# 尚马教育 JAVA 基础课程

# Java面向对象--接口

文档编号：A09

创建日期： 2017-04-12

最后修改日期：2019-09-23

版 本 号：V3.0

电子版文件名：尚马教育-第一阶段-9.接口专题课程.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-01 | 王绍成 | Java基础版本更新 | V2.0 |
| 2019-08-09 | 徐丽莎 | Java基础版本更新 | V3.0 |

**主讲人：**

**徐丽莎**

目录

[尚马教育 JAVA 基础课程 1](#_Toc31580)

[Java面向对象--接口 1](#_Toc27062)

[1. 接口 2](#_Toc12856)

[1.1. 生活中的接口 2](#_Toc14706)

[1.2. 面向接口编程 2](#_Toc28469)

[1.3. 接口的定义 3](#_Toc25224)

[1.4. 接口与抽象类的区别 4](#_Toc9705)

[1.5. 类实现接口(implements) 4](#_Toc12842)

[1.5.1. 语法 4](#_Toc16104)

[1.5.2. 案例 5](#_Toc9758)

[1.6. 接口继承接口 6](#_Toc8793)

[1.6.1. 语法: 6](#_Toc18816)

[1.6.2. 案例 6](#_Toc11318)

[2. 多态(在实现的关系下) 6](#_Toc14468)

[3. 抽象类与接口的使用场景 7](#_Toc32634)

[4. Final 7](#_Toc2924)

[5. 作业 7](#_Toc6088)

## 接口

### 生活中的接口

|  |
| --- |
|  |

### 面向接口编程

* 接口是设计层面的概念，往往由设计师设计，将定义与实现分离
* 程序员实现接口，实现具体方法
* 面向接口编程和面向对象编程并不是平级的，它并不是比面向对象编程更先进的一种独立的编程思想，而是附属于面向对象思想体系，属于其一部分。或者说，它是面向对象编程体系中的思想精髓之一
* 面向接口编程的意思是指在面向对象的系统中所有的类或者模块之间的交互是由接口完成的

### 接口的定义

* 语法:

|  |
| --- |
| [访问权限] **interface** 接口名 {  公开静态常量列表；  公开抽象方法列表；  } |

* 案例:

|  |
| --- |
| **public** **interface** MyInterface{  // 常量 public static final 常量名 都是大写  **int** ***NUM*** = 100;  String ***USER\_NAME*** = "admin";  //jdk1.8之前 只有抽象方法 public abstract  String a(**final** **int** num);  //没有构造 public MyInterface() {}  //jdk1.8+ 可以有普通的功能方法  **public** **default** **void** b() {  System.***out***.println("b.............");  }  //也可以有静态方法  **public** **static** **int** c() {  System.***out***.println("c............");  **return** ***NUM***;  }  } |

### 接口与抽象类的区别

* 在Java中接口是一个比抽象类更加抽象的概念，由于只声明行为，因此在接口中的方法均是抽象的,下表中罗列了接口和抽象类的差异：

|  |
| --- |
|  |

* 成员变量方面，在接口中只存在公开静态常量（即便没有使用static final修饰，Java也会默认确定其性质）
* 成员方法方面，在接口中只存在公开抽象方法（即便没有abstract修饰，Java也会默认其抽象性质）

### 类实现接口(implements)

* 接口是比抽象类抽象层次更高的一个概念，因此和抽象类一样，它不能用于实例化对象，只能作为继承树的高层结构被子类实现（子类实现接口被称为implements（实现），其体现形式和继承类似）
* 类实现接口，本质上与类继承类相似，区别在于“类最多只能继承一个类，即单继承，而一个类却可以同时实现多个接口”，多个接口用逗号隔开即可。实现类需要覆盖所有接口中的所有抽象方法，否则该类也必须声明为抽象类
* 接口定义的是多个类都要实现的操作，即“what to do”。类可以实现接口，从而覆盖接口中的方法，实现“how to do”

#### 语法

* 类实现接口的语法:

|  |
| --- |
| [修饰符] **class** <类名> [**extends** 父类名] [**implements** 接口列表]{  } |

* 一个类可以同时继承自一个父类并实现若干接口

#### 案例

|  |
| --- |
| **public** **class** MyInterfaceImpl **extends** Object **implements** MyInterface{  **public** **int** a;  @Override  **public** String a(**final** **int** num) {  System.***out***.println("num："+num);  System.***out***.println("实现类 实现接口里面的a方法");  b();  **return** "hello";  }  //重写父接口的功能方法  @Override  **public** **void** b() {  //在实现类里面 调用父接口里面的b方法的逻辑  // MyInterface.super.b();  System.***out***.println("MyInterfaceImpl b..........");  //调用父接口的静态方法  MyInterface.*c*();  System.***out***.println(***NUM***);  System.***out***.println(MyInterface.***USER\_NAME***);  }  } |

### 接口继承接口

* 新的接口可以继承自原有的接口，新接口将拥有原有接口的所有抽象方法
* Java接口继承接口的原则：
  + Java接口可以继承多个接口
  + 接口继承接口依然使用关键字extends，不要错用成implements
  + 因此，在声明接口之间的继承关系时，extends关键字后面可以是一个列表

#### 语法

|  |
| --- |
| [修饰符] **interface** <接口名> [**extends** 接口列表]{} |

#### 案例

|  |
| --- |
| **public** **interface** A {  **void** aa();  } |
| **public** **interface** B {  **void** bb();  } |
| **public** **interface** MyInterface **extends** A,B{  } |

## 多态(在实现的关系下)

* 通过下面案例，我们来掌握在接口实现的关系下，是如何实现多态的:
  + 为各学校开发这样一个小系统，包含类型：教员、学校、打印机，具体要求如下教员、以及学校都具有方法：输出详细信息
  + 学校具有属性：打印机，能够通过学校的打印机打印教员或学校的详细信息
  + 系统要具备良好的可扩展性与可维护性

|  |
| --- |
|  |

* 升级上述的系统，要求：
  + 打印机有多种类型，比如：黑白打印机、彩色打印机等
  + 学校可能配备其中任意一款打印机，负责打印教员、或者学校的详细信息
  + 系统要具备良好的可扩展性与可维护性

|  |
| --- |
| **public** **class** Teacher  InfoInterface {  **private** **int** tid;  **private** String name;  @Override  **public** **void** showInfo() {  System.***out***.println("老师的id:"+tid+",老师的姓名:"+name);  }  } |
| **public** **class** School **implements** InfoInterface{  **private** String sname;  **private** String address;  @Override  **public** **void** showInfo() {  System.***out***.println("学校的名称："+sname+",学校地址："+address);  }  } |
| **public** **abstract** **class** Printer {  //提高print()的可扩展性  **public** **abstract** **void** print(InfoInterface infoInterface);  } |
| **public** **class** ColorPrinter **extends** Printer{  @Override  **public** **void** print(InfoInterface infoInterface) {  System.***out***.println("彩色打印开始。。。。。");  infoInterface.showInfo();  System.***out***.println("彩色打印结束。。。。。");  }  } |
| **public** **class** BlackWhitePrinter **extends** Printer {  @Override  **public** **void** print(InfoInterface infoInterface) {  System.***out***.println("黑白打印开始。。。。。");  infoInterface.showInfo();  System.***out***.println("黑白打印结束。。。。。");  }  } |

## 抽象类与接口的使用场景

* 门-->有最基本的开和关功能
* 门-->宿舍门--->在基本功能之上，有门铃的功能
* 门--->教室门--->在基本的功能之上，有猫眼的功能
* 门--->公司门--->在基本的功能之上，有猫眼和门铃的功能

## Final

* 修饰符:
  + 修饰类: 类不能被继承。
  + 修饰方法:在继承的关系下，方法不能被重写
  + 修饰变量:为常量(规范: 在成员里面变量名全部大写，在局部里面还是驼峰方式)

常量值不可更改，

## 作业

* 需求：模拟腾迅QQ使用接口搭建系统
  + 为了使整套系统具备良好的维护性和扩展性，整套系统主要由一个主程序为主要框架，所有的功能，都以插件的形式进行加载到主程序中，主程序在运行时首先会加载插件，并会自动显示加载插件过程中的运行信息，并且主程序也可以卸载插件的使用。
  + 模拟分别添加QQ空间、QQ音乐、QQ游戏三个功能，并编写测试类测试加载功能插件的过程。
* 描述 图形 、 矩形、 圆形 三个类，每一个图形都是有计算面积与周长的方法。
  + 计算类中定义一个方法可以接收所有的图像对象,并打印图形对象的周长与面积.
  + 计算类中定义一个方法可以返回任意的图形对象
* 描述一个学生类，具备id 、 name 两个属性 、 学生都具备学习的行为功能，
  + 但是有部分学生在学习的过程中也会赚钱。
  + 普通学生： 学习 不会赚钱
  + 会赚钱的学生： 学习、 还会赚钱。