehicle,iMaterial{
    private int num;
    private double gas;

    public Car(int n, double g) {
        num = n;
        gas = g;
        System.out.println("生產了車號為" + num + "汽油量" + gas +"的車子");
    }
}
```

GitHub :

https://github.com/547320328/Homework_software/tree/main/3