Java 程式設計進階
方法Method

鄭安翔

ansel_cheng@hotmail.com

課程大綱

1) **方法宣告及呼叫**
2) 方法多載
3) 變動長度的方法呼叫

類別實體方法

- 將物件相關的重複操作行為定義為方法
  - 縮短主流程程式碼
  - 程式具結構化
  - 流程重複使用
  - 易於維護
- 實體方法
  - 透過特定物件實體來呼叫其操作行為

類別與方法

Shirt
+shirtID: int +colorCode: char +size: String +price: double +description : String
+Shirt (color: char, size: String, price: double, description: String)
+displayInformation ( ) +setPrice(p: double) +getPrice ( ) : double

```
01  public class Shirt {
02
03      public int shirtID = 0;
04      public char colorCode = 'R';
05      public String size = "XL" ;
06      public double price = 299.00;
07      public String description = "Polo Shirt";
08
09      public Shirt(char color, String size,
10                   double price, String desc) {
11          this.colorCode = color;
12          this.size = size;
13          this.price = price;
14          this.description = desc;
15      }
16
17      public void displayInformation() {
18          System.out.println("Shirt ID:" + shirtID);
19          System.out.println("Color:" + colorCode);
20          System.out.println("Size:" + size);
21          System.out.println("Price:" + price);
22      }
23      public void setPrice(double p) {
24          price = p;
25      }
26      public double getPrice( ) {
27          return price;
28      }
29
30  }
```

物件方法
(操作)

方法宣告 Method Declaration

- 方法
  - 定義類別物件的行為

```
[modifiers] <return_type> <name> ([arguments]) {
    return returnValue ;
}
```

Arguments list

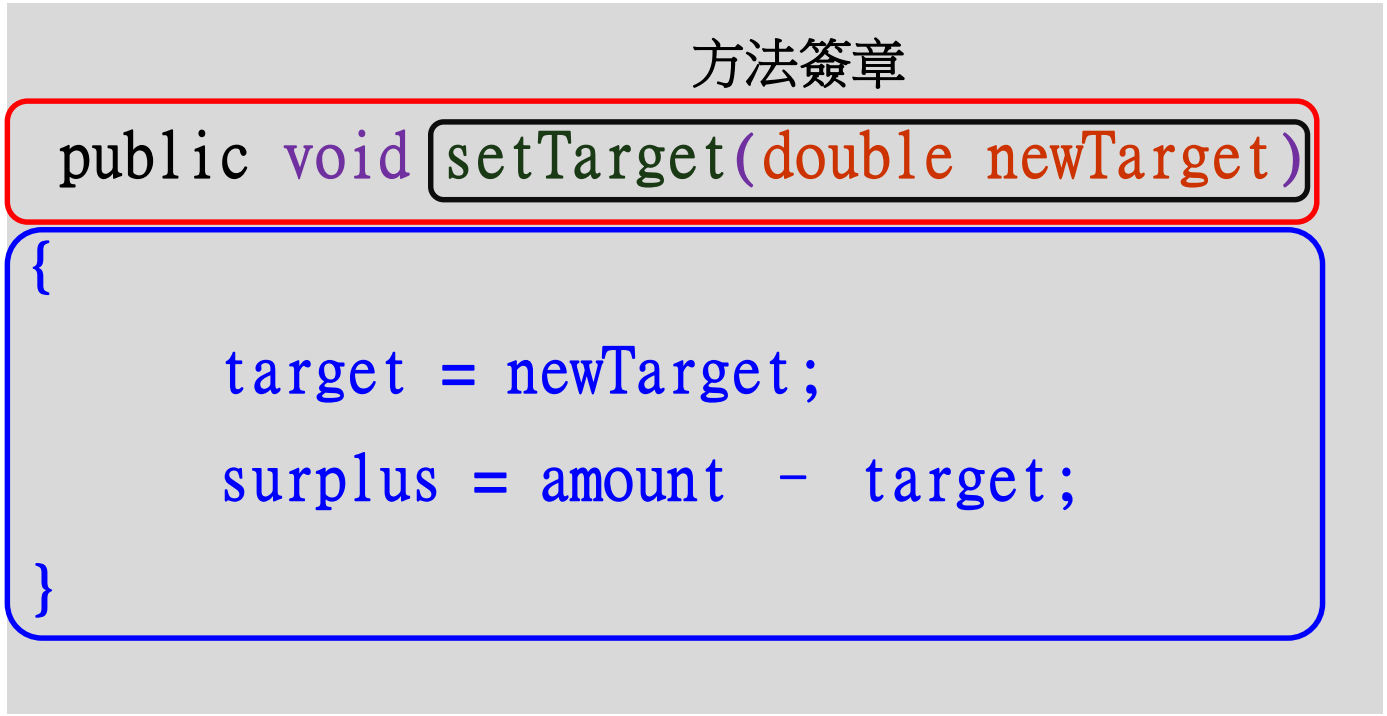
Name of the method

Return data type

Accessibility

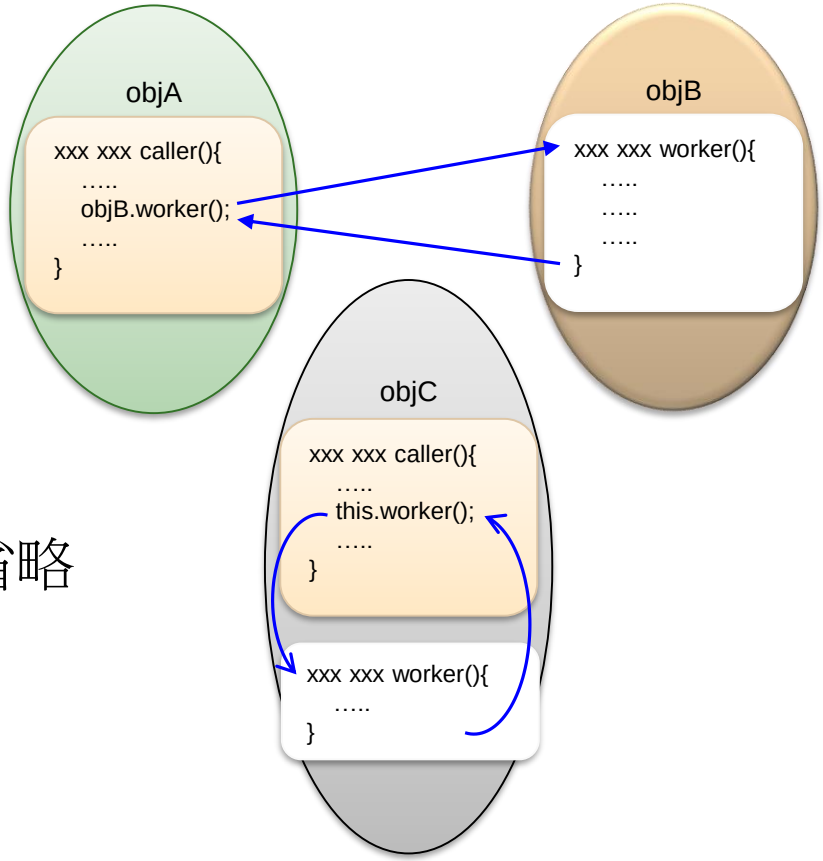
  - 傳回值
    - 有傳回值時, return 後的變數/常數型態須予宣告型態一致
    - 方法如不需傳回值,傳回值宣告為 void, return 可省略
  - Argument list:
    - 可為0~N個
    - Type Name
    - 若有一個以上,用，隔開

方法介面、簽章、本體


方法簽章
public void setTarget(double newTarget)
{
target = newTarget;
surplus = amount - target;
}
方法介面
方法本體

方法呼叫

- 方法呼叫
  - 呼叫其他物件執行方法
    - 物件參考.方法( );
  - 呼叫物件自身的其他方法
    - this.方法( );
    - this 表示物件自身參考，可省略



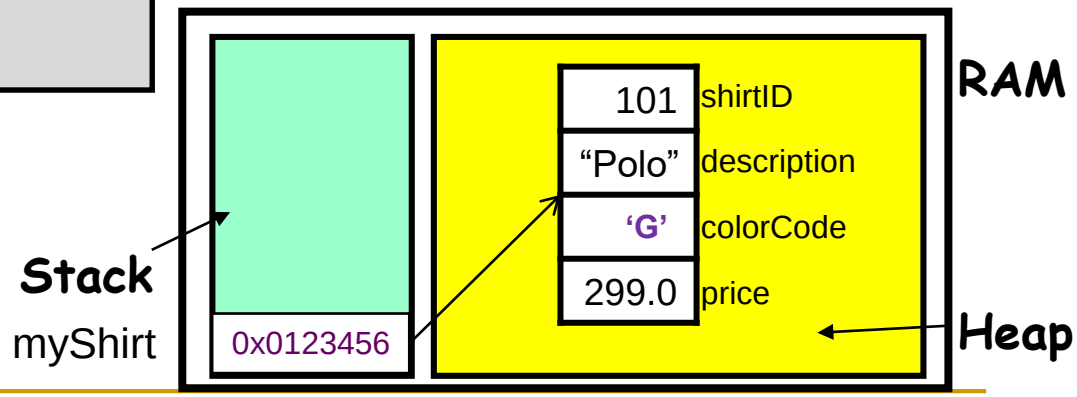
方法呼叫操作

- 物件參考.物件方法

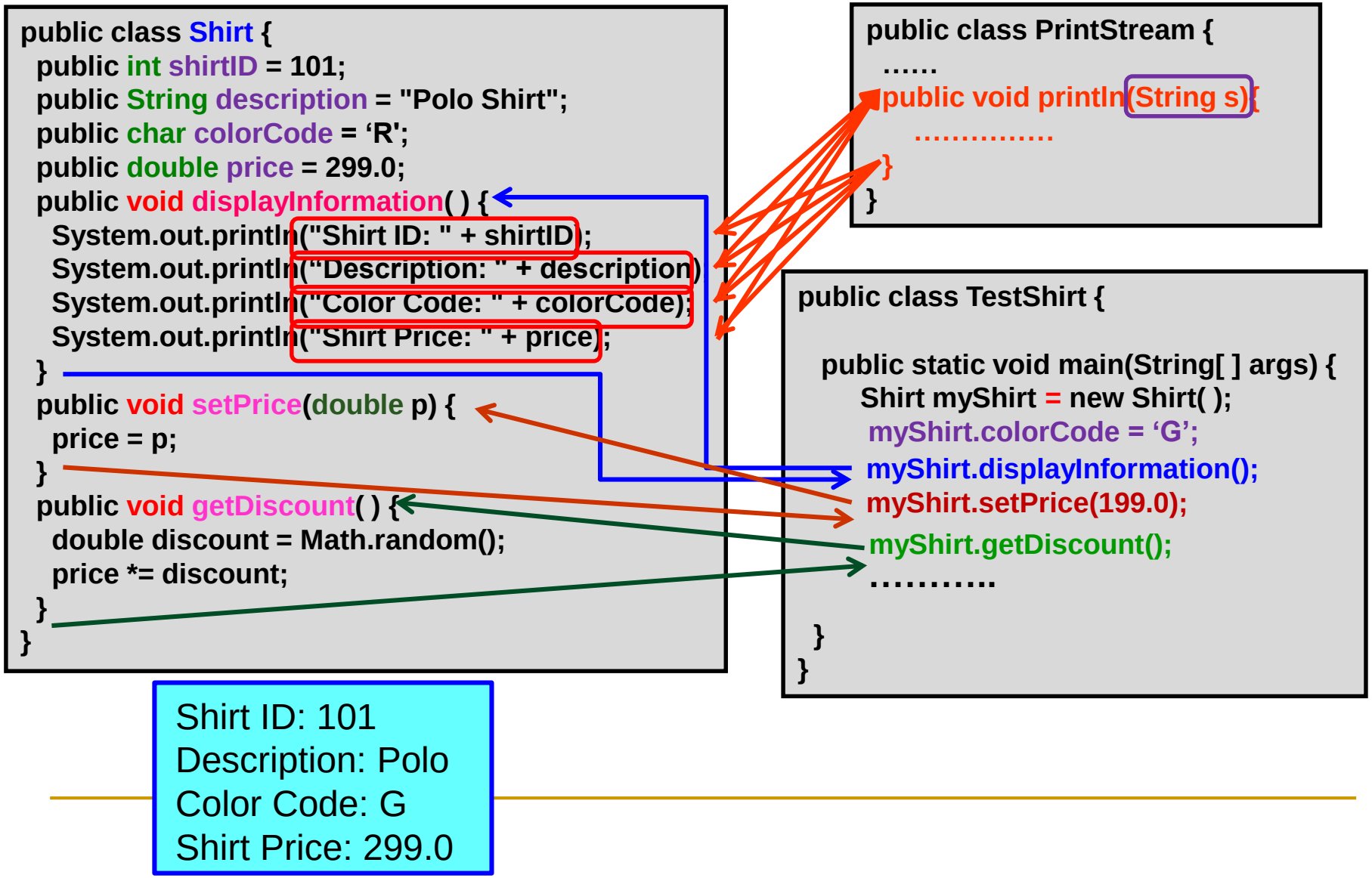
```
public class Shirt {
  public int shirtID = 101;
  public String description = "Polo Shirt";
  public char colorCode = 'R';
  public double price = 299.0;

public void displayInformation( ) {
    System.out.println("Shirt ID: " + shirtID);
    System.out.println("Description: " + description);
    System.out.println("Color Code: " + colorCode);
    System.out.println("Shirt Price: " + price);
  }
}
```

```
public class TestShirt {

  public static void main(String[ ] args) {
      Shirt myShirt = new Shirt( );
      myShirt.colorCode = 'G';
      myShirt.displayInformation();

  }
}
```

在螢幕上印
Shirt ID: 101
Description: Polo
Color Code: G
Shirt Price: 299.0



方法呼叫堆疊 Call Stack



方法呼叫參數傳遞

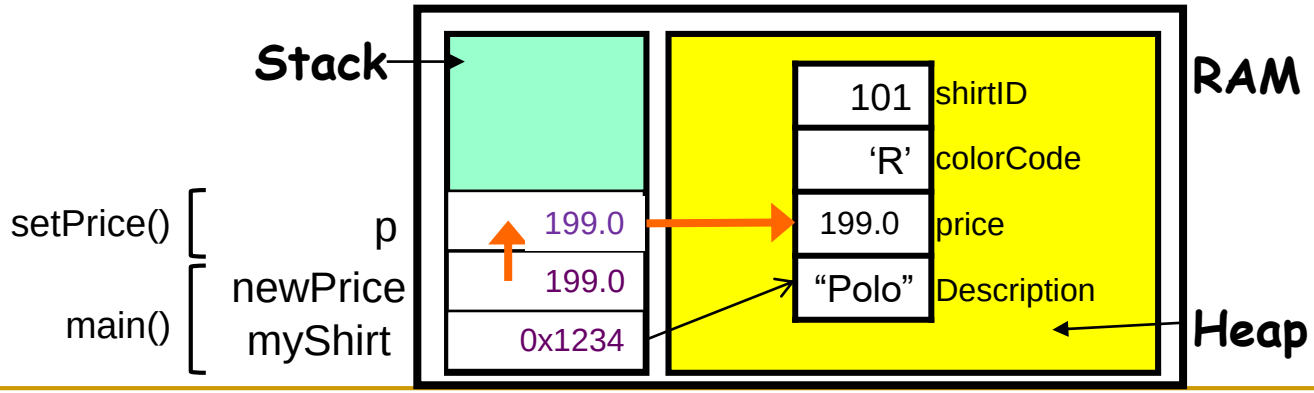
- 方法呼叫參數
  - Actual Argument (實際引數)：方法呼叫時呼叫端引數
  - Formal Parameter (形式參數 / 虛引數)：被呼叫端宣告的參數
- 方法呼叫參數傳遞方式
  - 傳值呼叫：實引數與虛引數使用不同記憶體空間，各自獨立，不互相影響。
  - 傳址呼叫：實引數與虛引數使用相同的記憶體空間，使得彼此互相影響。
- Java 的參數傳遞方式為傳值呼叫

Pass by Value (傳值)

- Java 中參數的指派是傳遞目前 stack 中的內容
  - primitive type - 內容值

```
public class Shirt {
  public int shirtID = 101;
  public char colorCode = 'R';
  public double price = 299.0;
  public String description = "Polo Shirt";
  public void setPrice(double p) {
    price = p;
  }
}
```

```
public class TestShirt {

  public static void main(String[ ] args) {
      Shirt myShirt = new Shirt( );
      double newPrice = 199.0;
      myShirt.setPrice(newPrice);
      ..........
  }
}
```

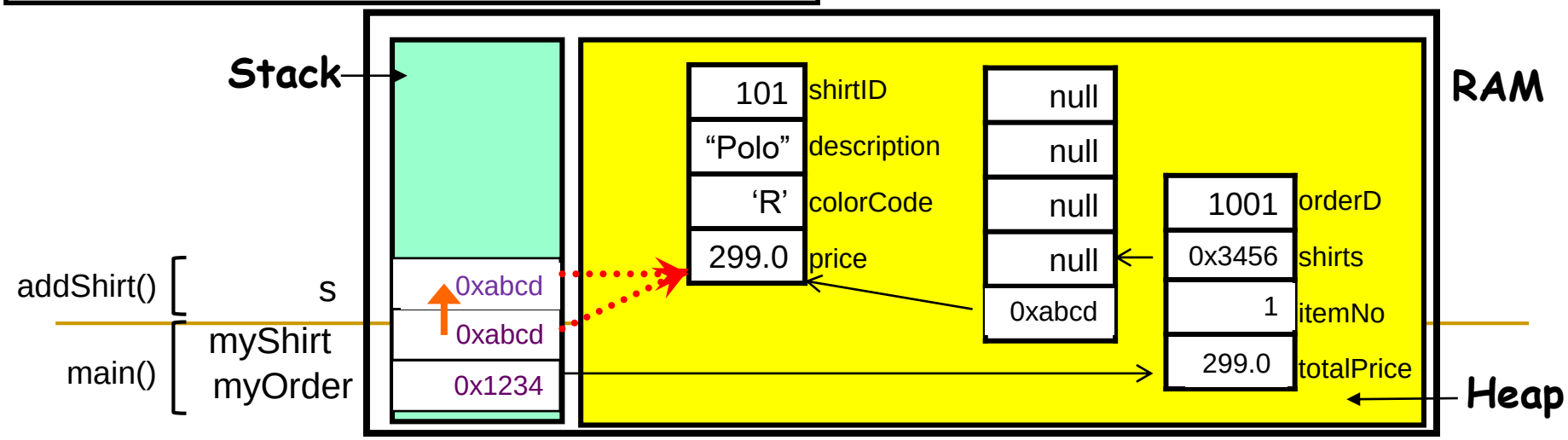


Pass by Value (傳值)

- Java 中參數的指派是傳遞目前 stack 中的內容
  - reference type - 參考值

```
public class Order {
   public int orderID = 1001;
   public Shirt[ ] shirts = new Shirt[5];
   public int itemNo = 0;
   public double totalPrice = 0.0;
   public void addShirt(Shirt s) {
      shirt[itemNo++] = s;
      totalPrice += s.price;
   }
}
```

```
public class TestOrder {

   public static void main(String[ ] args) {
      Order myOrder = new Order( );
      Shirt myShirt = new Shirt();
      myOrder.addShirt(myShirt);
      ……
   }
}
```



屬性、傳入參數、區域變數

- 屬性、傳入參數、區域變數都用來儲存資料

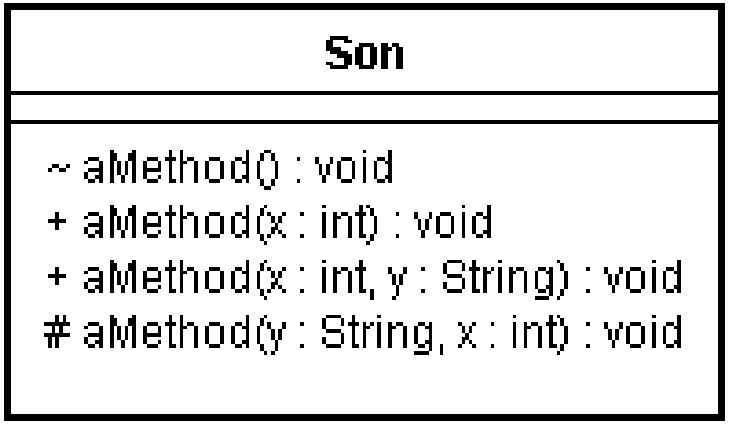
項目	屬性(Attribute)	傳入參數(Parameter)	區域變數(Local Variable)
目的	儲存物件屬性	傳遞輸入值給方法	方法內使用的暫時變數
宣告位置	類別本體之內,方法之外	方法簽章中	方法本體內
宣告方法	**modifier type name [=value] ;**	**method(type name, …)**	**type name [=value] ;**
初始值	預設值->指定值->建構子	呼叫方法	指定值
有效範圍	類別內所有方法	與方法簽章對應之方法	宣告處到程式區塊結束

課程大綱

1) 方法宣告及呼叫
2) **方法多載**
3) 變動長度的方法呼叫

方法多載 method overloading

- 同名方法名稱根據其不同的參數型別以對應執行到不同的實作。




```
01 public class Son {
02     void aMethod() { }
03     public void aMethod(int x) { }
04     public void aMethod(int x, String y) { }
05     protected void aMethod(String y, int x) { }
06 }
```

Overloaded Method

```
public class Calculator {
   public int sum(int numberOne, int numberTwo){
     System.out.println("Method One");
     return numberOne + numberTwo;
   }

  public float sum(float numberOne, float numberTwo) {
     System.out.println("Method Two");
     return numberOne + numberTwo;
  }

  public float sum(int numberOne, float numberTwo) {
    System.out.println("Method Three");
    return numberOne + numberTwo;
  }
}
```

```
public class CalculatorTest {

  public static void main(String [] args) {

    Calculator myCalculator = new Calculator();

    int totalOne = myCalculator.sum(2,3);
    System.out.println(totalOne);

    float totalTwo = myCalculator.sum(15.99F, 12.85F);
    System.out.println(totalTwo);

    float totalThree = myCalculator.sum(2, 12.85F);
    System.out.println(totalThree);
  }
}
```

常用 Java API方法多載

void	println() Terminates the current line by writing the line separator string.
void	println(boolean x) Prints a boolean and then terminate the line.
void	println(char x) Prints a character and then terminate the line.
void	println(char[] x) Prints an array of characters and then terminate the line.
void	println(double x) Prints a double and then terminate the line.
void	println(float x) Prints a float and then terminate the line.
void	println(int x) Prints an integer and then terminate the line.
void	println(long x) Prints a long and then terminate the line.
void	println(Object x) Prints an Object and then terminate the line.
void	println(String x) Prints a String and then terminate the line.

void	print(boolean b) Prints a boolean value.
void	print(char c) Prints a character.
void	print(char[] s) Prints an array of characters.
void	print(double d) Prints a double-precision floating-point number.
void	print(float f) Prints a floating-point number.
void	print(int i) Prints an integer.
void	print(long l) Prints a long integer.
void	print(Object obj) Prints an object.
void	print(String s) Prints a string.

課程大綱

1) 方法宣告及呼叫
2) 方法多載
3) **變動長度的方法呼叫**

變動長度傳入參數

- 已知個數的數字加總
  - 方法多載(overloading)的機制

```
public int sum(int x, int y) {
    return x + y;
}
public int sum(int x, int y, int z) {
    return x + y + z;
}
```

objectName.sum(1,2);

objectName.sum(1,2,3);

- 不定個數的數字加總
  - 利用 int[ ] 來當做傳入參數的資料型別，例如：

```
public int sum(int[] c) {
    int s = 0;
    for(int i=0;i<c.length;i++) {
        s += c[i];
    }
    return s;
}
```

int[] intArr = new int[ ]{1, 2, 3, 4};

objectName.sum(intArr);

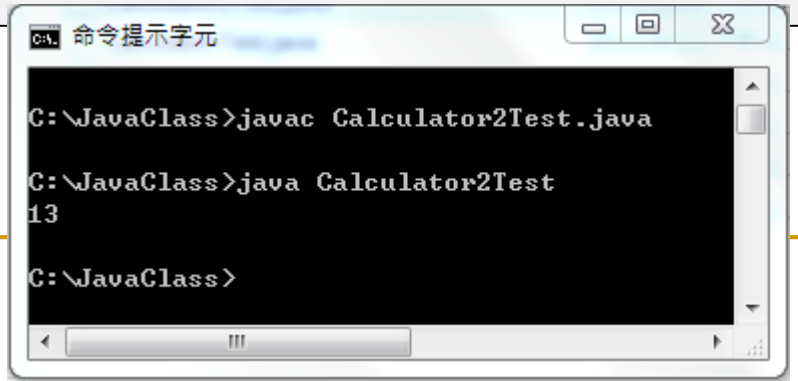
變動長度傳入參數

- 變動長度傳入參數機制
  - JavaSE 5.0 提供能隨意地增長方法中的參數
  - public int sum(**int... c**) { }

省略號(很多的意思)

```
public class Calculator2 {
    public int sum(int... c) {
        int s = 0;
        for(int i:c) {
            s += i;
        }
        return s;
    }
}
```

```
public static void main(String[] args) {
        Calculator2 calc = new Calculator2();
        int a = calc.sum(1, 2);
        int b = calc.sum(1, 2, 3, 4);
        int c = calc.sum(a, b);
        System.out.println(c);
    }
```



變動長度傳入參數使用注意事項

- 省略號的參數必須是方法參數列中最後一個參數
  - ✓ **void calc(int x, int... c) {} //** 正確
  - x **void calc(int... c, int x) {} //** 錯誤

- 當使用省略號做為方法中唯一的參數列時，呼叫端也可以不傳進參數

```
void calc(int... c) {
    …..
}
```

```
objectName.calc(1, 2, 3, 4);

objectName.calc();  //可以不傳進參數
```