




# 繼承與多型

## 學習目標

- 瞭解繼承目的
  - 瞭解繼承與多型的關係
  - 知道如何重新定義方法
- 

## 繼承 (Inheritance)

- 繼承(**inheritance**)的觀念指的是物件可自其他物件延續使用其成員變數及成員方法，物件導向程式設計的繼承觀念可方便資料及方法的重覆使用。
- 繼承是一種由已有的類別創建新類別的機制。利用繼承，我們可以先創建一個共有屬性的一般類別，根據該一般類別再創建具有特殊屬性的新類別，新類別繼承一般類的狀態和行為，並根據需要增加它自己的新的狀態和行為。由繼承而得到的類別稱為子類別，被繼承的類別稱為父類別。

# 子類別與父類別

- 父類別可以是自己編寫的類別也可以是JAVA類庫中的類別。
- 利用繼承有利於實現代碼的重複使用，子類別只需要添加新的功能代碼即可。JAVA不支援多重繼承，即子類別只能有一個父類別。

```
class 子類別名 extends 父類別名別{  
    ...  
}
```

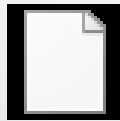
類別的宣告中沒有使用關鍵字extends時，被系統預設為是Object的子類別，Object是包java.lang中的類別。

class A{		class A extends <b>Object</b> {
... ..		... ..
}	=	}



CMath.java

```
package ch6_1;
public class CMath {
    public void getMax(int a, int b) {
        int bigNum;
        if (a > b)
            bigNum = a;
        else
            bigNum = b;
        System.out.println(a + " 與 " + b + " 的最大數為 " + bigNum);
    }
}
```



SonCMath.java

```
package ch6_1;
public class SonCMath extends CMath {
    public void getFactorial(int a) {
        int ans = 1, i;
        System.out.print(a + "! = ");
        for (i = 1; i < a; i++) {
            System.out.print(i + " * ");
            ans *= i;
        }
        ans *= a;
        System.out.println(a + " = " + ans);
    }
}
```

```
ch6_1
├── JRE 系統程式庫 [jdk-17.0.4.1]
└── src
    └── ch6_1
        ├── CMath.java
        ├── ExtendDemo.java
        └── SonCMath.java
```



ExtendDemo.java

```
package ch6_1;

public class ExtendDemo {
    public static void main(String[] args) {
        SonCMath math1 = new SonCMath();
        math1.getMax(15, 43);
        // 呼叫子類別繼承父類別的方法
        math1.getFactorial(6);
        // 呼叫子類別自己的方法
    }
}
```

結果:

15 與 43 的最大數為 43

$6! = 1 * 2 * 3 * 4 * 5 * 6 = 720$

# 子類別的繼承性

## 繼承的定義

- 子類別的成員中有一部分是子類別自己宣告的定義，另一部分是從它的父類別繼承的。
- 子類別繼承父類別的成員變數作為自己的一個成員變數，就好像它們是在子類別中直接宣告一樣，可以被子類別中自己宣告的任何實例方法操作。
- 子類別繼承父類別的方法作為子類別中的一個方法，就像它們是在子類別中直接宣告一樣，可以被子類別中自己宣告的任何實例方法呼叫。

## 存取限制相同類別庫

子類別和父類別在相同類別庫中

- 子類別繼承了其父類別`private`以外的成員變數作為自己的成員變數。
- 繼承了父類中`private`以外的方法作為自己的方法。
- 繼承的成員變數以及方法的存取權限保持不變。

## 存取限制相同類別庫

	修飾字	可否繼承	可否存取
在相同類別庫中	<b>public</b>	可	可
	<b>private</b>	否	否
	<b>protected</b>	可	可
	無修飾字	可	可



## 存取限制相同類別庫

```
package ch6_2;
public class Father {
    private int moneyDollar=300;
    int moneyNTD=200;
    int add(int x,int y){
        return x+y;
    }
}
```

```
public class Son extends Father{
    int moneyUSD=800;
    public void changMoneyNTD(int x){
        moneyNTD=x;
    }
    public void changMoneyUSD(int x){
        moneyUSD=x;
    }
    int subs(int x,int y){
        return x-y;
    }
}
```

```
package ch6_2;
public class GrandSon extends Son{
    int multi(int x,int y){
        return x*y;
    }
}
```

## 存取限制相同類別庫

```
package ch6_2;
public class ch6_2 {
    public static void main(String[] args) {
        int a=5,b=3;
        Son s=new Son();
        GrandSon sund=new GrandSon();
        s.changMoneyNTD(666);
        s.changMoneyUSD(5000);
        System.out.println("兒子的台幣是繼承的屬性, 當前的值是:"+s.moneyNTD);
        System.out.println("兒子的美金是新增的屬性, 當前的值是:"+s.moneyUSD);
        System.out.printf("減法是兒子新增的功能,%d-%d等於%d\n",a,b,s.subs(a,b));
        System.out.printf("加法是兒子繼承的功能,%d+%d等於%d\n",a,b,s.add(a,b));
        System.out.println("孫子的台幣和美金都是繼承的屬性, 當前的值是:");
        System.out.println("台幣:"+sund.moneyNTD+" 美金:"+sund.moneyUSD);
        System.out.printf("乘法是孫子新增的功能,%d*%d等於%d\n",a,b,sund.multi(a,b));
        System.out.printf("加法是孫子繼承的功能,%d+%d等於%d\n",a,b,sund.add(a,b));
        System.out.printf("減法是孫子繼承的功能,%d-%d等於%d\n",a,b,sund.subs(a,b));
    }
}
```

# 存取限制相同類別庫

## 結果:

兒子的台幣是繼承的屬性,當前的值是:666

兒子的美金是新增的屬性,當前的值是:5000

減法是兒子新增的功能, $5-3$ 等於2

加法是兒子繼承的功能, $5+3$ 等於8

孫子的台幣和美金都是繼承的屬性,當前的值是:

台幣:200 美金:800

乘法是孫子新增的功能, $5*3$ 等於15

加法是孫子繼承的功能, $5+3$ 等於8

減法是孫子繼承的功能, $5-3$ 等於2