Extend and implements区别

在类的声明中，通过关键字extends来创建一个类的子类。

一个类通过关键字implements声明自己使用一个或者多个接口。

extends 是继承某个类, 继承之后可以使用父类的方法, 也可以重写父类的方法; implements 是实现多个接口, 接口的方法一般为空的, 必须重写才能使用。

接口实现的注意点：

a.实现一个接口就是要实现该接口的所有的方法(抽象类除外)。

b.接口中的方法都是抽象的。

c.多个无关的类可以实现同一个接口，一个类可以实现多个无关的接口。

Java类不支持多继承，但可以通过接口间接实现。

Class A extends B implements C,D,E

Abstract class

抽象类。一个类用abstract修饰，这就是一个抽象类。

抽象方法，一个方法用abstract修饰，这就是一个抽象方法。

抽象类里面有抽象的方法（当然也可以有具体的方法也可以没有抽象方法），但是不具体实现，留给子类去实现。

抽象类最大的意义是作为父类存在，很好的体现面向对象的继承和多态，抽象类自己是不能进行实例化的。

Abstract 和interface的联系和区别

1、abstract修饰的类是抽象类，里面的抽象方法由abstract修饰，没有方法体，但是可以有其他成员变量，构造函数，非抽象方法

2、抽象类不能产生实例，必须要由子类去实现具体的方法，并且如果子类不是抽象类，要实现父类全部的抽象方法

3、一个类中有抽象方法，这个类就得是抽象类

4、一个类可以实现多个接口，用implements关键字，用逗号隔开多个接口

5、实现接口的类不是抽象类的话也要实现接口中所有的方法

6、接口中成员只能是常量，方法都是public的抽象函数

7、abstract类和interface的区别：一个类只能继承一个抽象类但可以实现多个接口;接口中的所有方法都是抽象的，所有的变量都是静态不可修改的

Java void关键字

Java语言是强类型（Strong typed）的，从方法申明的语法可以看出方法的定义必须有返回值，需要确定方法返回值的类型；当方法定义时用void修饰时，表示没有返回值，那么由此可见void的应该算是一种数据类型。

public final class Void extends Object {

/\*

\* The Void class cannot be instantiated.

\*/

private Void() {}

}

Void类是一个不可实例化的占位符类，用来保存一个引用代表了Java关键字void的Class对象。

java里接口和实现类什么关系

你可以把接口看成是一种职业，比如说教师。

你把实现类看成是小学教师，或是中学教师。

无论是小学教师还是中学教师，他们都是教师，都‘继承’了教师。

你对外封装时用接口，就好比是让你选择行业，你只能选择教师，而一般不会在诸多行业中还列举出小学，中学等等。

而实现类也表现出了多态性，因为原本的只有教师这一个职业，然后因为教授学生的不同，就出现了小学教师，中学教师等等。

如果使用接口类型作为方法的返回类型，返回类型是实现了接口的类，则返回包含接口所定义的内容。实现接口的类如果还实现了其他接口，要调用则需对返回类型做类型强制转换。

前置：

public interface C{

void c();

}

public interface D{

void d();

}

Class A implements C,D

假设以下方法都返回了A类的对象：

public C getClass(){

Return new A}

只能用C的c方法

public D getClass(){

Return new A }

只能用D的d方法

构造函数和普通函数区别

（1）. 一般函数是用于定义对象应该具备的功能。而构造函数定义的是，对象在调用功能之前，在建立时，应该具备的一些内容。也就是对象的初始化内容。

（2）. 构造函数是在对象建立时由jvm调用, 给对象初始化。一般函数是对象建立后，当对象调用该功能时才会执行。

（3）. 普通函数可以使用对象多次调用，构造函数就在创建对象时调用。

（4）. 构造函数的函数名要与类名一样，而普通的函数只要符合标识符的命名规则即可。

（5）. 构造函数没有返回值类型。

构造函数要注意的细节

（1）. 当类中没有定义构造函数时，系统会指定给该类加上一个空参数的构造函数。这个是类中默认的构造函数。当类中如果自定义了构造函数，这时默认的构造函数就没有了。

（2）.在一个类中可以定义多个构造函数，以进行不同的初始化。多个构造函数存在于类中，是以重载的形式体现的。因为构造函数的名称都相同。