多态

参考：https://www.cnblogs.com/chenssy/p/3372798.html

一、多态

多态分为编译时多态：重载

和运行时多态：重写

1、定义

多态就是指程序中定义的**引用变量**所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序**运行期间**才确定

（一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序**运行期间**才能决定）

因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。

2、置换法则

假设Manage extends Employee

可以:
Employee boss=new Manage();

不可以:
Manage m=new Employee();
但可以进行强制转换（不推荐使用）

3、作用

消除类型之间的耦合关系，,提高程序的可扩展性.。

多态的好处:

1.可替换性（substitutability）。多态对已存在代码具有可替换性。例如，多态对圆Circle类工作，对其他任何圆形几何体，如圆环，也同样工作。

2.可扩充性（extensibility）。多态对代码具有可扩充性。增加新的子类不影响已存在类的多态性、继承性，以及其他特性的运行和操作。实际上新加子类更容易获得多态功能。例如，在实现了圆锥、半圆锥以及半球体的多态基础上，很容易增添球体类的多态性。
3.接口性（interface-ability）。多态是超类通过方法签名，向子类提供了一个共同接口，由子类来完善或者覆盖它而实现的。如图8.3 所示。图中超类Shape规定了两个实现多态的接口方法，computeArea()以及computeVolume()。子类，如Circle和Sphere为了实现多态，完善或者覆盖这两个接口方法。
4.灵活性（flexibility）。它在应用中体现了灵活多样的操作，提高了使用效率。
5.简化性（simplicity）。多态简化对应用软件的代码编写和修改过程，尤其在处理大量对象的运算和操作时，这个特点尤为突出和重要。

4、实现技术

动态绑定（dynamic binding），是指在执行期间判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。

5、实现方法

- 子类继承父类（extends）
- 类实现接口（implements）

二、Java实现多态有三个必要条件：

1、继承：

在多态中必须存在有继承关系的子类和父类。

2、重写：

子类对父类中某些方法进行重新定义，在调用这些方法时就会调用子类的方法。

3、 向上转型（父类引用指向子类对象）

在多态中需要将子类的引用赋给父类对象，只有这样该引用才能够具备技能调用父类的方法和子类的方法。

三、实现形式

1、基于继承实现的多态

注意：假设 Object、Wine、JGJ三者继承链关系是：JGJ—>Wine—>Object

当子类重写父类的方法被调用时，只有对象继承链中的最末端的方法才会被调用

若：

```
Object o = new JGJ();
    System.out.println(o.toString());
```

输出的结果是Wine ： JGJ。

但是注意如果这样写：

```
 Object o = new Wine();
System.out.println(o.toString());
```

输出的结果应该是Null，因为JGJ并不存在于该对象继承链中。

2、基于接口实现的多态

四、多态的实现机制遵循一个原则：

当超类对象引用变量引用子类对象时，被引用对象的类型而不是引用变量的类型决定了调用谁的成员方法，但是这个被调用的方法必须是在超类中定义过的，也就是说被子类覆盖的方法。

五、总结

指向子类的父类引用由于向上转型了，它只能访问父类中拥有的方法和属性，而对于子类中存在而父类中不存在的方法，该引用是不能使用的，尽管是重载该方法。若子类重写了父类中的某些方法，在调用该些方法的时候，必定是使用子类中定义的这些方法（动态连接、动态调用）。