两种都是多态

1.父类 引用=new 子类();

2.接口 引用=new 接口实现();

A.接口的实现也是继承

B.方法的实现也是重写

C.一个引用，引用不同类的实例，实现不同的行为,这就是多态

二者有何区别，接口的设计主要还是为了实现了多态的，

当然抽象类也可以这么用，但是因为一个类只能继承一个类，如果把抽象类拿来当做接口使用，那么多态就是笑话了。

经常见:

List<?> list=new ArrayList<?>;

Map<?,?> map=new HashMap<?,?>;

但是发现list只是ArrayList的接口，不是它的父类，不是父类引用指向子类对象

如果是AbstractList ablist=new ArrayList();

或者直接写为ArrayList<?> list=new ArrayList<?>;

为什么要用接口引用指向实现类的对象

这种写法其实是java多态的表现形式

1. 多态的定义：指允许不同类的对象对同一消息作出相应，即同一消息可以根据发送对象的不同而采用多种不同的行为方式.（发送消息就是函数调用）

List list;是在栈区开辟一个空间放list引用，并没有创建对象，所以不知道ArrayList还是LinkedList，当你list=new ArrayList();就创建了ArrayList对象，并且把开始创建的list引用指向这个对象ArrayList和LinkedList都是List的实现类；

为什么一般都使用List list=new ArrayList();而不用

ArrayList alist=new ArrayList()呢？

问题就在于List有多个实现类，如LinkedList或者Vector等等，现在你用的是ArrayList，也许哪一天你需要更换成其它的实现类呢？这时你只要改变一行就行了：List list=new LinkedList();

其它使用了list地方的代码根本不需要改动。假设你开始用ArrayList alist=new ArrayList()；这下你有的改了，特别是如果你使用了ArrayList特有的方法和属性，如果没有特别需求的话，最好使用List list=new LinkedList();便于程序代码的重构，这就是面向接口编程的好处

注意事项

list只能使用ArrayList中已经实现了List接口的方法，ArrayList中那些自己的，没有在List接口定义的方法是不可以被访问到的

List.add() 其实是list接口的方法

如果调用ArrayList的方法：如clone()方法是调用不到的

接口的灵活性就在于规定一个类必须做什么，而不管你如何做。我们可以定义一个接口类型的引用变量来引用实现接口的类的实例，当这个引用调用方法时，它会根据实际引用的类的实例来判断具体调用哪个方法，这和上述的超类对象引用访问子类对象的机制相似。

需要注意一点是：Java在利用接口变量调用其实现类的对象的方法时，该方法必须已经在接口中被声明，而且在接口的实现类中该实现方法的类型和参数必须与接口中所定义的精确匹配。