# JAVA抽象类和接口异同点

[JAVA抽象类和接口异同点](#java抽象类和接口异同点)  
 [抽象类](#抽象类)  
 [抽象类的概念](#抽象类的概念)  
 [实例](#实例-1)  
 [接口类](#接口类)  
 [接口类的概念](#接口类的概念)  
 [实例](#实例-2)  
 [抽象类和接口的异同点](#抽象类和接口的异同点)

## 抽象类

### 抽象类的概念

在继承的层次结构中，每个新的子类都使类变得更加明确和具体。如果从一个子类向父类追溯，类就会变得更通用、更加不明确。类的设计应该确保父类包含它的子类的共同特征。有时候，一个父类设计的非常抽象，以至于它都没有任何具体的实例。这样的类称为抽象类。

抽象类的设计目的，是代码复用。当不同的类具有某些相同的行为(记为行为集合A)，且其中一部分行为的实现方式一致时（A的非真子集，记为B），可以让这些类都派生于一个抽象类。在这个抽象类中实现了B，避免让所有的子类来实现B，这就达到了代码复用的目的。而A减B的部分，留给各个子类自己实现。正是因为A-B在这里没有实现，所以抽象类不允许实例化出来（否则当调用到A-B时，无法执行）。

抽象方法的定义格式：访问修饰符 abstract 返回类型 方法名（参数列表）；

### 实例

abstract class A{//定义一个抽象类  
   
 public void fun(){//普通方法  
 System.out.println("存在方法体的方法");  
 }  
   
 public abstract void print();//抽象方法，没有方法体，有abstract关键字做修饰  
   
}  
//单继承  
class B extends A{//B类是抽象类的子类，是一个普通类  
  
 @Override  
 public void print() {//强制要求重写  
 System.out.println("Hello World !");  
 }  
   
}  
public class TestDemo {  
  
 public static void main(String[] args) {  
 A a = new B();//向上转型  
   
 a.print();//被子类所重写的过的方法  
 }  
}

运行结果：

Hello World !

* 抽象类继承子类里面有明确的方法重写要求，而普通类可以有选择性的来决定是否需要重写；
* 抽象类实际上就比普通类多了一些抽象方法而已，其他组成部分和普通类完全一样；
* 普通类对象可以直接实例化，但抽象类的对象必须经过向上转型之后才可以得到。

## 接口类

### 接口类的概念

接口是一种与类相似的结构，用于为对象定义共同的操作。接口在许多方面都与抽象类很相似，但是他的目的是指明相关或者不相关类的对象的共同行为。例如，使用适当的接口，可以指明这些对象是可比较的、可食用的或者可克隆的。为了区分接口和类，Java采用下面的语法来定义接口：

modifier interface InterfaceName{  
 /\*\*Constant declarations\*/  
 /\*\*Abatract method signatures \*/  
}

### 实例

interface A{//定义一个接口A  
  
 public static final String MSG = "hello";//全局常量  
  
 public abstract void print();//抽象方法  
}  
  
interface B{//定义一个接口B  
  
 public abstract void get();  
}  
  
class X implements A,B{//X类实现了A和B两个接口  
  
 @Override  
 public void print() {  
 System.out.println("接口A的抽象方法print()");  
 }  
  
 @Override  
 public void get() {  
 System.out.println("接口B的抽象方法get()");  
 }  
  
}  
  
public class TestDemo {  
  
 public static void main(String[] args){  
  
 X x = new X();//实例化子类对象  
 A a = x;//向上转型  
 B b = x;//向上转型  
  
 a.print();  
 b.get();  
 }  
  
}

运行结果：

接口A的抽象方法print()  
接口B的抽象方法get()

对于接口，里面的组成只有抽象方法和全局常量，所以很多时候为了书写简单，可以不用写public abstract 或者public static final。并且，接口中的访问权限只有一种：public，即：定义接口方法和全局常量的时候就算没有写上public，那么最终的访问权限也是public，注意不是default。

## 抽象类和接口的异同点

| 比较点 | 抽象类 | 接口 |
| --- | --- | --- |
| 关键字 | abstract class | interface |
| 字段 | 无限制 | 必须是public、static和final的 |
| 方法 | 既可以含普通方法，也可以含抽象方法 | 只能含抽象方法，且必须是public的 |
| 继承/实现 | 只能被类或抽象类继承 | 既可以被接口继承，也能被类或抽象类实现 |
| 多重继承 | 不支持 | 可以继承多个父接口 |

1.抽象类要被子类继承，接口要被类实现

2.接口只能做方法声明，抽象类中可以作方法声明，也可以做方法实现。

3.接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。

4.接口是设计的结果，抽象类是重构的结果。

5.抽象类和接口都是用来抽象具体对象的，但是接口的抽象级别最高。

6.抽象类可以有具体的方法和属性，接口只能有抽象方法和不可变常量。

7.一个抽象类只能继承一个抽象父类，而接口可以继承多个接口；一个子类只能继承一个抽象类，却可以实现多个接口（在Java中，接口的主要功能是解决单继承局限问题）