Java面向对象多态

多态概述

Java 多态是指同一种类型的对象，在不同的情况下有着不同的状态和行为。它是基于继承、重写和向上转型等特性实现的，多态是面向对象编程的三大特征之一，其他两个分别是封装和继承。

Java 多态包括以下三种方式

方法重写（Override）

子类可以对从父类继承的方法进行重写，以便根据子类的需要来覆盖掉父类中的方法实现。这样在调用时，可以根据对象的不同来执行对应的方法。

例如：

Java | 复制代码

```java
class Animal {
    public void move() {
        System.out.println("动物移动");
    }
}

class Cat extends Animal {
    @Override
    public void move() {
        System.out.println("猫在走路");
    }
}

public class Test {
    public static void main(String[] args) {
        Animal animal = new Animal();
        Animal cat = new Cat();
        animal.move();
        cat.move();
    }
}

```

向上转型（Upcasting）

将子类对象转换成父类引用，这样就可以使用父类中定义的方法和属性，但不能访问子类独有的方法和属性。这种特征使得可以将子类对象作为父类的参数进行传递，提高代码的灵活性。

例如：

Java | 复制代码

```java
class Animal {
    public void move() {
        System.out.println("动物移动");
    }
}

class Cat extends Animal {
    @Override
    public void move() {
        System.out.println("猫在走路");
    }
    public void catchMouse() {
        System.out.println("猫在抓老鼠");
    }
}

public class Test {
    public static void main(String[] args) {
        Animal animal = new Cat();
        animal.move(); // 调用子类中的方法
        //animal.catchMouse(); // 编译错误，不能访问子类独有的方法
    }
}
```

实现多态

通过父类引用指向子类对象，可以实现多态性，使得同一个方法调用可以在不同的对象上具有不同的行为。这种特征使得程序更加灵活，可以根据需要来确定对象的类型。

例如：

Java | 复制代码

```java
class Animal {
    public void move() {
        System.out.println("动物移动");
    }
}

class Cat extends Animal {
    @Override
    public void move() {
        System.out.println("猫在走路");
    }
}

class Dog extends Animal {
    @Override
    public void move() {
        System.out.println("狗在跑步");
    }
}

public class Test {
    public static void main(String[] args) {
        Animal animal1 = new Cat();
        Animal animal2 = new Dog();
        animal1.move(); // 同一种方法调用不同的对象
        animal2.move(); // 同一种方法调用不同的对象
    }
}

```