|  |
| --- |
| 1、面向对象：将实现的事物抽取出来，以及事物之间的关系，抽象成类，通过继承，实现来组合。  抽象出来的有属性、功能声明及方法实现。典型的体现就是抽象类。  而，通常会想到的另一个概念接口，是对功能的抽象，只是一种声明，没有实现。且，接口中定义的变量是公开的常量。  所以，可以说接口对比与抽象类是更加的抽象化、更加解耦。  2、面向对象的好处：提高了可维护性，降低了耦合、易复用、易扩展。  3、面向对象有三大特性，封装、继承和多态。  封装：将一类事物的属性和行为抽象成一个类，使其属性私有化，行为公开化，提高了数据的隐秘性的同时，使代码模块化，也提高了安全性。  继承： 提高代码复用性，继承是多态的前提。  多态： 父类或接口定义的引用变量可以指向子类或具体实现类的实例对象，提高了程序的拓展性。  4、方法重载：在同一个类中处理不同数据的多个相同方法名的多态手段。  方法重写：相对继承而言，子类中对父类已经存在的方法进行区别化的修改。  5、多态发生的条件：  1）要有继承  2）要有重写  3）父类引用指向子类对象  6、抽象类和接口的区别  （1）语法层面上的区别  　1.一个类只能继承一个抽象类，而一个类却可以实现多个接口。  　2.抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是public static final类型的；且必须给其初值，所以实现类中不能重新定义，也不能改变其值；抽象类中的变量默认是 friendly 型，其值可以在子类中重新定义，也可以重新赋值。  　3.抽象类中可以有非抽象方法，接口中则不能有非抽象方法。（现在接口还有默认方法实现了）  　4.接口可以省略abstract 关键字，抽象类不能。  　5.接口中不能含有静态代码块以及静态方法，而抽象类可以有静态代码块和静态方法；  （2）设计层面上的区别  　1）抽象类是对一种事物的抽象，即对类抽象，而接口是对行为的抽象。  　2）设计层面不同，抽象类作为很多子类的父类，它是一种模板式设计。 |

# 从面向过程到面向对象

很早很早以前的编程是面向过程的，比如实现一个算术运算1+1 = 2，通过这个简单的算法就可以解决问题。但是随着时代的进步，人们不满足现有的算法了，因为问题越来越复杂，不是1+1那么单纯了，比如一个班级的学生的数据分析，这样就有了对象这个概念，一切事物皆对象。将现实的事物抽象出来，注意抽象这个词是重点啊，把现实生活的事物以及关系，抽象成类，通过继承，实现，组合的方式把万事万物都给容纳了。实现了对现实世界的抽象和数学建模。这是一次飞跃性的进步。

面向对象的好处：提高了可维护性，降低了耦合。

举个最简单点的例子来区分 面向过程和面向对象

有一天你想吃鱼香肉丝了，怎么办呢？你有两个选择

1、自己买材料，肉，鱼香肉丝调料，蒜苔，胡萝卜等等然后切菜切肉，开炒，盛到盘子里。

2、去饭店，张开嘴：老板！来一份鱼香肉丝！

看出来区别了吗？这就是1是面向过程，2是面向对象。

面向对象有什么优势呢？首先你不需要知道鱼香肉丝是怎么做的，降低了耦合性。如果你突然不想吃鱼香肉丝了，想吃洛阳白菜，对于1你可能不太容易了，还需要重新买菜，买调料什么的。对于2，太容易了，大喊：老板！那个鱼香肉丝换成洛阳白菜吧，提高了可维护性。总的来说就是降低耦合，提高维护性！

面向过程是具体化的，流程化的，解决一个问题，你需要一步一步的分析，一步一步的实现。

面向对象是模型化的，你只需抽象出一个类，这是一个封闭的盒子，在这里你拥有数据也拥有解决问题的方法。需要什么功能直接使用就可以了，不必去一步一步的实现，至于这个功能是如何实现的，管我们什么事？我们会用就可以了。

面向对象的底层其实还是面向过程，把面向过程抽象成类，然后封装，方便我们我们使用的就是面向对象了。

面向过程：

优点：性能比面向对象好，因为类调用时需要实例化，开销比较大，比较消耗资源。

缺点：不易维护、不易复用、不易扩展.

面向对象：

优点：易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统 更加灵活、更加易于维护 .

缺点：性能比面向过程差

# 面向对象的三大特性：

在我理解,面向对象是向现实世界模型的自然延伸，这是一种“万物皆对象”的编程思想。在现实生活中的任何物体都可以归为一类事物，而每一个个体都是一类事物的实例。面向对象的编程是以对象为中心，以消息为驱动，所以程序=对象+消息。

面向对象有三大特性，封装、继承和多态。

（1）封装

封装就是将一类事物的属性和行为抽象成一个类，使其属性私有化，行为公开化，提高了数据的隐秘性的同时，使代码模块化，也提高了安全性。这样做使得代码的复用性更高。

需要隐藏的就隐藏隔离，提供适合的方法即可。

其实封装是面向对象思想的基础，不单单是提高安全性，更多是编程思想的转变。

（2）继承

继承则是进一步将一类事物共有的属性和行为抽象成一个父类，而每一个子类是一个特殊的父类--有父类的行为和属性，也有自己特有的行为和属性。这样做扩展了已存在的代码块，进一步提高了代码的复用性。

提高代码复用性；继承是多态的前提。

A、方法的重写

子类如果对继承的父类的方法不满意（不适合），可以自己编写继承的方法，这种方式就称为方法的重写。当调用方法时会优先调用子类的方法。

重写要注意：

a、返回值类型

b、方法名

c、参数类型及个数

都要与父类继承的方法相同，才叫方法的重写。

重载和重写的区别：

方法重载：在同一个类中处理不同数据的多个相同方法名的多态手段。

方法重写：相对继承而言，子类中对父类已经存在的方法进行区别化的修改。

B、继承的初始化顺序

1、初始化父类再初始化子类

2、先执行初始化对象中属性，再执行构造方法中的初始化。

基于上面两点，我们就知道实例化一个子类，java程序的执行顺序是：

父类静态属性初始化-->父类静态代码块-->子类静态属初始化性-->子类静态代码块

--> 父类对象属性初始化--父类非静态代码块-->父类构造方法---->子类对象属性初始化--->子类非静态代码块-->子类构造方法。

注意：静态代码块只执行一次。

（3）多态

父类或接口定义的引用变量可以指向子类或具体实现类的实例对象，提高了程序的拓展性。

如果说封装和继承是为了使代码重用，那么多态则是为了实现接口重用。多态的一大作用就是为了解耦--为了解除父子类继承的耦合度。如果说继承中父子类的关系式IS-A的关系，那么接口和实现类之之间的关系式HAS-A。简单来说，多态就是允许父类引用(或接口)指向子类(或实现类)对象。很多的设计模式都是基于面向对象的多态性设计的。

总结一下，如果说封装和继承是面向对象的基础，那么多态则是面向对象最精髓的理论。掌握多态必先了解接口，只有充分理解接口才能更好的应用多态。

（4）总结

1）抽象会使复杂的问题更加简单化。

2）从以前面向过程的执行者，变成了张张嘴的指挥者。

3）面向对象更符合人类的思维，面向过程则是机器的思想

封装——模块化，降低耦合；

继承——复用性；

多态——扩展性。

# 3、抽象类

定义：抽象类前使用abstract关键字修饰，则该类为抽象类。

使用抽象类要注意以下几点：

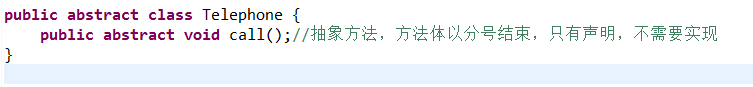
　　1. 抽象类是约束子类必须有什么方法，而并不关注子类如何实现这些方法。就是要有一类相同的接口。

　　2. 抽象类应用场景：

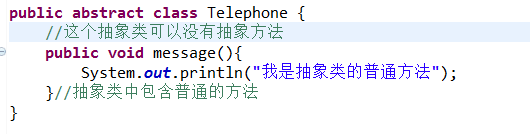
　　　a. 在某些情况下，某个父类只是知道其子类应该包含怎样的方法，但无法准确知道这些子类如何实现这些方法(可实现动态多态)。

　　　b. 从多个具有相同特征的类中抽象出一个抽象类，以这个抽象类作为子类的模板，从而避免子类设计的随意性。

　　3. 抽象类定义抽象方法，只有声明，不需要实现。抽象方法没有方法体以分号结束，抽象方法必须用abstract关键字来修饰。如:



4、包含抽象方法的类是抽象类。抽象类中可以包含普通的方法，也可以没有抽象方法。



　　5、抽象类不能直接创建，可以定义引用变量来指向子类对象，来实现抽象方法。

# ****4、接口****

　1、概念

　　接口可以理解为一种特殊的类，由**全局常量**和公共的**抽象方法**所组成。也可理解为一个特殊的抽象类，因为它含有抽象方法。

　　 如果说类是一种具体实现体，而接口定义了某一批类所需要遵守的规范，接口不关心这些类的内部数据，也不关心这些类里方法的实现细节，它只规定这些类里必须提供的某些方法。（这里与抽象类相似）

　2.接口定义的基本语法

　　　　　　[修饰符] [abstract] interface 接口名 [extends父接口1,2....]（**多继承**）{

　　　　　　　　0…n常量 (public static final)

　　　　　　　　0…n 抽象方法(public abstract)

　　　　　　}

　　其中[ ]里的内容表示可选项，可以写也可以不写;接口中的属性都是常量，即使定义时不添加public static final 修饰符，系统也会自动加上；接口中的方法都是抽象方法，即使定义时不添加public abstract修饰符，系统也会自动加上。

　3.使用接口

　　一个类可以实现一个或多个接口，实现接口使用implements关键字。java中一个类只能继承一个父类，是不够灵活的，通过实现多个接口可以补充。

**继承父类实现接口的语法为：**

[修饰符] class 类名 extends 父类 implements 接口1，接口2...{

　　　　　　　类体部分//如果继承了抽象类，需要实现继承的抽象方法；要实现接口中的抽象方法

　　　　　}

**注意：如果要继承父类，继承父类必须在实现接口之前,即extends关键字必须在implements关键字前**

补充说明：通常我们在命名一个接口时，经常以I开头，用来区分普通的类。如：IPlayGame

　　以下我们来补充在上述抽象类中的例子，我们之前已经定义了一个抽象类Telephone和子类Phone，这里我们再创建一个IPlayGame的接口，然后在原来定义的两个类稍作修改，代码如下：

1 public interface IPlayGame {

2 public void paly();//abstract 关键字可以省略，系统会自动加上

3 public String name="游戏名字";//static final关键字可以省略，系统会自动加上

4 }

[IMG_262](https://www.cnblogs.com/hysum/p/javascript:void(0);)

1 public class Phone extends Telephone implements IPlayGame{

2

3 public void call() {//继承抽象类的子类必须重写抽象方法

4 // TODO Auto-generated method stub

5 System.out.println("我重写了抽象类的方法");

6 }

7

8 @Override

9 public void paly() {

10 // TODO Auto-generated method stub

11 System.out.println("我重写了接口的方法");

12 }

13

14 }

# 5、抽象类和接口的区别

我们在多态的学习过程中认识到抽象类和接口都是实现java多态特性的关键部分，两者都包含抽象方法，只关注方法的声明而不关注方法的具体实现，那么这两者又有什么区别呢？？我们在编写java程序的时候又该如何抉择呢？

（1）语法层面上的区别

　1.一个类只能继承一个抽象类，而一个类却可以实现多个接口。  
　2.抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是public static final类型的；**且必须给其初值**，所以实现类中不能重新定义，也不能改变其值；抽象类中的变量默认是 friendly 型，其值可以在子类中重新定义，也可以重新赋值。   
　3.抽象类中可以有非抽象方法，接口中则不能有非抽象方法。（现在好像变了）  
　4.接口可以省略abstract 关键字，抽象类不能。  
　5.接口中不能含有静态代码块以及静态方法，而抽象类可以有静态代码块和静态方法；

（2）设计层面上的区别

　　1）抽象类是对一种事物的抽象，即对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。举个简单的例子，飞机和鸟是不同类的事物，但是它们都有一个共性，就是都会飞。那么在设计的时候，可以将飞机设计为一个类Airplane，将鸟设计为一个类Bird，但是不能将 飞行 这个特性也设计为类，因此它只是一个行为特性，并不是对一类事物的抽象描述。此时可以将 飞行 设计为一个接口Fly，包含方法fly( )，然后Airplane和Bird分别根据自己的需要实现Fly这个接口。然后至于有不同种类的飞机，比如战斗机、民用飞机等直接继承Airplane即可，对于鸟也是类似的，不同种类的鸟直接继承Bird类即可。从这里可以看出，继承是一个 "是不是"的关系，而 接口 实现则是 "有没有"的关系。如果一个类继承了某个抽象类，则子类必定是抽象类的种类，而接口实现则是有没有、具备不具备的关系，比如鸟是否能飞（或者是否具备飞行这个特点），能飞行则可以实现这个接口，不能飞行就不实现这个接口。

　　2）设计层面不同，抽象类作为很多子类的父类，它是一种模板式设计。而接口是一种行为规范，它是一种辐射式设计。什么是模板式设计？最简单例子，大家都用过ppt里面的模板，如果用模板A设计了ppt B和ppt C，ppt B和ppt C公共的部分就是模板A了，如果它们的公共部分需要改动，则只需要改动模板A就可以了，不需要重新对ppt B和ppt C进行改动。而辐射式设计，比如某个电梯都装了某种报警器，一旦要更新报警器，就必须全部更新。也就是说对于抽象类，如果需要添加新的方法，可以直接在抽象类中添加具体的实现，子类可以不进行变更；而对于接口则不行，如果接口进行了变更，则所有实现这个接口的类都必须进行相应的改动。