# 多型

多型包括：

1. 方法多型
2. 物件多型

class A{

public void print(){

System.out.println("A");

}

public void ppp(){

this.print();

}

}

class B extends A{

public void print(){

System.out.println("B");

}

}

public class OverRide(){

public static void main(String args[]){

B b = new B();

b.ppp();

}

}

由那一個子類別產生的物件，就用該子類別覆寫的方法。

### 向上轉變資料型態

public class OverRide(){

public static void main(String args[]){

A a = **new B();**

a.ppp();

}

}

輸出什麼？

為什麼？

### 向下轉變資料型態

public class OverRide(){

public static void main(String args[]){

A a = new B();

B b = (B) a;

b.ppp();

}

}

輸出什麼？

為什麼？

public class OverRide(){

public static void main(String args[]){

A a = new A();

B b = (B) a;

b.ppp();

}

}

輸出什麼？

為什麼？

ClassCastException兩個沒有關係的類別互相轉變。

## 抽象類別

類型中只要包含一個以上的抽象方法，就稱為抽象類別。

class A{

public void print(){

System.out.println("Hello");

}

public abstract void yes();**//注意：沒有{}**

}

抽象方法特徵：

1. 加abstract修飾字

2. 沒有{}

## 歸納

抽象類別可以有構造子

可以沒有抽象方法，但不能實體化

抽象類別不能使用final修飾字

模板設計

# 介面

實務上，類別只繼承抽象類別(Abstract)和實作介面(Interface)；不繼承已完成的類別。

## 介面的定義

interface沒有建構方法

interface中所有資料成員都必須初始化，且為常數，宣告變數必須是final，static和public。(可省)

interface中的method必須為abstact或public

實作用關鍵字implements

interface A{

public static final String MESSAGE = "HELLO";//全域常數

public abstract void yes();//抽象方法

}

介面必須有子類別。

子類別可以實作多個介面

interface A{

public static final String MESSAGE = "HELLO";

public abstract void yes();

}

interface B{

public abstract void good();

}

}

class Y implements A,B{

public void yes(){

System.out.println("YES");

}

public void good(){

System.out.println("GOOD");

}

}

public class InterfaceDemo{

public static void main(String args[]){

A a = new Y();

a.good();

}

}

## 介面與抽象類別的異同

abstract class的特徵：

1. abstract class和abstract method都必須用abstract關鍵字來修飾。
2. abstract class不能用new關鍵字去產生物件。
3. abstract method只需要宣告，不需要實作。
4. 繼承abstract class的子類別必須實作父類別的abstract method，否則這子類別還是個abstract class。

interface的特徵:

1. interface沒有建構方法。
2. interface中所有資料成員都必須初始化，均為常數。
3. 宣告變數必須是final，static和public。
4. interface中的method必須為abstact或public。

繼承類別是(is-a)

實作介面是(like-a)

提供一種多個class一起合作工作的方式，將多個class中相同的元素pull up method到public class中，再以繼承的方式使用它，目的是為了實現"多型"精神

|  |  |  |
| --- | --- | --- |
|  | 抽象類別Abstract Class | 介面Interface |
| 父類別/父介面繼承 | 只能繼承一個類別 | 能繼承多個介面(Java實現多重繼承) |
| 子類別繼承/實作 | 只能繼承(extends)一個抽象類別 | 能實作(implements)多個介面 |
| 方法 | 可包含非抽象方法 | 只能是抽象方法(Java8以前…XD) |
| 應用 | 關係密切的類別中如定義抽象類別Car，子類別Benz及Audi繼承Car | 定義一些功能給較不相干類別使用，如定義介面飛翔Fly子類別，AirPlane及Bird實作Fly，但實務上其實不會不相干類別實作同一介面 |
|  | 必定為父類別 | 可視為抽象類別的特例 |

繼承又實作

class Y extends A implements B,C{}

接口可實作多個接口，但不能繼承抽象類別。

## 介面的功用

做為操作標準

一種操作能力

將伺服器端的方法名稱提供給用戶端