CHAPTER 4

繼承 (ch11)

介面 (ch12)

本節介紹

- 類別的繼承
- 呼叫父類別的建構式
- 保護成員
- 覆寫(Overriding)
- 類別的階層
- 繼承物件類別
- 抽象類別

關鍵詞彙

- 繼承
- 父類別
- 子類別
- super()
- 保護成員
- 覆寫
- 抽象類別
- final
- instanceof

類別的繼承(1/3)

根據既有類別為基礎衍生出另一個類別，此動作稱為類別的延伸(extends)，新的類別完全接收既有類別的成員，此現象稱為繼承(inheritance)。

繼承類別需使用extends

語法

```
類別 子類別名稱 extends 父類別名稱
{
        子類別新增成員…
        子類別的建構式(參數列表)
        {
                …
        }
}
```

父類別

```
class 汽車
{
   車號;
   汽油量;
   顯示車號和汽油量的功能…
}
```

既有的汽車類別

子類別

```
class 賽車 extends 汽車
{
   賽車編號;
   表示賽車編號的功能…
}
```

新的類別

新增的性質和功能

類別的繼承(2/3)

Sample4_1.java

```
1  public class Sample4_1
2  {
3     public static void main(String[] args)
4     {
5        RacingCar rccar1;
6        rccar1 = new RacingCar();
7
8        rccar1.setCar(1234, 20.5);
9
10       rccar1.setCourse(5);
11    }
12 }
```

建立子類別的物件

呼叫子類別繼承的方法

呼叫子類別新增的方法

Console | Markers | Proper

<terminated> Sample4_1 [Java App

```
生產了車子
生產了賽車
將車號設為1234，汽油量設為20.5
將賽車編號設為5
```

輸出結果

此頁範例接續下一頁

類別的繼承(3/3)

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

宣告成私有的欄位，子類別無法繼承

被子類別繼承的方法

父類別

```
14  class Car
15  {
16    private int num;
17    private double gas;
18
19    public Car()
20    {
21      num = 0;
22      gas = 0.0;
23      System.out.println("生產了車子");
24    }
25
26    public void setCar(int n, double g)
27    {
28      num = n;
29      gas = g;
30      System.out.println("將車號設為" + num + "，汽油量設為" + gas);
31    }
32
33    public void show()
34    {
35      System.out.println("車號是" + num);
36      System.out.println("汽油量是" + gas);
37    }
38  }
```

子類別新增的欄位

賽車繼承成了汽車

子類別的建構式

子類別新增的方法

子類別

```
40  class RacingCar extends Car
41  {
42    private int course;
43
44    public RacingCar()
45    {
46      course = 0;
47      System.out.println("生產了賽車");
48    }
49
50    public void setCourse(int c)
51    {
52      course = c;
53      System.out.println("將賽車編號設為" + course);
54    }
55  }
```

呼叫父類別的建構式(1/2)

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

Sample4_2.java

```
1  public class Sample4_2
2  {
3      public static void main(String[] args)
4      {
5          RacingCar rccar1 = new RacingCar(1234, 20.5, 5);
6      }
7  }
```

呼叫子類別三個參數的建構式

Console | Markers | Properties

<terminated> Sample4_2 [Java Applicati

生產了車號為1234，汽油量為20.5的車子
生產了編號為5的賽車

輸出結果

此頁範例接續下一頁

呼叫父類別的建構式(2/2)

父類別

```
 9  class Car
10  {
11    private int num;
12    private double gas;
13
14    public Car()
15    {
16      num = 0;
17      gas = 0.0;
18      System.out.println("生產了車子");
19    }
20
21    public Car(int n, double g)
22    {
23      num = n;
24      gas = g;
25      System.out.println("生產了車號為" + num + "，汽油量為" + gas + "的車子");
26    }
27
28    public void setCar(int n, double g)
29    {
30      num = n;
31      gas = g;
32      System.out.println("將車號設為" + num + "，汽油量設為" + gas);
33    }
34
35    public void show()
36    {
37      System.out.println("車號是" + num);
38      System.out.println("汽油量是" + gas);
39    }
40  }
```

被子類別呼叫到的建構式

子類別

```
42  class RacingCar extends Car
43  {
44    private int course;
45
46    public RacingCar()
47    {
48      course = 0;
49      System.out.println("生產了賽車");
50    }
51
52    public RacingCar(int n, double g, int c)
53    {
54      super(n, g);
55      course = c;
56      System.out.println("生產了編號為" + course + "的賽車");
57    }
58
59    public void setCourse(int c)
60    {
61      course = c;
62      System.out.println("將賽車編號設為" + course);
63    }
64  }
```

呼叫父類別雙參數的建構式

保護成員(1/3)

成員(member)指的是一個類別(class)內部的「欄位(field)」和「方法(method)」。如果「父類別裡面的成員原本就是私有成員(private member)，則子類別將無法存取到這些私有類別的資料」，所以在父類別裡面，對於成員改用保護成員(protected)，子類別將可以存取父類別內部保護成員的資料。

✓ 私有成員 (private member)

私有成員不能被同一類別以外的其他程式存取，因此如果父類別裏面的成員原本就是私有成員，則子類別將無法存取到這些私有類別的資料。

✓ 保護成員 (protected member)

父類別內部的成員一旦設定為「protected」，其性質和私有成員有所不同，也就是說：子類別可以存取父類別內部保護成員的資料。

保護成員(2/3)

Sample4_3.java

```
1  public class Sample4_3
2  {
3      public static void main(String[] args)
4      {
5          RacingCar rccar1;
6          rccar1 = new RacingCar();
7
8          rccar1.newShow();
9      }
10 }
```

Console Markers

<terminated> Sample4_3

生產了車子
生產了賽車
賽車的車號是0
汽油量是0.0
賽車編號是0

輸出結果

此頁範例接續下一頁

保護成員(3/3)

小知識

若父類別的成員為保護成員的話，子類別可以存取，但是子類別的子類別就不行存取。

父類別

```
12  class Car
13  {
14    protected int num;
15    protected double gas;
16
17    public Car()
18    {
19      num = 0;
20      gas = 0.0;
21      System.out.println("生產了車子");
22    }
23
24    public void setCar(int n, double g)
25    {
26      num = n;
27      gas = g;
28      System.out.println("將車號設為" + num + "，汽油量設為" + gas);
29    }
30
31    public void show()
32    {
33      System.out.println("車號是" + num);
34      System.out.println("汽油量是" + gas);
35    }
36  }
```

將父類別的成員設為保護成員

子類別

```
38  class RacingCar extends Car
39  {
40    private int course;
41
42    public RacingCar()
43    {
44      course = 0;
45      System.out.println("生產了賽車");
46    }
47
48    public void setCourse(int c)
49    {
50      course = c;
51      System.out.println("將賽車編號設為" + course);
52    }
53
54    public void newShow()
55    {
56      System.out.println("賽車的車號是" + num);
57      System.out.println("汽油量是" + gas);
58      System.out.println("賽車編號是" + course);
59    }
60  }
```

子類別可以存取父類別的保護成員

覆寫(Overriding)(1/9)

覆寫(Overriding)是指「子類別」繼承父類別，但是改寫父類別既有的方法，但其方法的「參數」、「方法名稱」、「傳回值型態」都必須與父類別的方法相同。

Sample4_4.java

```
1  public class Sample4_4
2  {
3    public static void main(String[] args)
4    {
5        RacingCar rccar1;
6        rccar1 = new RacingCar();
7
8        rccar1.setCar(1234, 20.5);
9        rccar1.setCourse(5);
10
11       rccar1.show();
12   }
13 }
```

當我們呼叫show的方法後，看輸出結果我們可以發現只會做子類別的show方法，父類別的show方法已經被覆寫掉了

Console | Markers | Prope

<terminated> Sample4_4 [Java App

```
生產了車子
生產了賽車
將車號設為1234，汽油量設為20.5
將賽車編號設為5
賽車的車號是1234
汽油量是20.5
賽車編號是5
```

輸出結果

此頁範例接續下一頁

覆寫(Overriding)(2/9)

父類別

```
15  class Car
16  {
17    protected int num;
18    protected double gas;
19
20    public Car()
21    {
22      num = 0;
23      gas = 0.0;
24      System.out.println("生產了車子");
25    }
26
27    public void setCar(int n, double g)
28    {
29      num = n;
30      gas = g;
31      System.out.println("將車號設為" + num + "，汽油量設為" + gas);
32    }
33
34    public void show()
35    {
36      System.out.println("車號是" + num);
37      System.out.println("汽油量是" + gas);
38    }
39  }
```

如果複寫了父類別的方法，eclipse會有個三角形表示此方法已覆寫

父類別的Show方法

子類別

```
41  class RacingCar extends Car
42  {
43    private int course;
44
45    public RacingCar()
46    {
47      course = 0;
48      System.out.println("生產了賽車");
49    }
50
51    public void setCourse(int c)
52    {
53      course = c;
54      System.out.println("將賽車編號設為" + course);
55    }
56
57    public void show()
58    {
59      System.out.println("賽車的車號是" + num);
60      System.out.println("汽油量是" + gas);
61      System.out.println("賽車編號是" + course);
62    }
63  }
```

子類別的Show方法

覆寫(Overriding)(3/9)

之前的範例，我們在子類別當中宣告變數，並完成指向物件的動作。

事實上，使用父類別的變數也能存取子類別的物件。

```
RacingCar rccar1;
rccar1 = new RacingCar();
```

以子類別的變數來操作子類別的物件

```
Car car1;
car1 = new RacingCar();
```

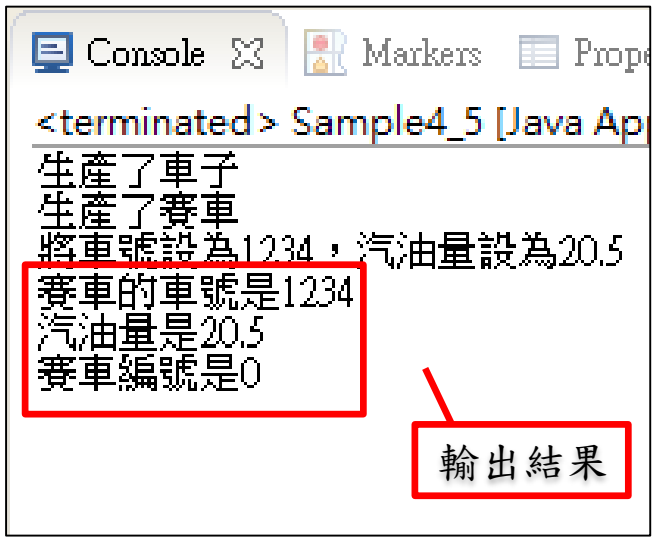
以父類別的變數來操作子類別的物件

覆寫(Overriding)(4/9)

從父類別的變數操作子類別的物件

Sample4_5.java

```
1  public class Sample4_5
2  {
3      public static void main(String[] args)
4      {
5          Car car1;
6          car1 = new RacingCar();
7
8          car1.setCar(1234, 20.5);
9
10         car1.show();
11     }
12 }
```



從輸出結果我們可以發現還是呼叫出子類別的show方法，由此可知要呼叫哪個方法與類別本身變數無關，與new的物件有關

我們沒有辦法呼叫到RacingCar類別自己新增的方法，因為我們是以Car類別的變數存取，所以只能對Car類別有的方法存取

此頁範例接續下一頁

覆寫(Overriding)(5/9)

父類別

```
19  public Car()
20  {
21    num = 0;
22    gas = 0.0;
23    System.out.println("生產了車子");
24  }
25
26  public void setCar(int n, double g)
27  {
28    num = n;
29    gas = g;
30    System.out.println("將車號設為" + num + "，汽油量設為" + gas);
31  }
32
33  public void show()
34  {
35    System.out.println("車號是" + num);
36    System.out.println("汽油量是" + gas);
37  }
38
```

父類別的Show方法

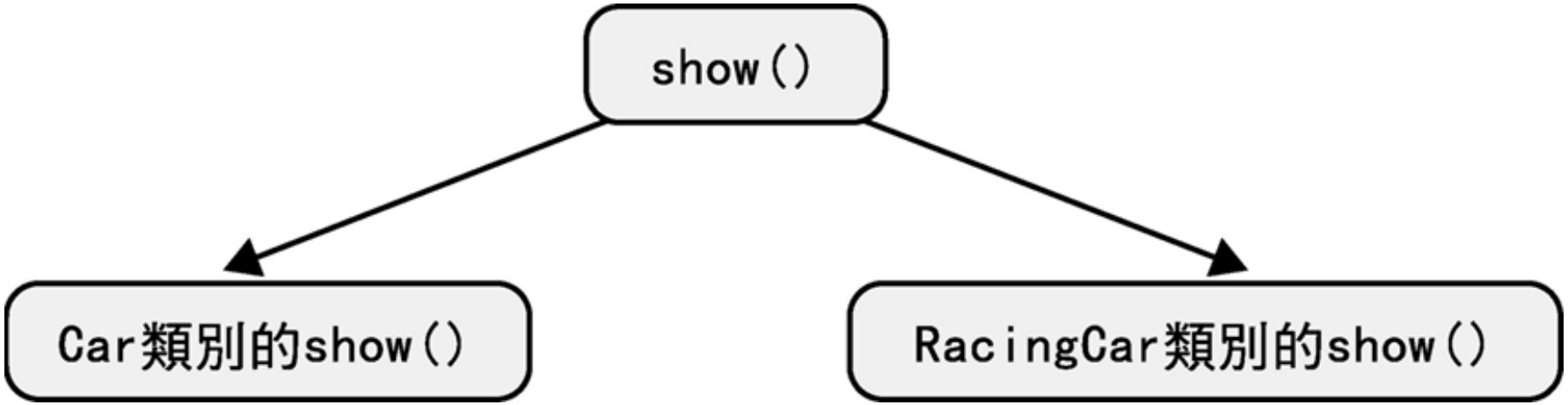
子類別

```
40  class RacingCar extends Car
41  {
42    private int course;
43
44    public RacingCar()
45    {
46      course = 0;
47      System.out.println("生產了賽車");
48    }
49
50    public void setCourse(int c)
51    {
52      course = c;
53      System.out.println("將賽車編號設為" + course);
54    }
55
56    public void show()
57    {
58      System.out.println("賽車的車號是" + num);
59      System.out.println("汽油量是" + gas);
60      System.out.println("賽車編號是" + course);
61    }
62  }
```

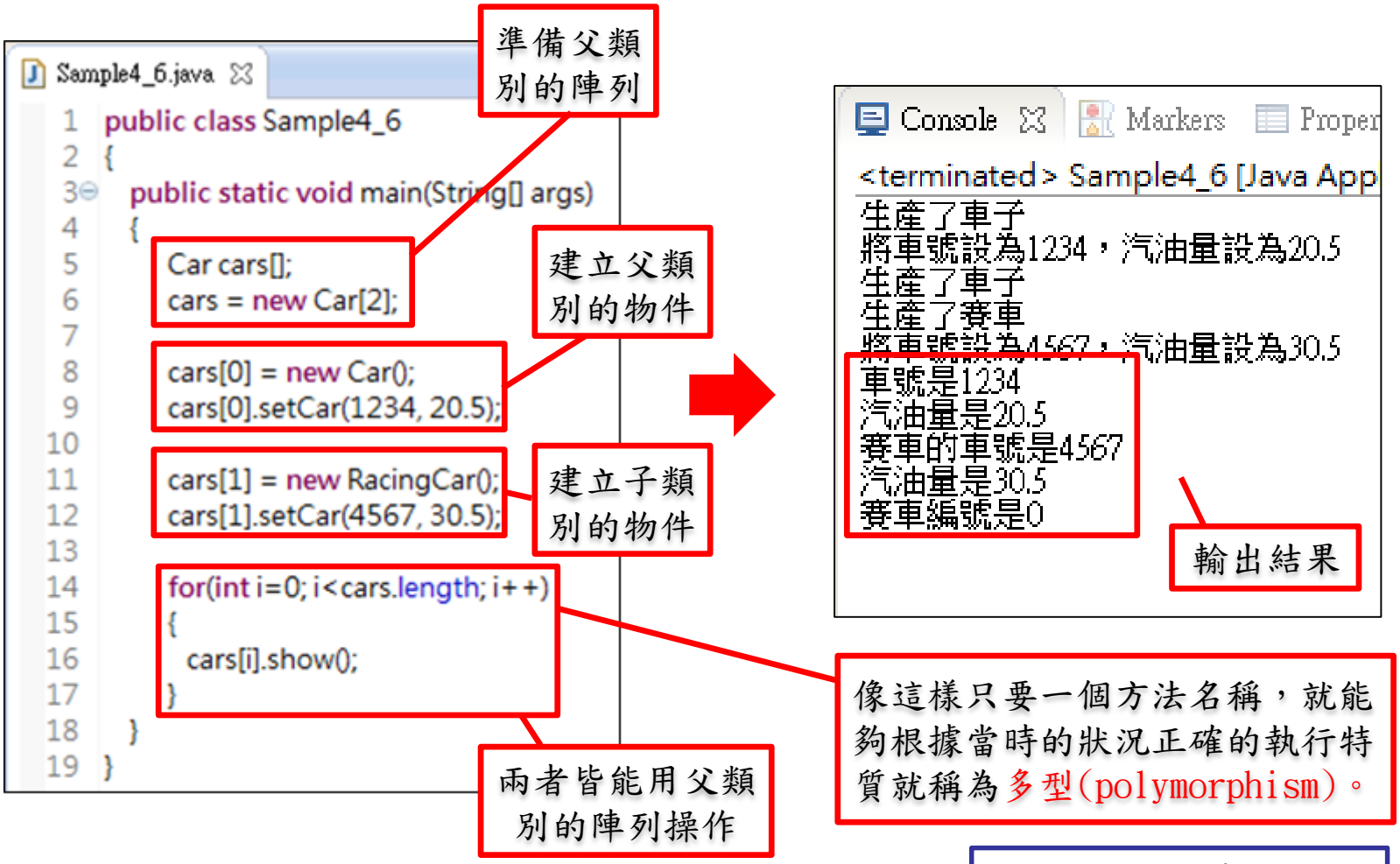
子類別的Show方法

覆寫(Overriding)(6/9)

在一個稍為大型的程式中一定會產生各式各樣的物件，因此對各種類別所產生的物件加以管理是必然的事，於是有人就想到利用父類別的陣列來儲存各種類別所產生的物件。透過覆寫就能一併操作父類別和子類別的物件，像這樣只要一個方法名稱，就能夠根據當時的狀況正確的執行特質就稱為**多型(polymorphism)**。



覆寫(Overriding)(7/9)



此頁範例接續下一頁

覆寫(Overriding)(8/9)

父類別

```
21 class Car
22 {
23   protected int num;
24   protected double gas;
25
26   public Car()
27   {
28     num = 0;
29     gas = 0.0;
30     System.out.println("生產了車子");
31   }
32
33   public void setCar(int n, double g)
34   {
35     num = n;
36     gas = g;
37     System.out.println("將車號設為" + num + "，汽油量設為" + gas);
38   }
39
40   public void show()
41   {
42     System.out.println("車號是" + num);
43     System.out.println("汽油量是" + gas);
44   }
45 }
```

父類別的Show方法

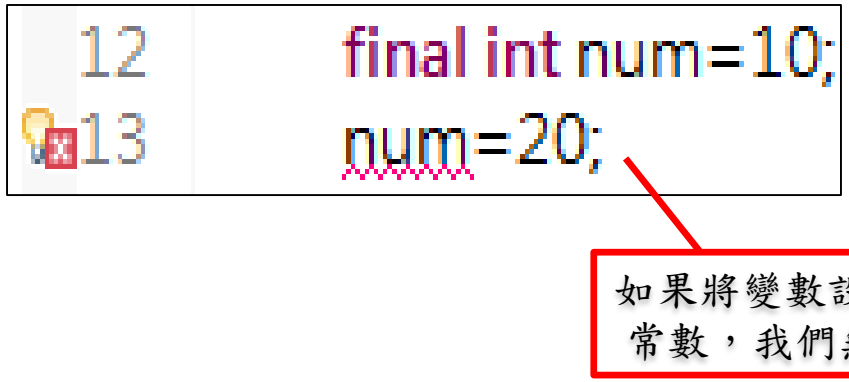
子類別

```
47 class RacingCar extends Car
48 {
49   private int course;
50
51   public RacingCar()
52   {
53     course = 0;
54     System.out.println("生產了賽車");
55   }
56
57   public void setCourse(int c)
58   {
59     course = c;
60     System.out.println("將賽車編號設為" + course);
61   }
62
63   public void show()
64   {
65     System.out.println("賽車的車號是" + num);
66     System.out.println("汽油量是" + gas);
67     System.out.println("賽車編號是" + course);
68   }
69 }
```

子類別的Show方法

覆寫(Overriding)(9/9)

當在設計類別的時候，其中可能有一些父類別的方法根本不想讓它在子類別中產生被覆寫(overriding)的現象。如果有這種需求時，必須在父類別的方法的第一行加上**final修飾子**，就會禁止被覆寫現象繼續產生。簡單來說，final修飾子功能如C語言中的const，當變數一旦被初始化後便不可做修改。



覆寫(Overriding)(9/9)

父類別

```
21  class Car
22  {
23    protected int num;
24    protected double gas;
25
26    public Car()
27    {
28      num = 0;
29      gas = 0.0;
30      System.out.println("生產了車子");
31    }
32
33    public void setCar(int n, double g)
34    {
35      num = n;
36      gas = g;
37      System.out.println("將車號設為" + num + "，汽油量設為" + gas);
38    }
39
40    public final void show()
41    {
42      System.out.println("車號是" + num);
43      System.out.println("汽油量是" + gas);
44    }
45  }
```

只要在方法中指定final

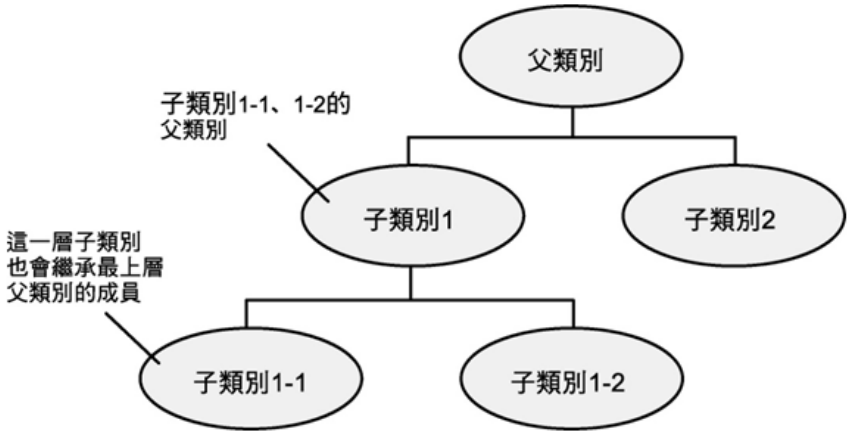
子類別

```
47  class RacingCar extends Car
48  {
49    private int course;
50
51    public RacingCar()
52    {
53      course = 0;
54      System.out.println("生產了賽車");
55    }
56
57    public void setCourse(int c)
58    {
59      course = c;
60      System.out.println("將賽車編號設為" + course);
61    }
62
63    public void show()
64    {
65      System.out.println("賽車的車號是" + num);
66      System.out.println("汽油量是" + gas);
67      System.out.println("賽車編號是" + course);
68    }
69  }
```

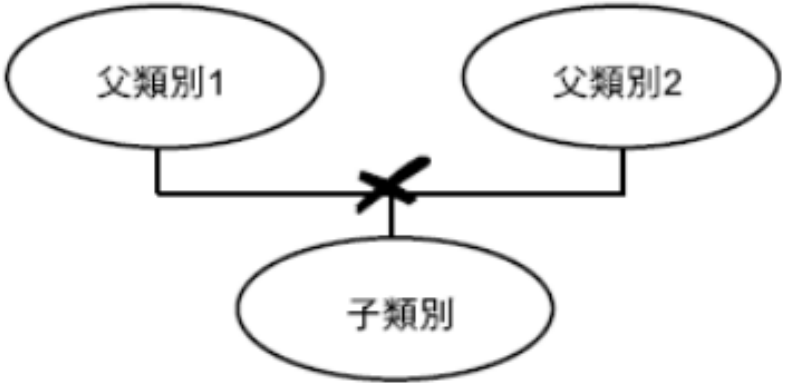
就無法在子類別中定義show()方法

類別的階層

✓ 對Java來說，一個父類別可以延伸出數個子類別，子類別再繼續往下延伸，可以再產生新的子類別。這時候整個類別家族的結構如下圖所示:



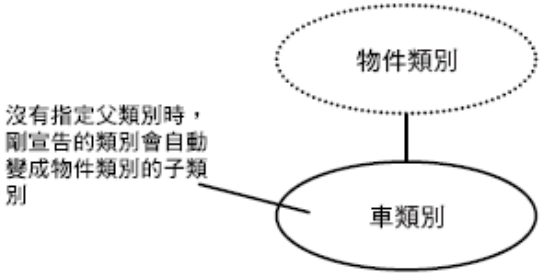
✓ 但Java無法讓一個子類別同時繼承自多個父類別，如下圖:



繼承物件類別(object class)(1/6)

在Java中，所有的物件都隱含擴充了**Object類別**，Object類別是Java程式中所有類別的父類別。當定義一個類別時，如下：

如果什麼都沒有指定的話，就會變成以物件(object)類別為父類別的子類別了




```
class Car
{
  protected int num;
  protected double gas;

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

=

```
class Car extends Object
{
  protected int num;
  protected double gas;

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

繼承物件類別(object class)(2/6)

下面介紹幾種Object類別常用的方法:

方法名稱	傳回型別	功能
equals(Object obj)	boolean	驗證某兩物件是否相同
getClass()	Class	傳回[該物件究竟屬於哪一個類別]等資訊
toString()	String	將物件轉成字串

繼承物件類別(object class)(3/6)

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。

Sample4_7.java

```
1  public class Sample4_7
2  {
3     public static void main(String[] args)
4     {
5        Car car1 ;
6        car1 = new Car();
7
8        car1.setcar(1234, 20.5);
9        System.out.println(car1);
10    }
11 }
12 class Car
13 {
14    protected int num;
15    protected double gas;
16
17    public Car()
18    {
19       num = 0;
20       gas = 0.0;
21       System.out.println("生產了車子");
22    }
23
24    public void setcar(int n, double g)
25    {
26       num = n;
27       gas = g;
28       System.out.println("將車號設為"+num+"，汽油量設為"+gas);
29    }
30
31    public String toString()
32    {
33       String str = "車號:"+num+";汽車量:"+gas;
34       return str;
35    }
36
37 }
```

一旦執行System.out.println類別方法，此方法內會呼叫Object類別的toString方法來期輸入的物件轉為字串來避免輸出錯誤

小知識

如果我們這裡不覆寫的，傳回會是[Car@XXX]，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

Console | Markers | Prop

<terminated> Sample4_7 [Java Ap

```
生產了車子
將車號設為1234，汽油量設為20.5
車號:1234;汽車量:20.5
```

輸出結果

此字串為傳回值

覆寫Object類別的toString方法

繼承物件類別(object class)(4/6)

equals()方法的其主要用意是：驗證某兩個物件是否相同，是的話傳回true，否的話傳回false。

Sample4_8.java

```
1  public class Sample4_8
2  {
3    public static void main(String[] args)
4    {
5      Car car1 = new Car();
6      Car car2 = new Car();
7
8      Car car3;
9      car3 = car1;
10
11     System.out.println("car1與 car2相同 " + car1.equals(car2));
12     System.out.println("car1與 car3相同 " + car1.equals(car3));
13   }
14 }
15
16 class Car
17 {
18   protected int num;
19   protected double gas;
20
21   public Car()
22   {
23     num = 0;
24     gas = 0.0;
25     System.out.println("生產了車子");
26   }
27 }
```

car1與car2分別代表不同的物件

car1與car3分別代表相同的物件

Console Markers

<terminated> Sample4_8

生產了車子
生產了車子
car1與 car2相同 false
car1與 car3相同 true

輸出結果

繼承物件類別(object class)(5/6)

Sample4_9.java

```
1  public class Sample4_9
2  {
3    public static void main(String[] args)
4    {
5      Car[] cars;
6      cars = new Car[2];
7
8      cars[0] = new Car();
9      cars[1] = new RacingCar();
10
11     for(int i=0; i<cars.length; i++)
12     {
13       Class cl = cars[i].getClass();
14       System.out.println("第" + (i+1) + "個物件的類別是" + cl);
15     }
16   }
17 }
```

第一個物件為Car類別

第二個物件為RacingCar類別

getClass方法回傳Class物件

Console | Markers | Prop

<terminated> Sample4_9 [Java Ap

```
生產了車子
生產了車子
生產了賽車
第1個物件的類別是class Car
第2個物件的類別是class RacingCar
```

輸出結果

此頁範例接續下一頁

繼承物件類別(object class)(6/6)

getClass()方法的用意是：傳回該物件究竟屬於哪一個類別的相關資訊。

父類別

```
19  class Car
20  {
21    protected int num;
22    protected double gas;
23
24    public Car()
25    {
26      num = 0;
27      gas = 0.0;
28      System.out.println("生產了車子");
29    }
30  }
```

子類別

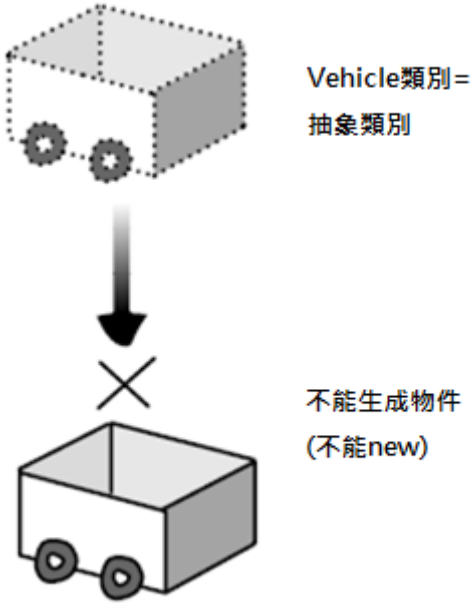
```
32  class RacingCar extends Car
33  {
34    public RacingCar()
35    {
36      System.out.println("生產了賽車");
37    }
38  }
```

抽象類別(1/6)

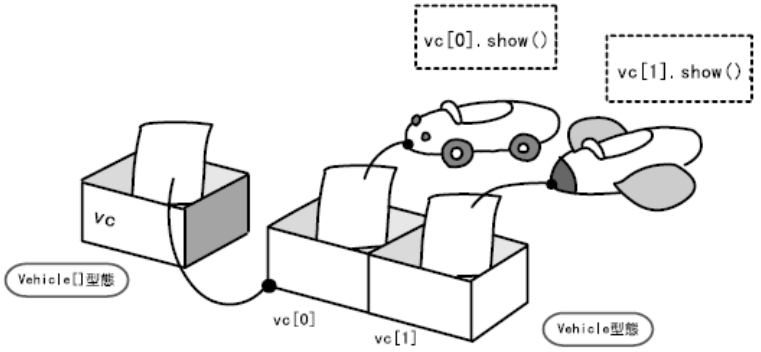
當定義類別時，可以僅宣告方法名稱而不實作當中的邏輯，這樣的方法稱之為抽象方法(abstract method），如果一個類別中包括了抽象方法，則該類別稱之為抽象類別(abstract class），抽象類別是個未定義完全的類別，所以它不能被用來生成物件(new)，它只能被擴充，並於擴充後完成未完成的抽象方法定義。

語法

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



抽象類別(2/6)



Sample4_10.java

```
1  public class Sample4_10
2  {
3    public static void main(String[] args)
4    {
5      Vehicle[] vc = new Vehicle[2];
6
7      vc[0] = new Car(1234, 20.5);
8      vc[0].setSpeed(60);
9
10     vc[1] = new Plane(232);
11     vc[1].setSpeed(500);
12
13     for (int i=0; i<vc.length;i++)
14     {
15       vc[i].show();
16     }
17   }
18 }
```

準備抽象類別的陣列

第一個物件是Car類別

第二個物件是Plane類別

呼叫Show方法會呼叫出對應該物件之類別的Show方法

Console | Markers | Properties

```
<terminated> Sample4_10 [Java Applicat
生產了車號為1234汽油量為20.5的車子
將速度設為60了
生產了232班次的飛機
將速度設為500了
車號是1234
汽油量是20.5
速度是60
飛機的班次是232
速度是500
```

輸出結果

此頁範例接續下一頁

抽象類別(3/6)

父類別

這是抽象類別

```
20  abstract class Vehicle
21  {
22     protected int speed;
23     public void setSpeed(int s)
24     {
25        speed = s;
26        System.out.println("將速度設為"+speed+"了");
27     }
28
29     abstract void show();
30  }
```

抽象方法Show

子類別1

延伸抽象類別

```
32  class Car extends Vehicle
33  {
34     private int num;
35     private double gas;
36
37     public Car (int n, double g)
38     {
39        num = n;
40        gas = g;
41        System.out.println("生產了車號為"+num+"汽油量為"+gas+"的車子");
42     }
43
44     public void show()
45     {
46        System.out.println("車號是"+num);
47        System.out.println("汽油量是"+gas);
48        System.out.println("速度是"+speed);
49     }
50  }
```

子類別2

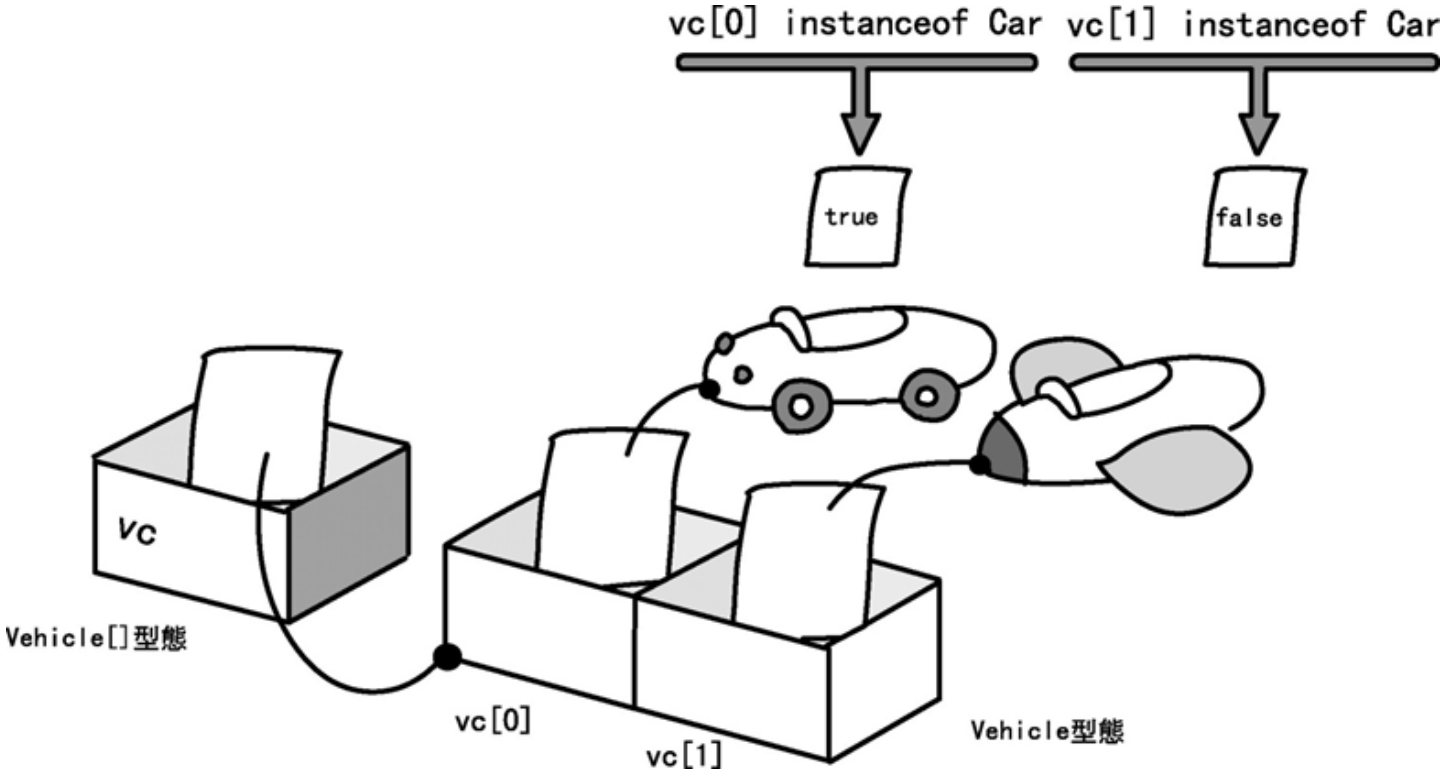
定義Show方法內的內容

```
52  class Plane extends Vehicle
53  {
54     private int flight;
55
56     public Plane(int f)
57     {
58        flight = f;
59        System.out.println("生產了"+flight+"班次的飛機");
60     }
61
62     public void show()
63     {
64        System.out.println("飛機的班次是"+flight);
65        System.out.println("速度是"+speed);
66     }
67  }
```

如果實作了父類別的抽象方法會出現此三角形

抽象類別(4/6)

使用instanceof運算子，可以知道變數所指向的物件，其所屬的究竟是哪一個類別(class)。



抽象類別(5/6)

Sample4_11.java

```
1  public class Sample4_11
2  {
3     public static void main(String[] args)
4     {
5        Vehicle[] vc;
6        vc = new Vehicle[2];
7
8        vc[0] = new Car(1234, 20.5);
9        vc[1] = new Plane(232);
10
11       for (int i = 0; i < vc.length; i++)
12       {
13          if (vc[i] instanceof Car)
14             System.out.println("第" + (i + 1) + "個物件是Car類別");
15          else
16             System.out.println("第" + (i + 1) + "個物件不是Car類別");
17       }
18    }
19 }
```

第一個物件是Car類別

第二個物件是Plane類別

檢查物件是否為Car類別

此行也可以用getclass方法達成一樣的的效果
if(vc[i].getClass() == Car.class)

Console | Markers | Properties

<terminated> Sample4_11 [Java Applicat

生產了車號為1234，汽油量為20.5的車子
生產了232班次的飛機
第1個物件是Car類別
第2個物件不是Car類別

輸出結果

此頁範例接續下一頁

抽象類別(6/6)

父類別

```
21  abstract class Vehicle
22  {
23     protected int speed;
24
25     public void setSpeed(int s)
26     {
27        speed = s;
28        System.out.println("將速度設為" + speed + "了");
29     }
30
31     abstract void show();
32  }
```

子類別1

```
34  class Car extends Vehicle
35  {
36     private int num;
37     private double gas;
38
39     public Car(int n, double g)
40     {
41        num = n;
42        gas = g;
43
44        System.out.println("生產了車號為" + num + "，汽油量為" + gas + "的車子");
45     }
46
47     public void show()
48     {
49        System.out.println("車號是" + num);
50        System.out.println("汽油量是" + gas);
51        System.out.println("速度是" + speed);
52     }
53  }
```

子類別2

```
55  class Plane extends Vehicle
56  {
57     private int flight;
58
59     public Plane(int f)
60     {
61        flight = f;
62        System.out.println("生產了" + flight + "班次的飛機");
63     }
64
65     public void show()
66     {
67        System.out.println("飛機的班次是" + flight);
68        System.out.println("速度是" + speed);
69     }
70  }
```

本節介紹

- 介面的基本介紹
- 多重繼承
- 介面的延伸
- 介面 vs 抽象類別

關鍵詞彙

- 介面
- 實作
- 多重繼承
- 父介面
- 子介面

介面的基本介紹(1/5)

介面(interface)的宣告，和宣告一般類別不同的是，原本宣告類別時一開始應該使用的「class」，在宣告介面時則改成「interface」，其他的部份和一般類別宣告一樣。

語法

```
interface  介面名稱
{
  資料型態 field名稱 = 運算式 ;
  傳回值的資料型態 method名稱();

}
```

介面的基本介紹(2/5)

使用介面的方式是把介面和類別融合在一起，特別是把介面納入到類別當中，這種情況稱為介面的實作(implementation)。

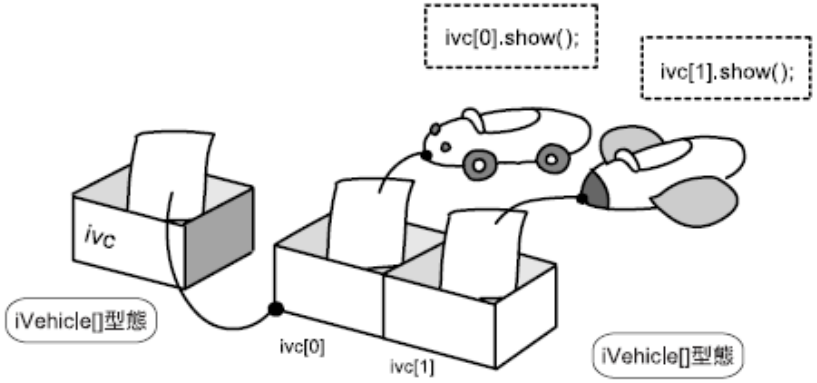
語法

```
class 類別名稱 implements 介面名稱
{
   …
}
```

介面的基本介紹(3/5)

- ✓ 介面和類別一樣有成員和方法，但是沒有建構式。
- ✓ 介面的成員不會加修飾子，但還是等於在成員前面加public static final修飾子、方法前面加上abstract修飾子。也就是說介面的欄位是常數，方法是抽象方法。
- ✓ 介面和類別很像，但是介面卻無法建立物件。也就是說，無法使用new來建立新的物件。

介面的基本介紹(4/5)



Sample4_12.java

```
1  public class Sample4_12
2  {
3    public static void main(String[] args)
4    {
5      iVehicle[] ivc;
6      ivc = new iVehicle[2];
7
8      ivc[0] = new Car(1234, 20.5);
9
10     ivc[1] = new Plane(232);
11
12     for(int i=0; i<ivc.length; i++)
13     {
14       ivc[i].show();
15     }
16   }
17 }
```

準備介面的陣列

第一個物件是Car類別

第二個物件是Plane類別

一個Show方法可以根據狀況表示成不同的結果這也是應用多型(polymorphism)的概念

Console | Markers | Properties

```
<terminated> Sample4_12 [Java Applicat
生產了車號為1234，汽油量為20.5的車子
生產了232班次的飛機
車號是1234
汽油量是20.5
飛機的班次是232
```

輸出結果

此頁範例接續下一頁

介面的基本介紹(5/5)

介面

```
19  interface iVehicle
20  {
21      int weight = 1000;
22      void show();
23  }
```

宣告介面

這是抽象方法

欄位一定要初始化為常數

類別1

```
24  class Car implements iVehicle
25  {
26    private int num;
27    private double gas;
28
29    public Car(int n, double g)
30    {
31      num = n;
32      gas = g;
33      System.out.println("生產了車號為" + num + "，汽油量為" + gas + "的車子");
34    }
35
36    public void show()
37    {
38      System.out.println("車號是" + num);
39      System.out.println("汽油量是" + gas);
40    }
41  }
```

實作介面

類別2

```
43  class Plane implements iVehicle
44  {
45    private int flight;
46
47    public Plane(int f)
48    {
49      flight = f;
50      System.out.println("生產了" + flight + "班次的飛機");
51    }
52
53    public void show()
54    {
55      System.out.println("飛機的班次是" + flight);
56    }
57  }
```

定義抽象方法的處理內容

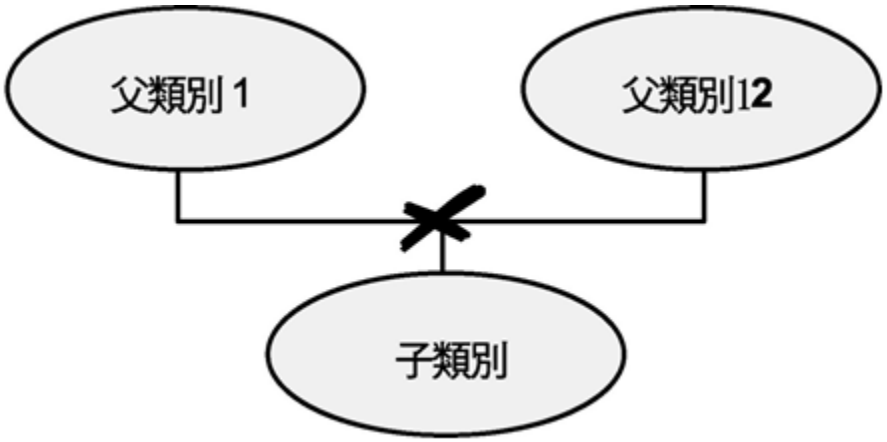
多重繼承(1/3)

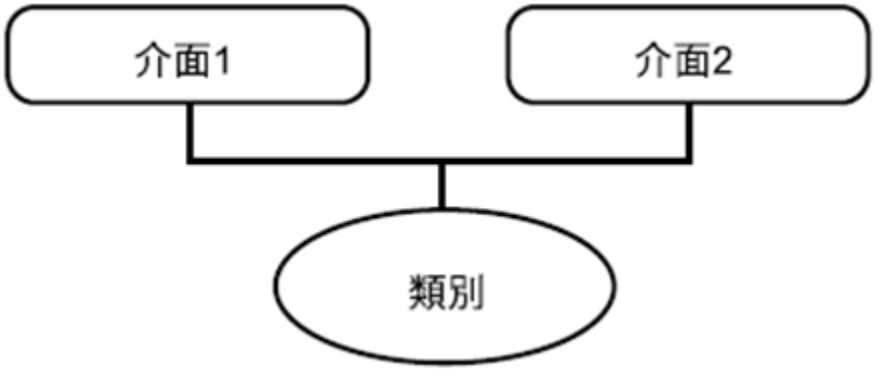
雖然Java並不直接支援多重繼承(多個父類別的繼承)，但是透過實作2個以上介面的方式，仍然可以達成「多重繼承」方法名稱的目的。

語法

```
class 類別名稱 implements 介面名稱1, … , 介面名稱2
{
  …
}
```

在實作(implements)介面的同時還是可以在延伸(extends)生類別





多重繼承(2/3)

Sample4_13.java

```
1  public class Sample4_13
2  {
3      public static void main(String[] args)
4      {
5          Car car1 = new Car(1234, 20.5);
6          car1.vShow();
7          car1.mShow();
8      }
9  }
```

Console Markers Properties

<terminated> Sample4_13 [Java Applicat

```
生產了車號為1234，汽油量為20.5的車子
車號是1234
汽油量是20.5
車子的材質是鐵
```

輸出結果

此頁範例接續下一頁

多重繼承(3/3)

介面1

```
11  interface iVehicle
12  {
13    void vShow();
14  }
```

這是iVehicle的抽象方法

介面2

```
16  interface iMaterial
17  {
18    void mShow();
19  }
```

這是iMaterial的抽象方法

類別

可以實作兩個以上的介面

```
21  class Car implements iVehicle,iMaterial
22  {
23    private int num;
24    private double gas;
25
26    public Car(int n, double g)
27    {
28      num = n;
29      gas = g;
30      System.out.println("生產了車號為" + num + "，汽油量為" + gas + "的車子");
31    }
32
33    public void vShow()
34    {
35      System.out.println("車號是" + num);
36      System.out.println("汽油量是" + gas);
37    }
38
39    public void mShow()
40    {
41      System.out.println("車子的材質是鐵");
42    }
43  }
```

定義iVehicle的方法

定義iMaterial的方法

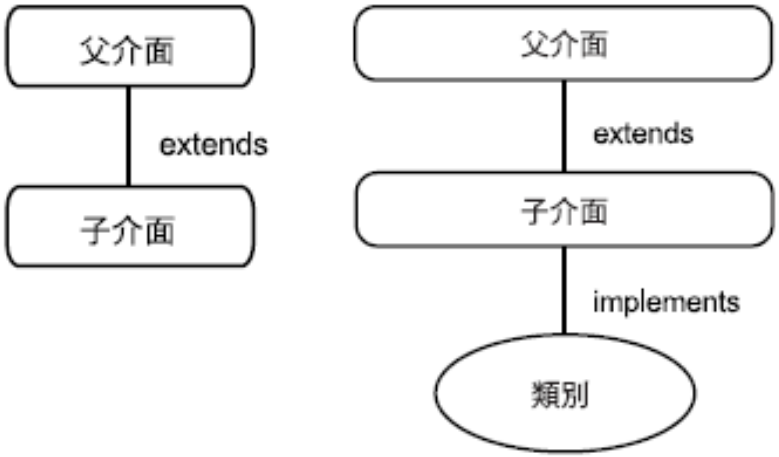
介面的延伸

介面和一般類別一樣，可以透過延伸(extends)的方式產生新的介面。原來的介面稱為父介面(superinterface)，衍生出來的稱為子介面(subinterface)，和先前的經驗一樣，介面的延伸也是要透過關鍵字**extends**。

語法

```
interface 子介面名稱 extends 父介面1,父介面2
{
        ...
}
```

類別只能延伸(extends)類別，介面也只能延伸(extends)介面




```
//會動的東西的介面
interface iMovable
{
  ...
}

//交通工具介面
interface iVehicle extends iMovable
{
  ...
}

//汽車類別
class Car implements iVehicle
{
  ...
}
```

介面 vs 抽象類別(1/2)

- ✓ **抽象類別:**是用來定義相同種類的物件，所需要具備的共通特性；例如動物(Animal)，是所有動物的基底，所有動物具有吃、喝、睡覺等等共同的特性，但是，每種動物吃的東西不一樣(人類屬雜食，老虎只吃肉...)，所以我們可以將這些基本特性，寫成抽象類別與方法，讓其他類別來繼承並且實做方法，這就就抽象類別的意義。
- ✓ **介面:**是用來定義不同種類的物件中，針對某種特性，所需要具備的相同功能；我們說，飛機(airplane)會飛，鳥(bird)也會飛，這兩個物件是不同種類的，可是他們都具有會飛的特性，差別在於飛行的方式不同，一個靠機械完成，一個則是拍動翅膀，所以，我們可以將飛行(Fly)寫成介面，物件只要繼承這個介面並且方法，該物件就具有飛行的特性了。

出處:http://www.dotblogs.com.tw/atowngit/archive/2009/08/26/10253.aspx

介面 vs 抽象類別(2/2)

✓ 下表說明了介面和抽象類別用法的比較。

描述	介面	抽象類別	備註
不能建立物件	○	○	
有建構式	×	○	
可以多重繼承	○	×	
必須實作所宣告之抽象方法	○	○	
方法皆為抽象方法	○	×	抽象類別有一般方法和抽象方法
欄位皆為常數	○	×	抽象類別則比照一般類別