CHAPTER 3

類別的基礎知識(ch8)

類別的功能(ch9)

類別的應用方式(ch10)

本節介紹

- 類別的基本介紹
- 建立新物件及存取成員資料
- 物件的欄位與方法
- 引數及傳回值

關鍵詞彙

- 類別
- 資料成員
- 物件
- 欄位
- 方法
- 傳回值
- 引數

類別的基本介紹(1/2)

Java語言的核心就是類別（使用class保留字），類別用來定義物件導向程式設計裡頭的「**物件**」（object）的格式。這些物件形成了程式裡基本的建構區塊。

語法

```
class 類別名稱
{
  資料型態 欄位名稱 ;
  …
  傳回值的資料型態 方法名稱(參數清單)
  {
     程式敘述 ;
     …
     return 運算式;
  }
  …
};
```

```
//汽車類別
class 汽車
{
   車號 ;
   汽油量 ;
   換車牌的功能…
   加油的功能…
   顯示車號及汽油量
}
```

匯整車子當時的狀態

匯整車子的功能

類別的基本介紹(2/2)

類別當中可以包含兩種項目：

✓ **欄位**（field）：

欄位也稱為類別變數（class variable）、資料成員（data member）、屬性（attribute）等，欄位是儲存資料，提供給本類別的方法處理。

✓ **方法**（method）：

在建立類別的過程中，每個物件都擁有不同的功能，匯集有關其類別的各種「功能」，所組成的架構則稱為「方法」，通常操作的對象為本類別的欄位。

建立新物件及存取成員資料(1/4)

類別中簡單的欄位宣告實體化再加上必要的實體內容可以建立一個該類別型態的物件，也就是說該物件具備該類別所有的屬性及方法。

語法

```
類別名稱 變數名稱 ;
變數名稱 = new 類別名稱();
```

=

語法

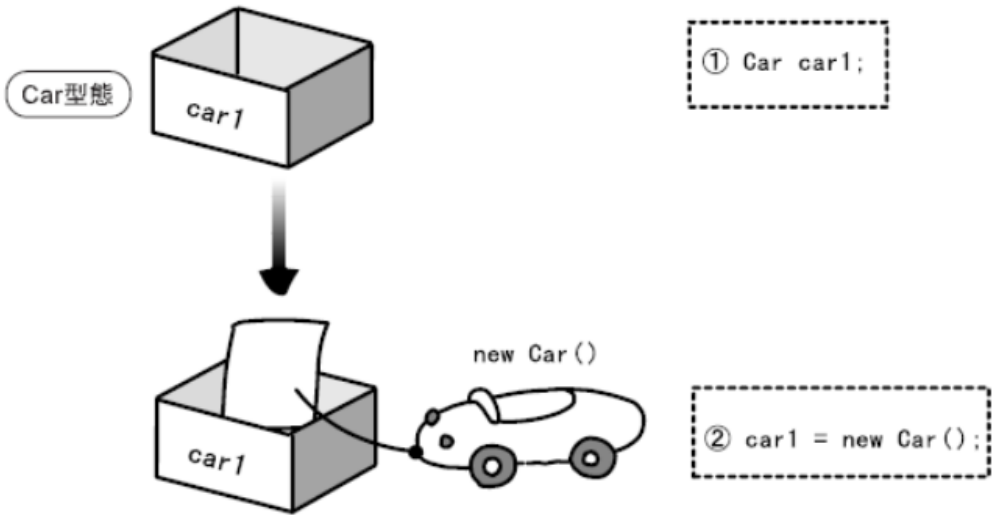
```
類別名稱 變數名稱 = new 類別名稱();
```

Sample3_1.java

```
1
2  public class Sample3_1
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8    }
9  }
10
11 class Car
12 {
13   //Car類別
14 }
```

宣告car1為Car類別的變數

建立Car物件指定給變數car1



建立新物件及存取成員資料(2/4)

新物件建立完成後，就可以對物件的成員進行存取的動作，包括將特定數值指定給欄位儲存。

Sample3_2.java

```
1
2  public class Sample3_2
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      car1.num = 1234;
10     car1.gas = 20.5;
11
12     System.out.println("車號是" + car1.num );
13     System.out.println("汽油量是" + car1.gas );
14   }
15 }
16
17 //Car類別
18 class Car
19 {
20   int num;
21   double gas;
22 }
23
```

建立新物件

指定車號

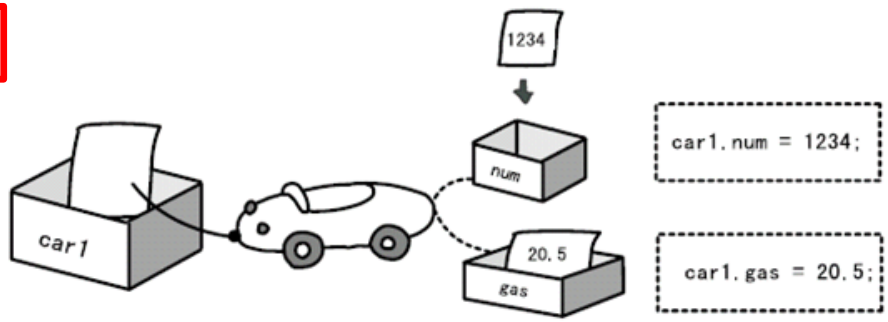
指定汽油量

Console Markers

<terminated> Sample3_2

車號是1234
汽油量是20.5

輸出結果



建立新物件及存取成員資料(3/4)

透過這種方法建立新物件，要有幾個就可以建立幾個。

Sample3_3.java

```
1
2  public class Sample3_3
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      car1.num = 1234;
10     car1.gas = 20.5;
11
12     Car car2;
13     car2 = new Car();
14
15     car2.num = 2345;
16     car2.gas = 30.5;
17
18     System.out.println("car1車號是" + car1.num );
19     System.out.println("car1汽油量是" + car1.gas );
20
21     System.out.println("car2車號是" + car2.num );
22     System.out.println("car2汽油量是" + car2.gas );
23   }
24 }
25
26 //Car類別
27 class Car
28 {
29   int num;
30   double gas;
31 }
32
```

建立新物件1

指定車號和汽油量給物件1

建立新物件2

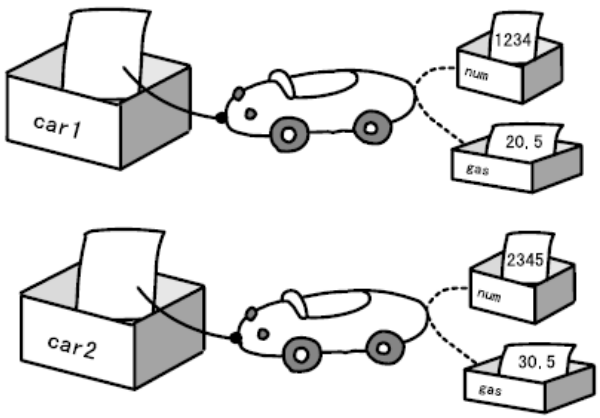
指定車號和汽油量給物件2

Console Markers

<terminated> Sample3_2

car1車號是1234
car1汽油量是20.5
car2車號是2345
car2汽油量是30.5

輸出結果



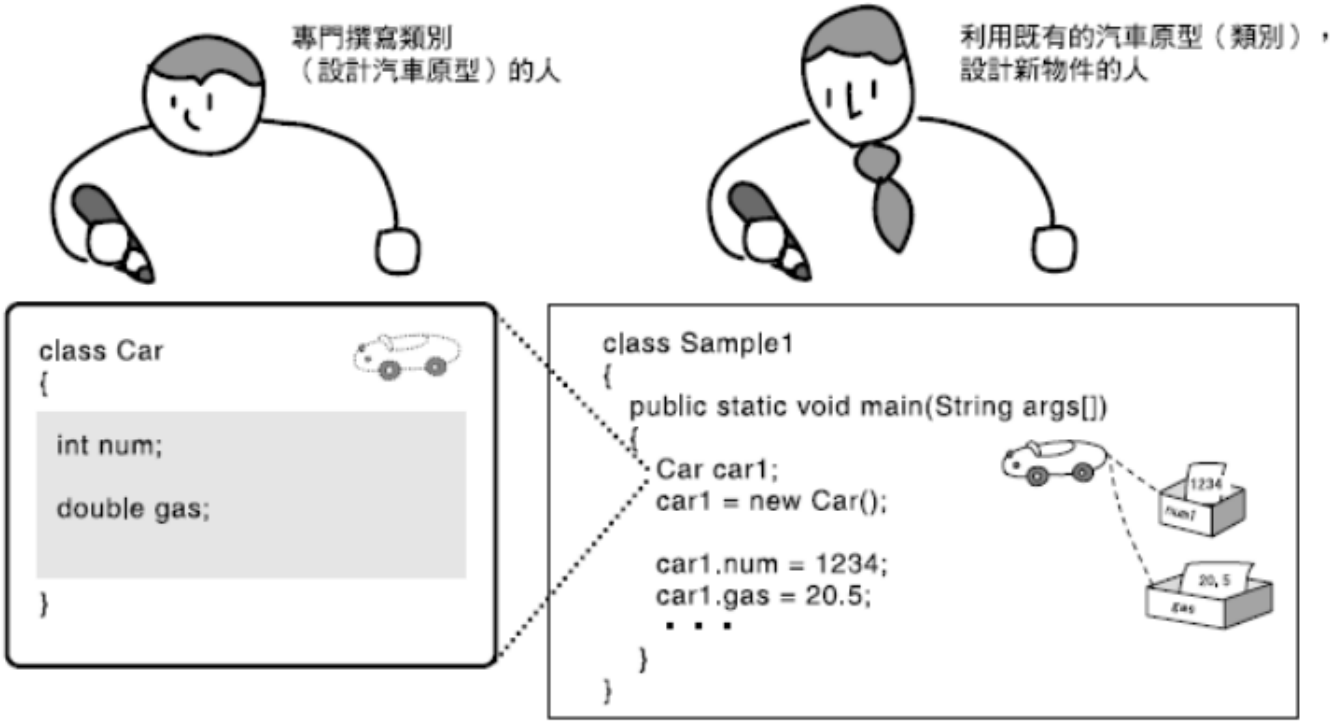
建立新物件及存取成員資料(4/4)

- ✓ 步驟1: 宣告類別.

  「宣告類別」的工作，可以想像成設計一台車的子的原始雛形(類別)。

- ✓ 步驟2: 利用步驟1宣告的類別建立新物件

  建立新物件是根據既有的車子原始雛形的基礎，分別建立新的車子物件，並透過指定的方式讓新物件記得相關的數據(如:車號和汽油量)。



物件的欄位與方法(1/4)

1. 如果是在類別宣告的外面存取欄位時，必須使用指向物件的變數名稱.欄位名稱的存取方式。

```
2  public class Sample3_4
3  {
4     public static void main(String[] args)
5     {
6        Car car1;
7        car1 = new Car();
8
9        car1.num = 1234;
10       car1.gas = 20.5;
11    }
12 }
```

物件名稱. 欄位名稱

2. 如果是在類別宣告內存取欄位時，除了直接使用欄位名稱之外，前面加上「this.」也是合法的存取欄位方式。

```
17 //Car類別
18 class Car
19 {
20    int num;
21    double gas;
22
23    void show()
24    {
25       System.out.println("車號是" + num );
26       System.out.println("汽油量是" + gas );
27    }
28 }
```

欄位名稱

=

```
17 //Car類別
18 class Car
19 {
20    int num;
21    double gas;
22
23    void show()
24    {
25       System.out.println("車號是" + this.num );
26       System.out.println("汽油量是" + this.gas );
27    }
28 }
```

this. 欄位名稱

物件的欄位與方法(2/4)

在建立類別的過程中，每個物件都擁有不同的功能，匯集有關其類別的各種「功能」，所組成的架構則稱為「方法」(method)。

語法

```
傳回值的資料型態   方法名稱(參數…)
  {
      程式敘述句 ;
      …
      return 運算式 ;
  }
```

```
26 //Car類別
27 class Car
28 {
29   int num;
30   double gas;
31
32   void show()
33   {
34     System.out.println("車號是" + num );
35     System.out.println("汽油量是" + gas );
36   }
37 }
```

void表示無傳回值
所以沒有return

定義方法

物件的欄位與方法(3/4)

在類別中定義好方法後，利用該類別建立新的物件，實際進行方法的處理，此稱為「呼叫方法」。

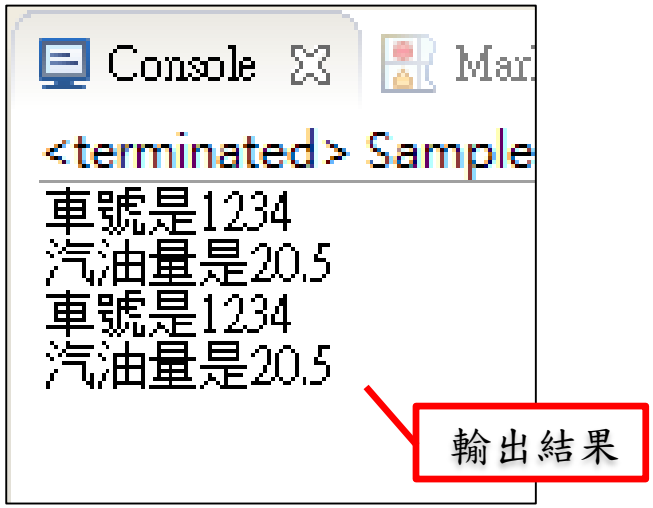
Sample3_4.java

```
1
2  public class Sample3_4
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      car1.num = 1234;
10     car1.gas = 20.5;
11
12     car1.show();
13     car1.show();
14   }
15 }
16
17 //Car類別
18 class Car
19 {
20   int num;
21   double gas;
22
23   void show()
24   {
25     System.out.println("車號是" + num );
26     System.out.println("汽油量是" + gas );
27   }
28 }
```

呼叫方法

再一次呼叫方法

定義方法



物件的欄位與方法(4/4)

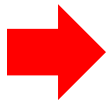
我們可以在類別本身的方法呼叫類別內其他方法。

Sample3_5.java

```
1
2  public class Sample3_5
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      car1.num = 1234;
10     car1.gas = 20.5;
11
12     car1.showCar();
13   }
14 }
15
16 //Car類別
17 class Car
18 {
19   int num;
20   double gas;
21
22   void show()
23   {
24     System.out.println("車號是" + this.num );
25     System.out.println("汽油量是" + this.gas );
26   }
27
28   void showCar()
29   {
30     System.out.println("開始顯示車子資料");
31     this.show();
32   }
33 }
```

this可不加

呼叫物件本身的方法



Console

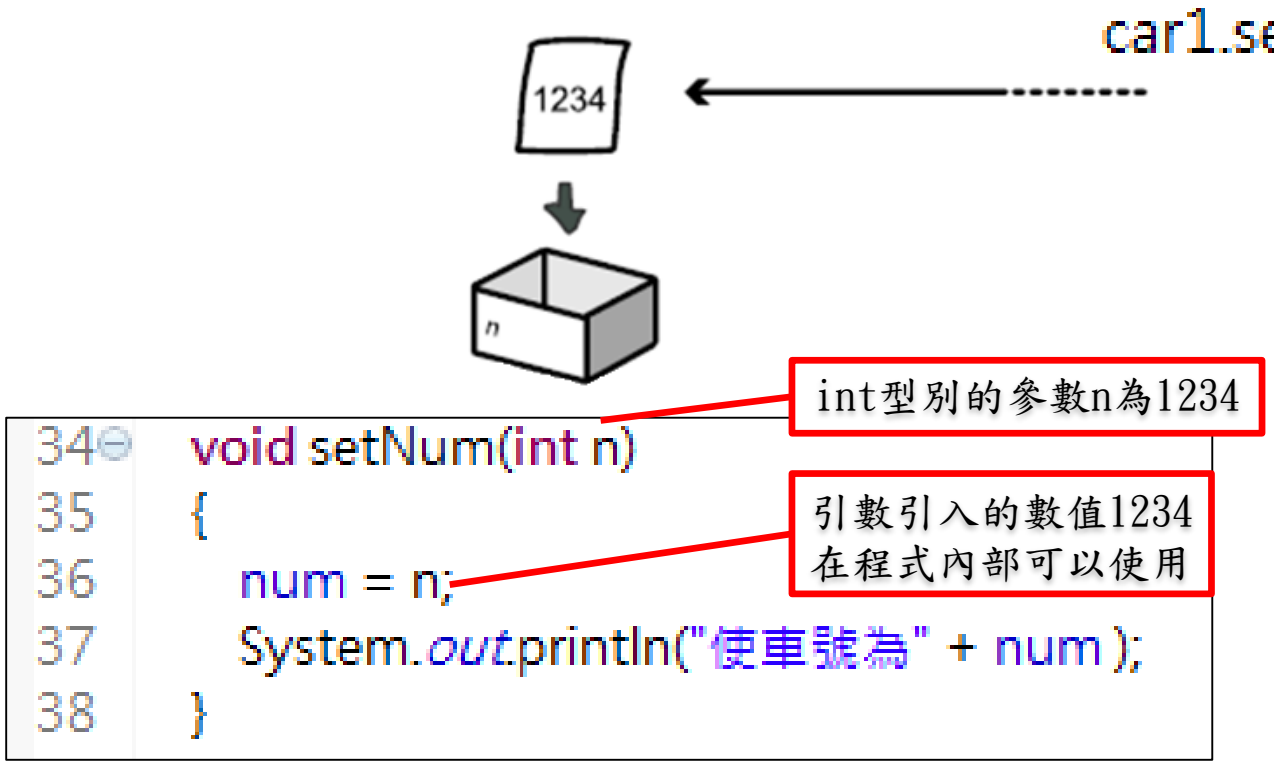
<terminated> Sample

開始顯示車子資料
車號是1234
汽油量是20.5

輸出結果

引數及傳回值(1/5)

呼叫方法的主程式可以在括號內加上引數，並傳遞給方法進行後續處理，使用引數的時候，必須在括號()內指定引數的資料型態和變數名稱(通常是數值或一般資料)。



引數及傳回值(2/5)

Sample3_6.java

```
1
2  public class Sample3_6
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      car1.setNum(1234);
10     car1.setGas(20.5);
11   }
12 }
13 // Car類別
14 class Car
15 {
16   int num;
17   double gas;
18
19   void setNum(int n)
20   {
21     num = n;
22     System.out.println("將車號設為" + num);
23   }
24
25   void setGas(double g)
26   {
27     gas = g;
28     System.out.println("將汽油量設為" + gas);
29   }
30 }
```

引數亦可傳遞變數值例如：

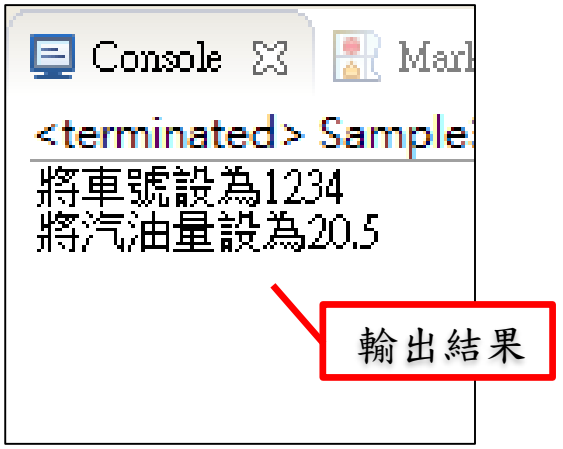
```
int num = 1234;
car1.setNum(num);
```

傳遞引數1234來呼叫setNum方法

傳遞引數20.5來呼叫setGas方法

接收int型別引數的參數n

接收double型別引數的參數g



引數及傳回值(3/5)

我們也可以一次傳遞多個引數。

Sample3_7.java

```
1
2  public class Sample3_7
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      int number = 1234;
10     double gasoline = 20.5;
11
12     car1.setNumGas(number, gasoline);
13   }
14 }
15
16 // Car類別
17 class Car
18 {
19   int num;
20   double gas;
21
22   void setNumGas(int n, double g)
23   {
24     num = n;
25     gas = g;
26     System.out.println("將車號設為" + num + "，汽油量設為" + gas);
27   }
28
29   void show()
30   {
31     System.out.println("車號是" + num);
32     System.out.println("汽油量是" + gas);
33   }
34 }
```

傳遞2個引數的方法

這個方法擁有2個參數

Console Markers Prope

<terminated> Sample3_6 [Java App

將車號設為1234，汽油量設為20.5

輸出結果

引數及傳回值(4/5)

透過適當的設定，讓方法傳回特定值給當初呼叫端的程式，這時候傳回來的值稱為**傳回值(return value)**。對方法來說，只能有一個傳回值被傳送到原來的呼叫程式，雖然前面曾經提到過方法的參數可以有很多個，不過**傳回值只能有一個**。

語法

```
傳回值的型態 方法名稱（參數列表）
{
    運算式;
   …
    return 運算式;
}
```

引數及傳回值(5/5)

Sample3_8.java

```
1  public class Sample3_8
2  {
3     public static void main(String[] args)
4     {
5        Car car1;
6        car1 = new Car();
7        car1.setNumGas(1234, 20.5);
8        int number = car1.getNum();
9        double gasoline = car1.getGas();
10       System.out.println("調查樣品車時得知");
11       System.out.println("車號是" + number + "，汽油量是" + gasoline);
12    }
13 }
14 // Car類別
15 class Car
16 {
17    int num;
18    double gas;
19    int getNum()
20    {
21       System.out.println("調查車號");
22       return num;
23    }
24    double getGas()
25    {
26       System.out.println("調查汽油量");
27       return gas;
28    }
29    void setNumGas(int n, double g)
30    {
31       num = n;
32       gas = g;
33       System.out.println("將車號設為" + num + "，汽油量設為" + gas);
34    }
35 }
```

呼叫具有傳回值的getNum方法，並將傳回值指定給number

呼叫具有傳回值的getGas方法，並將傳回值指定給gasoline

有回傳值的方法，回傳型別為int

有回傳值的方法，回傳型別為double

無回傳值的方法

Console | Markers | Prope

<terminated> Sample3_8 [Java App

```
將車號設為1234，汽油量設為20.5
調查車號
調查汽油量
調查樣品車時得知
車號是1234，汽油量是20.5
```

輸出結果

本節介紹

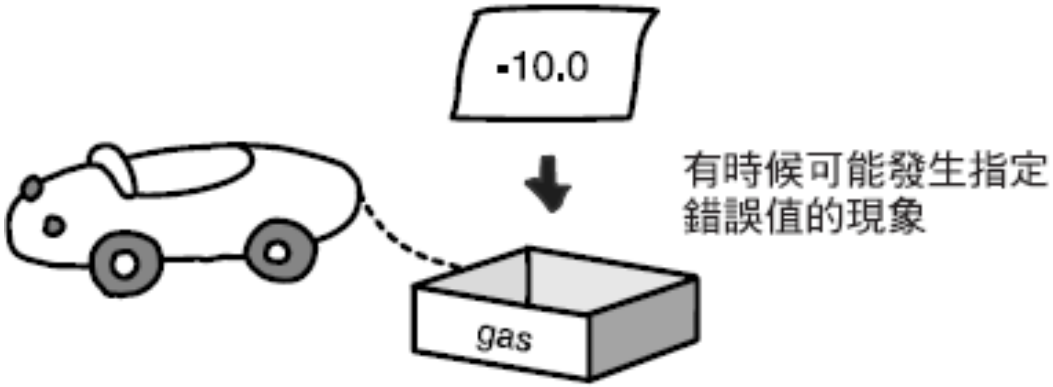
- 成員存取限制
- 封裝(encapsulation)
- 多載(overloading)
- 建構式(constructor)
- 實體變數和實體方法
- 類別變數和類別方法

關鍵詞彙

- 成員
- 修飾子
- 封裝
- 多載
- 建構式
- 實體變數
- 實體方法
- 類別變數
- 類別方法

成員存取限制(1/4)

成員(member)的資料被毫無限制的存取，任誰都可以指定任意值給成員，Java 語言為了防止這種現象的產生，規定：有一種成員的資料不能任由類別外部的任何人隨意存取。



成員存取限制(2/4)

如果沒有做存取限制，就有可能讓外部的任何人隨意存取資料。

Sample3_9.java

```
1  public class Sample3_9
2  {
3    public static void main(String[] args)
4    {
5      Car car1;
6      car1 = new Car();
7
8      car1.num = 1234;
9      car1.gas = -10;
10
11     car1.show();
12   }
13 }
14 // Car類別
15 class Car
16 {
17   int num;
18   double gas;
19
20   void show()
21   {
22     System.out.println("車號是"+this.num);
23     System.out.println("汽油量是"+this.gas);
24   }
25 }
```

指定錯誤的汽油量資料

Console

<terminated> Sample

車號是1234
汽油量是-10.0

輸出結果

成員存取限制(3/4)

為了避免前頁所說的情況發生，所以出現了下面三種修飾子(modifier)。

✓ 私有成員(private member)

- 宣告成員的時候如果在前面加上 **private** 字樣，就無法從類別以外的地方存取到類別內部的成員資料，具有這種特性的成員稱為私有成員(private member)。

✓ 公有成員(public member)

- 宣告成員的時候如果在前面加上 **public** 字樣，表示可以從類別以外的地方存取內部的成員資料，具有這種特性的成員稱為公用成員(public member)。

✓ 保護成員 (protected member)

- 宣告成員的時候如果在前面加上 **protected** 字樣，表示只有子類別可以存取父類別內部的成員資料，具有這種特性的成員稱為保護成員(protected member)。此方法我們會在第四章在做介紹。

成員存取限制(4/4)

```java
public class Sample3_10
{
  public static void main(String[] args)
  {
    Car car1;
    car1 = new Car();

    car1.setNumGas(1234, 20.5);
    car1.show();

    System.out.println("指定不正確的汽油輛(-10.0)看看");

    car1.setNumGas(1234, -10.0);
    car1.show();
  }
}
// Car類別
class Car
{
  private int num;
  private double gas;

  public void setNumGas(int n, double g)
  {
    if (g>0 && g<100)
    {
      num = n;
      gas = g;
      System.out.println("將車號設為"+num+"，汽油量設為"+gas);
    }
    else
    {
      System.out.println(g+"不是正確的汽油量");
      System.out.println("無法變更汽油量");
    }
  }

  public void show()
  {
    System.out.println("車號是"+this.num);
    System.out.println("汽油量是"+this.gas);
  }
}
```

因為Car類別的欄位num & gas都是宣告private所以無法直接存取，例如：car1.num =1234;
car1.gas = 20.5;
都無法使用。

將欄位設為private使外部無法作直接存取

判斷汽油值要在此條件範圍才能更改

雖然無法對欄位作直接存取，我們還是可以透過設為public的方法進行存取，並且我們可以在方法內增加判斷來排除錯誤的設定。

```
<terminated> Sample3_9 [Java App
將車號設為1234，汽油量設為20.5
車號是1234
汽油量是20.5
指定不正確的汽油輛(-10.0)看看
-10.0不是正確的汽油量
無法變更汽油量
車號是1234
汽油量是20.5
```

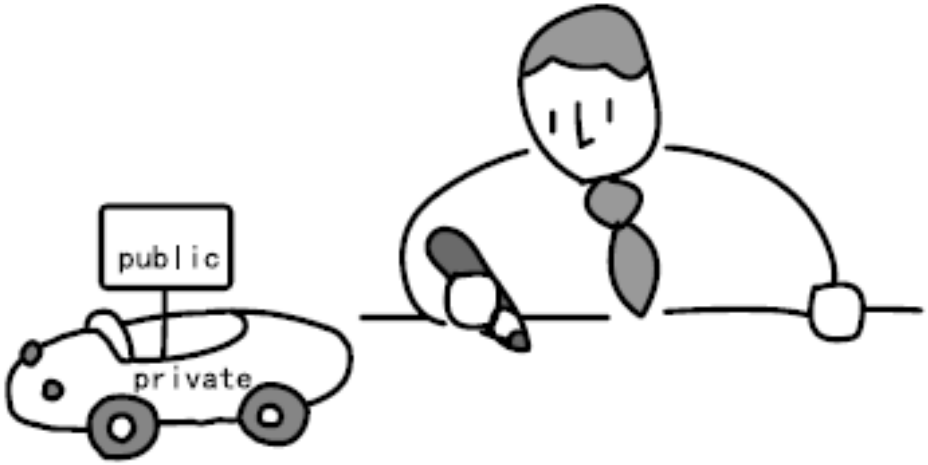
輸出結果

小知識

一旦類別的成員前的修飾子被省略掉時，表示這個成員只能存在於它們所宣告的套件(package)中。有關套件的部分我們將會在第五章的設計大型程式篇談到。

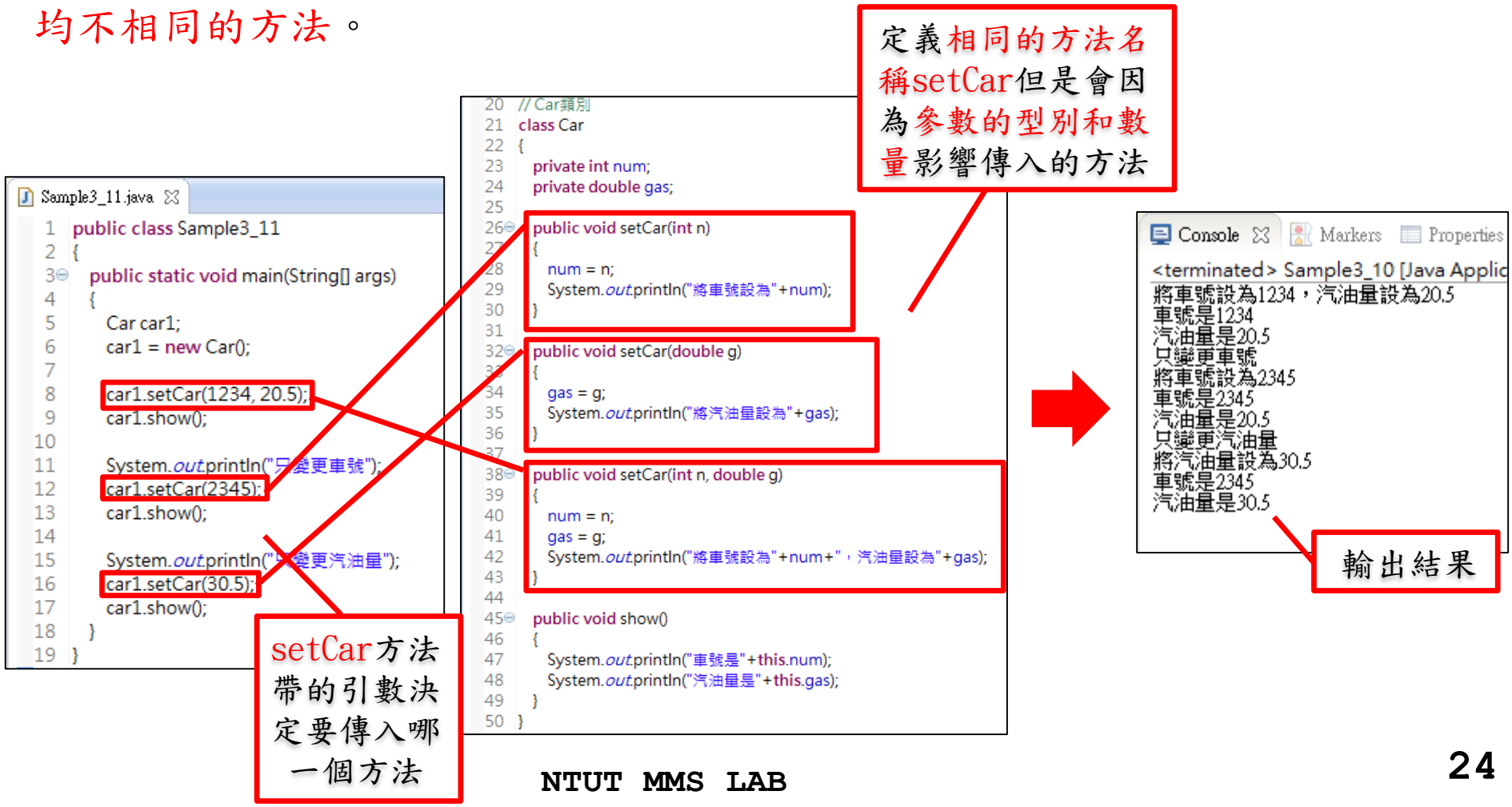
封裝(encapsulation)

Java的類別也提供封裝(encapsulation)的功能。也就是說將類別內的私有成員(private member)隱藏起來，避免不必要的外在存取，僅提供必要的公共成員(public member)供他人呼叫使用，如此一來其他程式設計師即使不知道class內部的運作方式，也不會因為誤用class而產生錯誤的現象。前一頁的範例就是就是一種封裝的例子。



多載(overloading)

在同一個類別內，可以同時定義數個名稱相同，但是參數數目和資料型態均不相同的方法。



建構式(constructor) (1/4)

定義建構式(constructor)的定義和方法非常相似，但建構式的名稱一定要和類別名稱相同。另外，與方法不同的是，建構式並不需要有任何傳回值。當我們透過某個既有的類別建立物件時，會自動執行該建構式內容，我們可以在建構式設定物件成員初始值。

語法

```
修飾子 類別名稱(引數清單)
{
   程式敘述 ;
   …
}
```

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

}
```

建構式(constructor)(2/4)

Sample3_12.java

```
1  public class Sample3_12
2  {
3     public static void main(String[] args)
4     {
5        Car car1;
6        car1 = new Car();
7
8        car1.show();
9     }
10 }
11 // Car類別
12 class Car
13 {
14    private int num;
15    private 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 show()
25    {
26       System.out.println("車號是"+num);
27       System.out.println("汽油輛是"+gas);
28    }
29 }
```

建立新物件就會呼叫建構式

建構式名稱與類別名稱相同

被呼叫的建構式通常用來作初始化設定

Console | Marker

<terminated> Sample3_

生產了車子
車號是0
汽油輛是0.0

輸出結果

建構式(constructor) (3/4)

建構元也有多載化(Overloading)的功能，可在不同的狀況下就完成物件初始值的設定工作。

Sample3_13.java

```
1  public class Sample3_13
2  {
3     public static void main(String[] args)
4     {
5        Car car1;
6        car1 = new Car();
7        car1.show();
8
9        Car car2;
10       car2 = new Car(1234, 25.0);
11       car2.show();
12    }
13 }
14 // Car類別
15 class Car
16 {
17    private int num;
18    private 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 Car(int n, double g)
28    {
29       num = n;
30       gas = g;
31       System.out.println("生產了車號為"+num+"，汽油量為"+gas+"的車子");
32    }
33
34    public void show()
35    {
36       System.out.println("車號是"+num);
37       System.out.println("汽油輛是"+gas);
38    }
39 }
```

呼叫沒有參數的建構式

呼叫兩個參數的建構式

如果修飾子宣告成private則此物件就無法透過此建構式建立

沒有參數的建構式

兩個參數的建構式

Console | Markers | Properties

<terminated> Sample3_13 [Java Applicatio

```
生產了車子
車號是0
汽油輛是0.0
生產了車號為1234，汽油量為25.0的車子
車號是1234
汽油輛是25.0
```

輸出結果

建構式(constructor) (4/4)

在建構式內也可以呼叫其他建構式。

Sample3_14.java

```
1  public class Sample3_14
2  {
3     public static void main(String[] args)
4     {
5        Car car1;
6        car1 = new Car();
7        car1.show();
8
9        Car car2;
10       car2 = new Car(1234, 25.0);
11       car2.show();
12    }
13 }
14 // Car類別
15 class Car
16 {
17    private int num;
18    private 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 Car(int n, double g)
28    {
29       this();
30       num = n;
31       gas = g;
32       System.out.println("生產了車號為"+num+"，汽油量為"+gas+"的車子");
33    }
34
35    public void show()
36    {
37       System.out.println("車號是"+num);
38       System.out.println("汽油輛是"+gas);
39    }
40 }
```

也可以在此建構式呼叫兩個參數的建構式 this(1234,25.0);
但是要注意不能形成遞迴!

呼叫沒有帶參數的建構式(this一定要寫在建構式第一行)

this()呼叫沒有帶參數的建構式產生的結果

Console | Markers | Properties

```
<terminated> Sample3_14 [Java Applicat
生產了車子
車號是0
汽油輛是0.0
生產了車子
生產了車號為1234，汽油量為25.0的車子
車號是1234
汽油輛是25.0
```

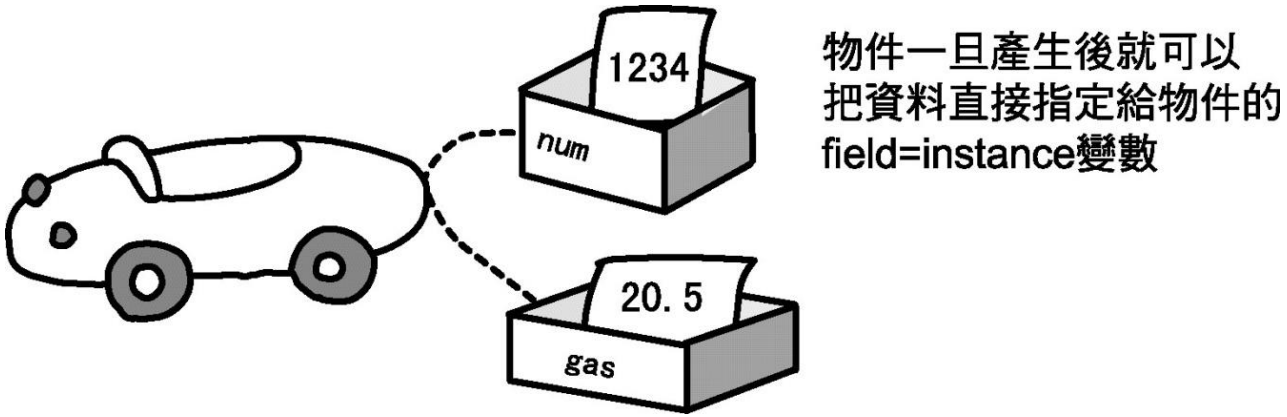
輸出結果

小知識

之前還沒提到建構式時，我們都沒使用到，所以代表說建構式是可以省略的，我們沒建構式時，Java會自動建立一個無參數的建構式，又稱為預設建構式(default constructor)

實體變數和實體方法

與物件關係密切的欄位被稱為實體變數(instance variable)、與物件關係密切的方法被稱為實體方法(instance method) 。只要是實體變數或實體方法就可以在物件產生之後，立刻透過物件呼叫實體變數或實體方法，並傳值給它們。前面提到的Car類別的範例全都是使用實體變數和實體方法。



類別變數和類別方法(1/2)

對類別(class)來說，和物件沒關連的成員仍可成為類別的一部份，這樣的成員算是和類別整體有關的成員。

✓ 類別變數(class variable)

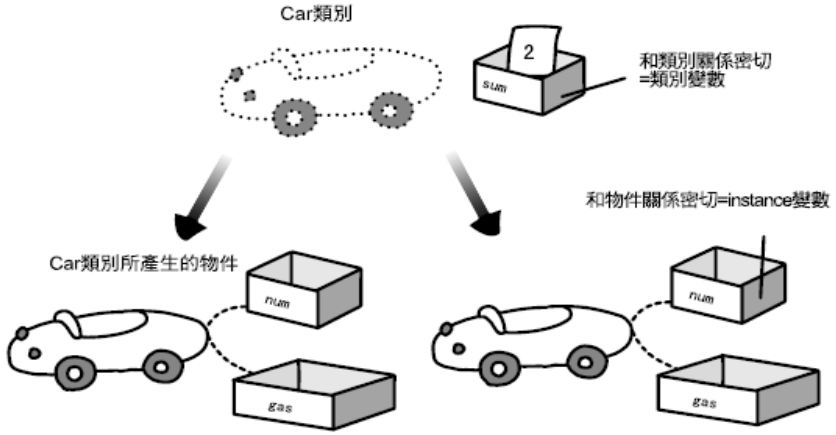
和類別有關的欄位被稱為類別變數(class variable)。

語法

```
class 類別名稱
{
    static 資料型態 類別變數名稱;
    ...
}
```

✓ 類別方法(class method)

和類別有關的方法被稱為類別方法(class method)。



語法

```
static 傳回值資料型態 方法名稱(參數、參數…)
    程式敘述;
    ...
}
```

類別變數和類別方法(2/2)

小知識

我們一值在使用的main方法也是加上static後所形成的一種類別方法。

不需要建立物件即可呼叫類別方法

Sample3_15.java

```
1  public class Sample3_15
2  {
3    public static void main(String[] args)
4    {
5      Car.showSum();
6
7      Car car1;
8      car1 = new Car();
9      car1.setCar(1234, 20.5);
10
11     Car.showSum();
12
13     Car car2;
14     car2 = new Car();
15     car2.setCar(4567, 30.5);
16   }
17 }
```

```
18 // Car類別
19 class Car
20 {
21   public static int sum = 0;
22
23   private int num;
24   private double gas;
25
26   public Car()
27   {
28     num = 0;
29     gas = 0.0;
30     sum++;
31     System.out.println("生產了車子");
32   }
33
34   public void setCar(int n, double g)
35   {
36     num = n;
37     gas = g;
38     System.out.println("將車號設為"+num+"，汽油量設為"+gas);
39   }
40
41   public static void showSum()
42   {
43     System.out.println("車子總共有"+sum+"台");
44   }
45
46   public void show()
47   {
48     System.out.println("車號是"+num);
49     System.out.println("汽油量是"+gas);
50   }
51 }
```

類別變數

類別方法即變數前面會加上一個static修飾子

類別方法

Console | Markers | Properti

<terminated> Sample3_15 [Java App

```
車子總共有0台
生產了車子
將車號設為1234，汽油量設為20.5
車子總共有1台
生產了車子
將車號設為4567，汽油量設為30.5
```

輸出結果

本節介紹

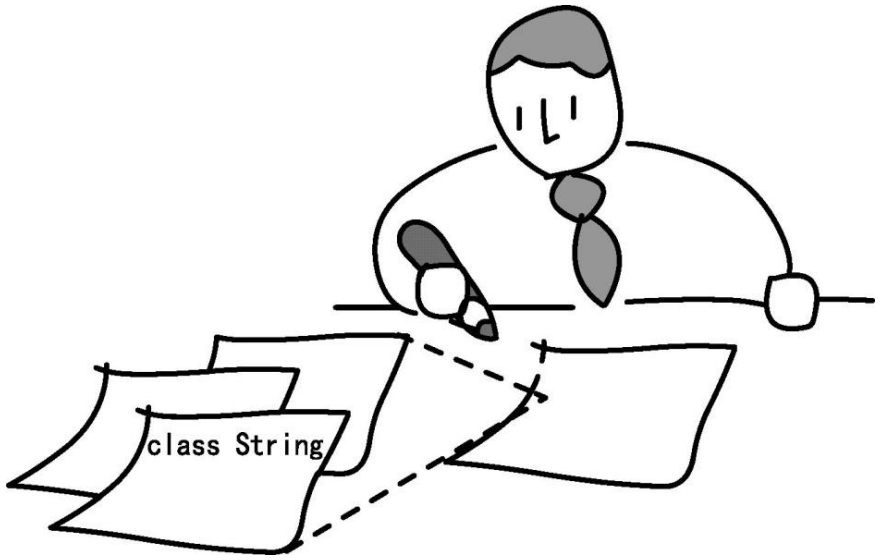
- 類別庫
- 操作字串的類別
- 包裝類別
- Math類別
- 類別型態的變數
- 傳址呼叫(passing by address)
- 傳值呼叫(passing by value)
- 物件的陣列

關鍵詞彙

- 類別庫
- String類別
- 類別型態的變數
- 物件陣列
- 包裝類別
- Math類別
- 傳址呼叫
- 傳值呼叫

類別庫(1/2)

Java語言可以把經常使用到的功能匯集起來形成所謂的類別庫(class library)，而類別庫簡單來說就是許多類別的一個集合體。



類別庫(2/2)

透過類別庫所提供的類別，可更容易的撰寫及達到程式所需要的的行為。

Sample3_16.java

```
1  import java.io.BufferedReader;
4
5  public class Sample3_16
6  {
7    public static void main(String[] args) throws IOException
8    {
9      System.out.println("請輸入一個整數");
10
11     BufferedReader br =
12         new BufferedReader(new InputStreamReader(System.in));
13
14     String str = br.readLine();
15
16     int num = Integer.parseInt(str);
17
18     System.out.println("您輸入的數字是:" + num);
19   }
20 }
21
22
```

框起來的都是類別庫所提供的類別

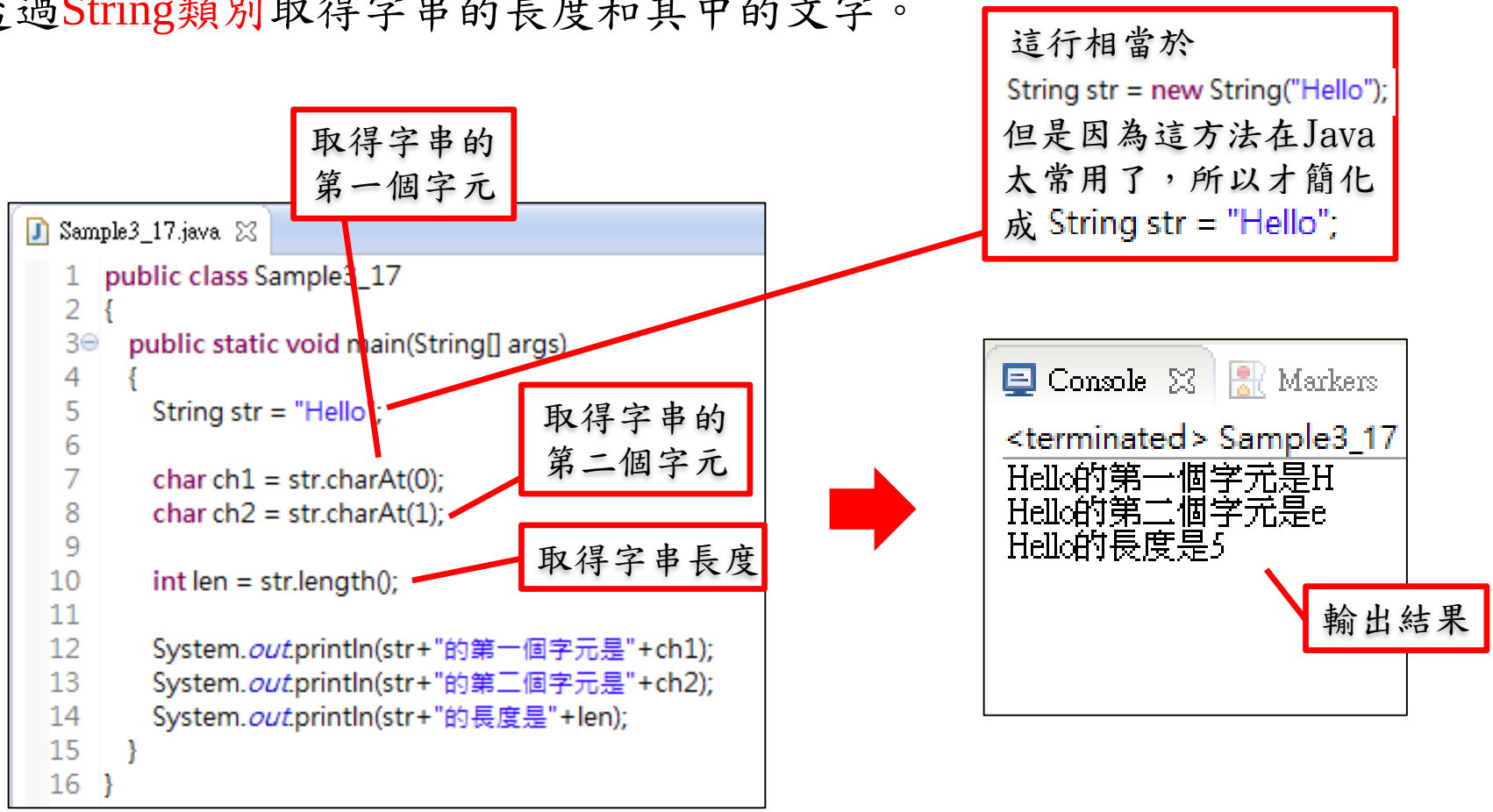
操作字串的類別(1/6)

String類別就是專門用來處理字串的類別。所以像"Hello"和"你好"這種個別的字串，都可以利用String類別加以產生。

方法名稱	傳回型別	功能
charAt(int index)	char	取得某索引值位置的字元
endsWith(String suffix)	boolean	要是該字串結尾與所指定的字尾相同的話，則傳回true
equals(Object anObject)	Boolean	判斷是否為引數的字串
equalsIgnoreCase(String anotherString)	boolean	將字串與另一個物件做比較，但忽略大小寫
indexOf(int ch)	int	根據指定字元找出它第一個出現在字串中的索引位置
indexOf(String str)	int	根據指定字元找出它第一個出現在字串中的索引位置，而且是從指定的索引值開始搜尋
lastIndex(int ch)	int	根據指定字元找出它最後出現在字串中的索引位置
lastIndex(String str)	int	根據指定字元找出它最後出現在字串中的索引位置，而且是從指定的索引值開始搜尋
length()	int	取得字串的長度
Substring(int beginIndex)	String	根據指定的位置索引產生子字串
Substring(int beginIndex, int endIndex)	String	根據指定的位置索引產生子字串，可指定起始和結束的索引位置
startsWith(String prefix)	boolean	測試該字串是否以指定的字首開頭
toLowerCase()	String	將String物件裡面的所有字元轉換為小寫
toUpperCase()	String	將String物件裡面的所有字元轉換為大寫

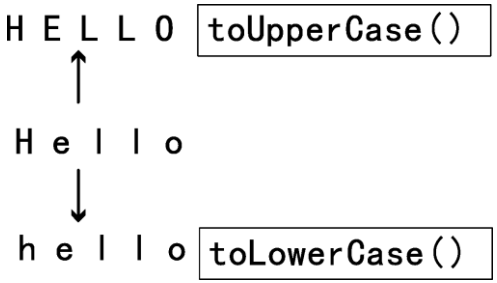
操作字串的類別(2/6)

透過String類別取得字串的長度和其中的文字。



操作字串的類別(3/6)

透過String類別做大寫字母和小寫字母的轉換。

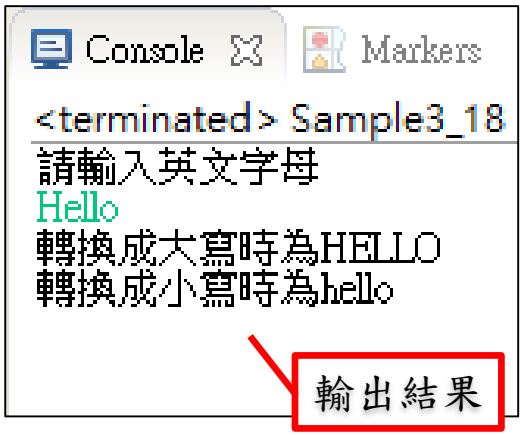


Sample3_18.java

```
1  import java.io.BufferedReader;
4
5  public class Sample3_18
6  {
7    public static void main(String[] args) throws IOException
8    {
9      System.out.println("請輸入英文字母");
10
11     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12
13     String str = br.readLine();
14
15     String stru = str.toUpperCase();
16     String strl = str.toLowerCase();
17
18     System.out.println("轉換成大寫時為"+stru);
19     System.out.println("轉換成小寫時為"+strl);
20   }
21 }
```

將str轉成大寫放到stru

將str轉成小寫放到stri



操作字串的類別(4/6)

透過String類別做搜尋文字的功能。

Sample3_19.java

```
1  import java.io.BufferedReader;
4
5  public class Sample3_19
6  {
7    public static void main(String[] args) throws IOException
8    {
9  System.out.println("請輸入字串");
10
11     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12
13     String str1 = br.readLine();
14
15     System.out.println("請輸入要檢索的文字");
16
17     String str2 = br.readLine();
18     char ch = str2.charAt(0);
19
20     int num = str1.indexOf(ch);
21
22     if (num !=-1)
23       System.out.println(str1+"的第" +(num+1)+"個字就是「"+ch+"」");
24     else
25       System.out.println(str1+"中沒有「"+ch+"」這個字");
26   }
27 }
```

搜尋字串str1內ch字元的位置並回傳位置索引值給num

由於搜尋不到字會回傳-1，所以在這這用if做錯誤處理

Console Markers Prope

<terminated> Sample3_19 [Java Ap

```
請輸入字串
輕鬆學JAVA
請輸入要檢索的文字
學
輕鬆學JAVA的第3個字就是「學」
```

輸出結果

操作字串的類別(5/6)

除了 String 類別是專門用處理字串之外，還有其他類別可以處理字串。如果您要修改字串物件，最好還是透過「StringBuffer 類別」（字串緩衝區類別）。也就是說 StringBuffer 類別除了具有 String 類別的所有功能之外，它還能夠修改裏面的字串。StringBuffer 類別當中的各個 method 在處理字串時，會把整個字串當成參數來傳遞並加以處理。

方法名稱	傳回型別	功能
oppend(char c)	StringBuffer	將字元連接到字串緩衝區的後面
oppend(String str)	StringBuffer	將字串連接到字串緩衝區的後面
deleteCharAt(int index)	StringBuffer	移除字串緩衝區裡面某個指定索引位置的字元
insert(int offset,char c)	StringBuffer	將字元引數插入到字串緩衝區的後面
insert(int offset,String str)	StringBuffer	將字元插入到字串緩衝區的後面
length()	int	取得字串緩衝區的長度(以字元為單位)
replace(int start, int end, String str)	StringBuffer	將字串緩衝區裡面的某個子字串以另一個字串取代之
reverse()	StringBuffer	將字串緩衝區裡面的字串反向排列
setCharAt(int index, char ch)	void	將字串緩衝區裡面某個特定索引位置的字元換成ch
toString()	String	將字串緩衝區裡的資料轉換成字串

操作字串的類別(6/6)

小知識

String類別不能修改字串內容
StringBuffer類別可以修改字串內容

Sample3_20.java

```
1  import java.io.BufferedReader;
4
5  public class Sample3_20
6  {
7    public static void main(String[] args) throws IOException
8    {
9      System.out.println("請輸入字串。");
10
11     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12
13     String str1 = br.readLine();
14
15     System.out.println("請輸入要新增的字串。");
16
17     String str2 = br.readLine();
18
19     StringBuffer sb = new StringBuffer(str1);
20     sb.append(str2);
21
22     System.out.println("在「" + str1 + "」後新增「" + str2 + "」的話，就會變成「" + sb + "」。");
23   }
24 }
```

從類別StringBuffer建立物件sb並傳入字串str1到字串緩衝區

使用類別方法append將字串str2接到字串緩衝區後面

輸出結果

包裝類別

包裝類別(wrapper class)提供一些方法給八大基本型別，所以每個類別都會和基本型別所對應，我們之前常用到Interger包裝類別對應到int基本型別就是一個例子，詳情如下表所示。

包裝類別	基本型別
Byte	byte
Character	char
Short	short
Integer	int
Long	long
Float	float
Double	double
Boolean	boolean

3.3 類別的應用方式

Math類別(1/2)

Math類別提供了一些常用的數學運算式給使用者做關於運算上的操作。

方法名稱	傳回型別	功能
abs(double a)	double	將double型態數值轉換成絕對值
abs(int a)	int	將int型態數值轉換成絕對值
ceil(double a)	double	產生一個最接近、卻又大於double型態引數值的整數，並且以double型態傳回結果
cos(double a)	double	根據引數值產生三角函數cosine值
floor(double a)	double	產生一個最接近、卻又小於double型態引數值的整數，並且以double型態傳回結果
max(double a, double b)	double	比較兩個double型態的引數何者較大
max(int a, int b)	double	比較兩個int型態的引數何者較大
min(double a, double b)	double	比較兩個double型態的引數何者較小
min(int a, int b)	double	比較兩個int型態的引數何者較小
pow(double a, double b)	double	根據a的b次方進行運算
random()	double	在0.0 ~ 0.1的範圍內建立亂數
rint(double a)	int	產生一個最接近double型態的引數
sin(double a)	double	根據引數值產生三角函數sine值
sqrt(double a)	double	根據引數值開平方根
tan(double a)	double	根據引數值產生三角函數tangent值

Math類別(2/2)

Sample3_21.java

```
1  import java.io.BufferedReader;
4
5  public class Sample3_21
6  {
7    public static void main(String[] args) throws IOException
8    {
9      System.out.println("請輸入兩個整數");
10
11     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12
13     String str1 = br.readLine();
14     String str2 = br.readLine();
15
16     int num1 = Integer.parseInt(str1);
17     int num2 = Integer.parseInt(str2);
18
19     int ans = Math.max(num1, num2);
20
21     System.out.println(num1+"與"+num2+"中較大的是"+ans);
22   }
23 }
```

使用Math類別的類別方法max找出num1和num2中的最大值傳入ans

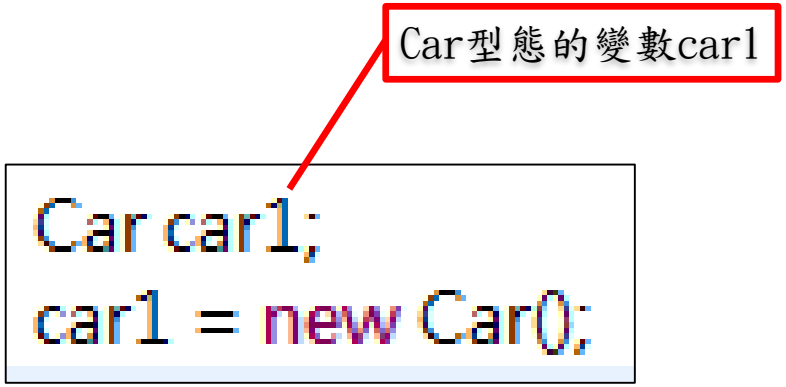
Console Markers

<terminated> Sample3_21
請輸入兩個整數
5
10
5與10中較大的是10

輸出結果

類別型態的變數(1/4)

如下圖所示，car1實際上是一個指向物件Car的變數，因此又稱為Car型態的變數，也是一個標準的類別型態變數。上述步驟的運作過程是：利用new運算子產生一個新物件，然後將物件指定給類別型態變數car1。但事實上這種類別型態變數即使在沒有建立物件的前提下，然可進行**指定**的動作。

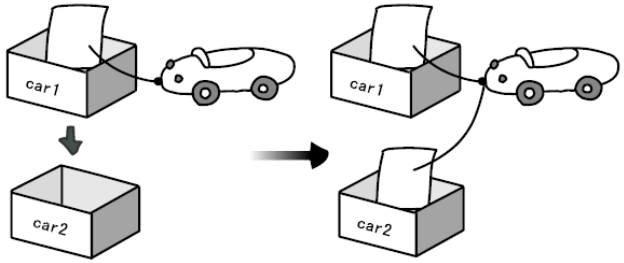


3.3 類別的應用方式

類別型態的變數(2/4)

宣告car1

建立新物件並指定給car1



Sample3_22.java

```
1
2  public class Sample3_22
3  {
4    public static void main(String[] args)
5    {
6      System.out.println("宣告car1");
7      Car car1;
8      car1 = new Car();
9      car1.setCar(1234,20.5);
10
11     System.out.println("宣告car2");
12     Car car2;
13
14     System.out.println("將car1指定給car2");
15     car2 = car1;
16
17     System.out.print("car1的");
18     car1.show();
19
20     System.out.print("car2的");
21     car2.show();
22
23     System.out.println("改變car1的相關資料");
24     car1.setCar(2345, 30.5);
25
26     System.out.print("car1的");
27     car1.show();
28
29     System.out.print("car2的");
30     car2.show();
31   }
32 }
```

```
25 class Car
26 {
27   private int num;
28   private double gas;
29
30   public Car()
31   {
32     num = 0;
33     gas = 0.0;
34     System.out.println("生產了車子");
35   }
36
37   public void setCar(int n, double g)
38   {
39     num = n;
40     gas = g;
41     System.out.println("將車號設為" + num + "，汽油量設為" + gas);
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   }
49 }
```

Console | Markers | Prope

<terminated> Sample3_22 [Java A

```
宣告car1
生產了車子
將車號設為1234，汽油量設為20.5
宣告car2
將car1指定給car2
car1的車號是1234
汽油量是20.5
car2的車號是1234
汽油量是20.5
改變car1的相關資料
將車號設為2345，汽油量設為30.5
car1的車號是2345
汽油量是30.5
car2的車號是2345
汽油量是30.5
```

輸出結果

宣告car2

將car1指定給car2

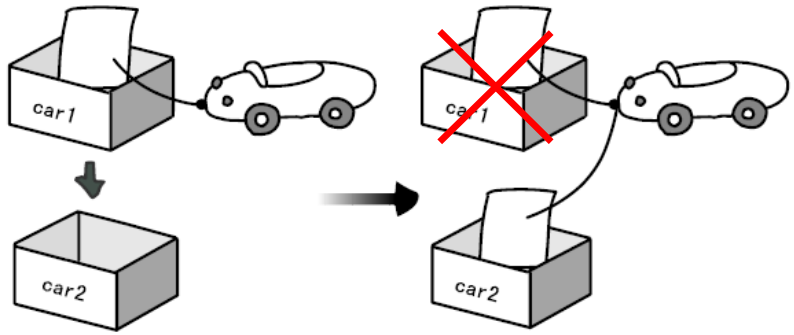
變更car1物件

我們將car1指定給car2後car2會隨著car1改變，這表示說並沒有存在兩個物件，而是car1和car2指向同一個車子物件

類別型態的變數(3/4)

指定 null 後，變數 car1 就不再指向任何一個物件。經過這樣的設定後，某個物件不再被任何物件參考到，則Java會認為該物件已經準備被廢棄，並使用垃圾回收(garbage collection)的程序來處理這樣的工作。

```
4⊖  public static void main(String[] args)
5   {
6       System.out.println("宣告car1");
7       Car car1;
8       car1 = new Car();
9       car1.setCar(1234,20.5);
10
11      System.out.println("宣告car2");
12      Car car2;
13
14      System.out.println("將car1指定給car2");
15      car2 = car1;
16
17      car1 = null;
18
19      System.out.print("car2的");
20      car2.show();
21  }
```



car1 和 car2 指向同一個物件

即使把car1指向null，car2所指向的物件仍然存在

類別型態的變數(4/4)

類別型態的變數也可以像基本型態的變數一樣可以做為方法和建構式的引數。

Sample3_23.java

```
1
2  public class Sample3_23
3  {
4    public static void main(String[] args)
5    {
6      Car car1;
7      car1 = new Car();
8
9      car1.show();
10
11     int number = 1234;
12     double gasoline = 20.5;
13     String str = "1號車";
14
15     car1.setCar(number, gasoline);
16     car1.setName(str);
17
18     car1.show();
19   }
20 }
```

引數str是一個指向字串物件的類別型態變數

```
22 class Car
23 {
24   private int num;
25   private double gas;
26   private String name;
27
28   public Car()
29   {
30     num = 0;
31     gas = 0.0;
32     name = "沒有名稱";
33     System.out.println("生產了車子");
34   }
35
36   public void setCar(int n, double g)
37   {
38     num = n;
39     gas = g;
40     System.out.println("將車號設為" + num + "，汽油量設為" + gas);
41   }
42
43   public void setName(String nm)
44   {
45     name = nm;
46     System.out.println("將車名設為" + name);
47   }
48
49   public void show()
50   {
51     System.out.println("車號是" + num);
52     System.out.println("汽油量是" + gas);
53     System.out.println("車名是" + name);
54   }
55 }
```

使用類別型態變數作為欄位

使用類別變數作為參數

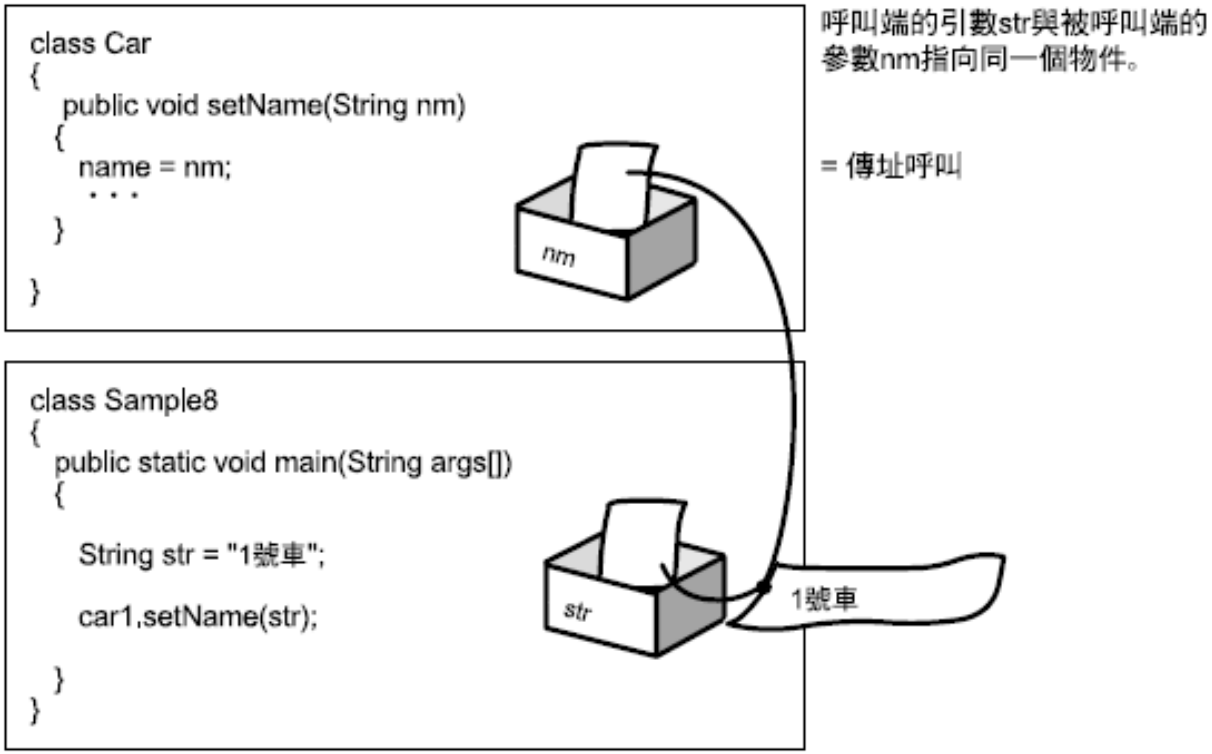
Console | Markers | Prope

```
<terminated> Sample3_23 [Java Ap
生產了車子
車號是0
汽油量是0.0
車名是沒有名稱
將車號設為1234，汽油量設為20.5
將車名設為1號車
車號是1234
汽油量是20.5
車名是1號車
```

輸出結果

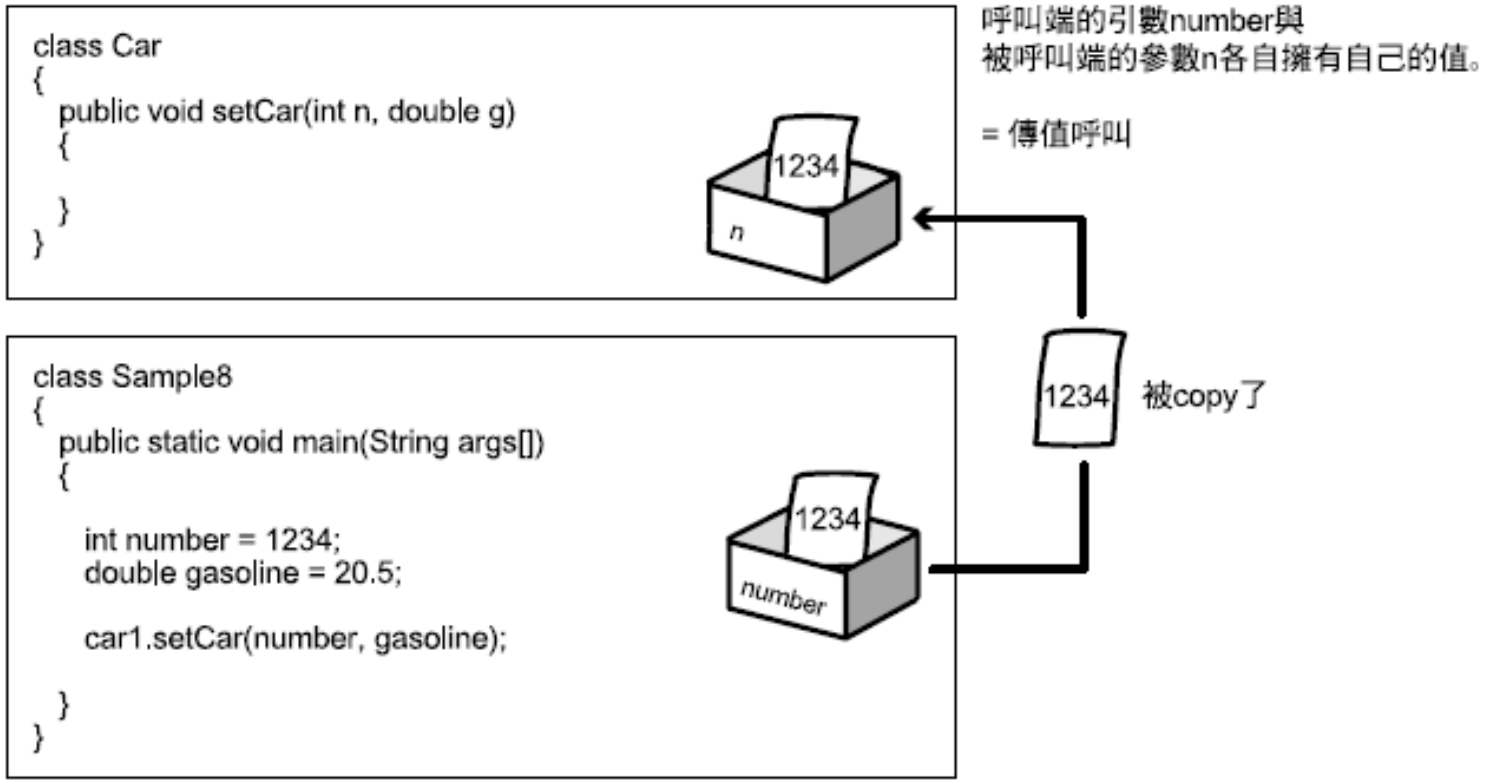
傳址呼叫(passing by address)

呼叫方法時，實際上只是把該物件的參考(記憶體位址)傳進去給方法而已，透過這個共同的參考位址，呼叫端與被呼叫端會**指向同一個物件**。通常用來傳遞物件和陣列給方法。



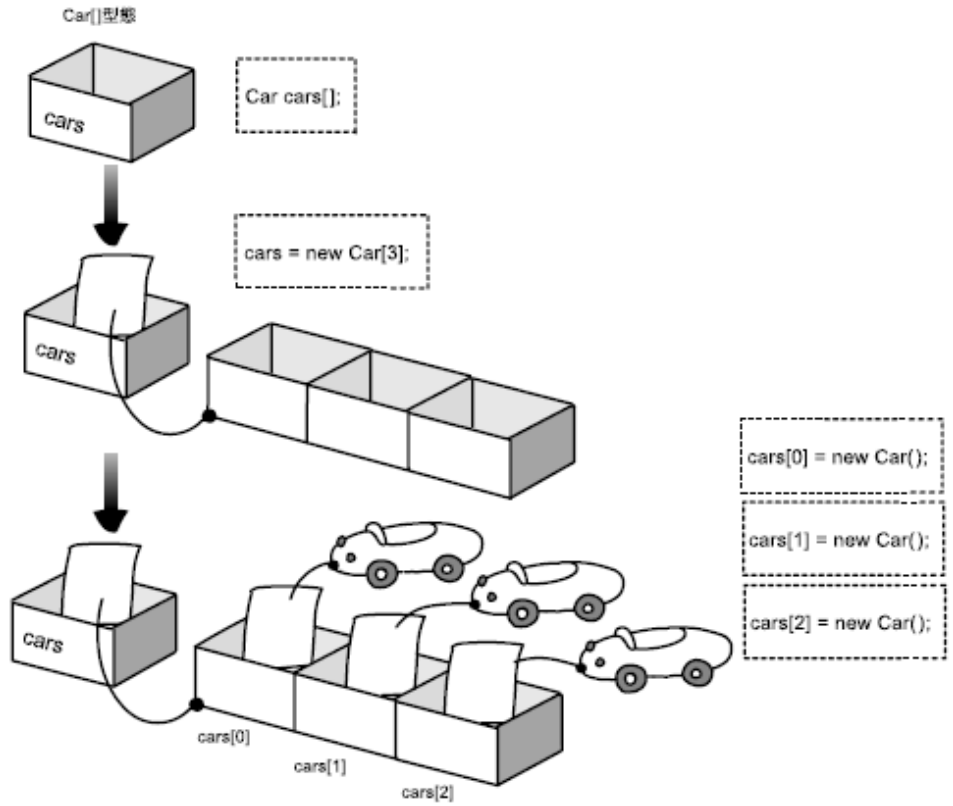
傳值呼叫(passing by value)

把一般性的資料**複製一份**並傳進去給方法，呼叫端與被呼叫端各自擁有自己的值。通常用來傳遞數字、文字等一般資料給方法。



物件的陣列(1/2)

我們可以像儲存整數一樣，利用陣列一次儲存多個物件。



物件的陣列(2/2)

準備好Car[]型態陣列變數並宣告三個Car類別型態的陣列元素

建立三個新物件，分別指定給三個陣列元素

Sample3_24.java

```
1
2  public class Sample3_24
3  {
4    public static void main(String[] args)
5    {
6      Car[] car1;
7      car1 = new Car[3];
8
9      for(int i=0; i<car1.length; i++){
10       car1[i] = new Car();
11     }
12
13     car1[0].setCar(1234, 20.5);
14     car1[1].setCar(2345, 30.5);
15     car1[2].setCar(3456, 40.5);
16
17     for(int i=0; i< car1.length; i++)
18     {
19       car1[i].show();
20     }
21   }
22 }
```

```
24 class Car
25 {
26   int num;
27   double gas;
28
29   public Car()
30   {
31     num = 0;
32     gas = 0.0;
33     System.out.println("生產了車子");
34   }
35
36   public void setCar(int n, double g)
37   {
38     num = n;
39     gas = g;
40     System.out.println("將車號設為" + num + "，汽油量設為" + gas);
41   }
42
43   public void show()
44   {
45     System.out.println("車號是" + num);
46     System.out.println("汽油量是" + gas);
47   }
48 }
```

Console | Markers | Prope

<terminated> Sample3_24 [Java Ap

```
生產了車子
生產了車子
生產了車子
將車號設為1234，汽油量設為20.5
將車號設為2345，汽油量設為30.5
將車號設為3456，汽油量設為40.5
車號是1234
汽油量是20.5
車號是2345
汽油量是30.5
車號是3456
汽油量是40.5
```

輸出結果

分別為陣列元素做setCar方法