前提：

1. 要有**继承**关系。
2. 子类要**重写**父类的方法。
3. 父类**引用**指向子类。

实现方式：

1. **接口实现**。
2. 继承父类进行方法**重写**。（动态）
3. 同一个类中进行方法**重载**。（静态）

要点：

1. 多态是方法的多态，不是属性的多态(多态与属性无关)。

2. 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）。

3. 父类引用指向子类对象后，用该父类引用调用子类重写的方法，此时多态就出现了。

引用对象的实际类型而不是引用变量的类型决定了调用谁的成员方法，（一般调用子类的方法**见）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。如果这个方法没有被重写（父类无定义），方法调用的优先级：show（方法）

①this.show（object）>②super.show（object）

> ③this.show（（super）object）>④super.show（（super）object）

即先查this对象的父类，没有重头再查参数的父类。

多态和类型转换测试

```java
class Animal {
    public void shout() {
        System.out.println("叫了一声！");
    }
}
class Dog extends Animal {
    public void shout() {
        System.out.println("旺旺旺！");
    }
    public void seeDoor() {
        System.out.println("看门中....");
    }
}
class Cat extends Animal {
    public void shout() {
        System.out.println("喵喵喵喵！");
    }
}
public class TestPolym {
    public static void main(String[] args) {
        Animal a1 = new Cat(); // 向上可以自动转型
        //传的具体是哪一个类就调用哪一个类的方法。大大提高了程序的可扩展性。
        animalCry(a1);
        Animal a2 = new Dog();
        animalCry(a2);//a2为编译类型，Dog对象才是运行时类型。

        //编写程序时，如果想调用运行时类型的方法，只能进行强制类型转换。
        // 否则通不过编译器的检查。
        Dog dog = (Dog)a2;//向下需要强制类型转换
        dog.seeDoor();
    }

    // 有了多态，只需要让增加的这个类继承Animal类就可以了。
    static void animalCry(Animal a) {
        a.shout();
    }

    /* 如果没有多态，我们这里需要写很多重载的方法。
     * 每增加一种动物，就需要重载一种动物的喊叫方法。非常麻烦。
    static void animalCry(Dog d) {
        d.shout();
    }
    static void animalCry(Cat c) {
        c.shout();
    }*/
}
```

向上转型：，父类类型的引用可以调用父类中定义的所有属性和方法，却不能调用子类中的方法和属性。

向下转型：父类引用要向下转型必须其实际类型为子类类型。为了解决不安全的向下转型引入泛型概念。