1. 多态的定义

多态，从名字可以看出，是 同一个对象在不同时可表现出来的不同形态。

如：猫是一种动物，也是一个具体的物种猫。

cat = new cat();

animal = new cat();

猫在不同的场合可以被形容为不同的对象，既可以是猫，也可以是动物，这就是多态。

1. 多态的前提
2. 继承关系
3. 方法重写
4. 父类引用指向子类对象
5. 多态中成员访问特点
6. 成员变量的访问

无法通过多态访问父类中不存在的成员变量，只能访问父类中存在的成员变量。因为多态是父类的引用指向子类的对象，即

Animal a = new Cat();

创建了子类对象Cat，多态表示为Animal，那么只有Animal父类中的成员变量才能够访问到，而子类中的特有的成员变量是无法访问到的。

1. 成员方法的访问

成员方法是可以重写的，那么同样是创建了子类对象Cat,多态表示为Animal。

Animal a = new Cat();

通过多态可以调用子类中已经重写的成员方法，但是和成员变量一样，无法访问子类中特有的成员方法。

1. 多态中的转型
2. 向上转型

所谓向上，即子类转为父类，通常用的父类的引用指向子类对象就是向上转型。如：

Anima a = new Cat();

创建了子类对象，并将它赋值给了父类对象，即子类转为父类。

1. 向下转型

所谓向下，即父类转为子类，将父类对象强转为子类对象。如：

Animal a = new Cat();

Cat c = (Cat)a;

父类对象a被强转为了子类对象c。

1. 多态的利弊
2. 多态的优点

多态能够提高程序的扩展性，当我们定义一个方法时，只需要使用父类型作为参数，使用的时候，传入具体的子类型就可以实现我们的需求。特别是对于多种相似的子类重复操作时，非常方便。

1. 多态的缺点

多态只能使用父类中的成员变量，无法使用子类中特有的成员变量。

多态只能使用父类和子类中重写的成员方法，无法使用子类中特有的成员方法。