Java 中多态性之解析

张国梁
(四川化工职业技术学院信息工程系，泸州 646005)

摘 要：面向对象编程 (Object Oriented Programming，简称 OOP) 描述的是对象之间的相互作用。在面向对象编程中，类作为最小程序单元，就像以往面向过程编程中，函数作为最小程序单元一样。多态性是面向对象的重要特性之一，Java 中的多态体现在类的继承和实现接口等方面。在使用 Java 时理解和掌握多态性是非常重要的，对以后学习和掌握其他面向对象的编程技术也尤为重要。

关键词：Java；多态性；软件开发

Brief Analysis of Multi-State In Java

ZHANG Guoliang
(Department of Information Engineering Sichuan Chemical Technical College , Luzhou 646005)

Abstract: Object Oriented Programming (Object Oriented Programming, referred to as OOP) describes the interaction between objects. In object-oriented programming, class as the minimum program unit, just like the past, process-oriented programming, the same function as the minimum program unit. Multi-state is an important feature of object-oriented, Java in the multi-state embodied in the class inheritance and implement the interface and so on. so to understand and master the Multi-State is very important. When you are using Java,and for further study and master the other object-oriented programming technology is particularly important.

Key words: Java ; Multi-State ; Software Development

1 多态

假设有一个“小提琴演奏者”类，有一个“钢琴演奏者”类，还有一个“萨克斯演奏者”类。它们都是弹奏乐器的，都继承自“乐器演奏者”类。“乐器演奏者”类有个方法叫做“演奏”。3 种不同的子类有不同的“演奏”方法实现 (拉、弹、吹)。

那么指挥家如果要指挥一个交响乐，有两个办法：

(1) 分别是“小提琴拉起来”、“钢琴弹起来”、“萨克斯吹起来”。

(2) 只需要说一句“乐器们都演奏起来”。

显然第一种方法是很笨拙的。因为指挥家不需要知道每种乐器演奏者具体怎么样去演奏乐器 (是吹还是弹还是拉)，只需要指挥就可以了。因此应当采用第二种做法。

2 必要性

通过以下示例来分析：

需求：在教师类的基础上，开发一个类代表教育局，负责对上例中的各教师进行评估，评估内容包括：(1) 教师的自我介绍；(2) 教师的授课。

第一种解决方案

```
public class Education {
public void judge (MathTeacher t) {
t.introduction () ;
t.giveLesson () ;
}
public void judge (PhysicsTeacher t) {
t.introduction () ;
t.giveLesson () ;
}
public static void main (String [] args) {
Education hq = new Education () ;
hq.judge (new MathTeacher (" 李明" ," 成都七中")) ;
hq.judge (new PhysicsTeacher (" 张三" ," 川师附中")) ;
}
}
```

此方案的问题：需要为每种类型的教师建立一个 judge 方法，其参数类型为不同类型的教师类，这样势必存在大量重复代码。假如再增加一个英语教师类 (EnglishTeacher)，教育局需要对这种类型的教师进行评估，那么就需要在 Education 类中增加一个 public void judge (EnglishTeacher t) 方法，由此每增加一种教师都会产生重复代码，增加相应的 judge (评估) 方法，代码的可扩展性及可维护性极差，违背了“write once, only once”的原则。为了解决此问题，可以考虑使用多态。

3 实现

3.1 继承

继承是指一个对象从另一个对象中获得属性的过程，是面向对象程序设计的三大原则之一，它支持按层次分类的概念。例如，东北虎是虎的一种，虎又是哺乳动物的一种，哺乳动物又是动物的一种。如果不使用层次的概念，每个对象需要明确定义各自的全部特征。通过层次分类方式，一个对象只需要在它的类中定义它成为唯一的各个属性，然后从父

作者简介：张国梁 (1984-)，男，学士，助教，研究方向：应用软件开发。

收稿日期：2010-10-10

类中继承它的通用属性。因此，正是由于继承机制，才使得一个对象可以成为一个通用类的一个特定实例。一个深度继承的子类将继承它在类层次中的每个祖先的所有属性。

由图 1 可以看出，子类具有父类的一般特性（包括属性和行为），又具有自身特殊的特性。继承属于 is-a 关系（子类对象总属于一个父类对象）：父类更通用，子类更具体。

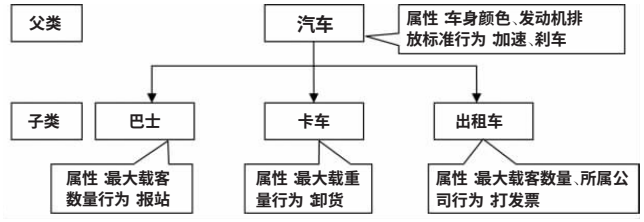

图 1

如果类之间具有继承关系，那么就用继承来实现多态，其步骤为：

（1）子类重写父类的方法。

（2）编写方法时，在方法体内调用父类定义的方法（即父类作为参数类型，去调用此类型对象的方法）。

（3）运行时，根据实际传递的对象类型动态决定使用哪个方法。

由于上例中类之间具有继承关系，修改上述 Education 类为：

```
public class Education {
/** 父类作为形参，可以接收所有子类类型对象，根据实际传递的对象类型调用相应方法 */
public void judge (Teacher t) {
t.introduction () ;
t.giveLesson () ;
}
public static void main (String [] args) {
Education hq = new Education () ;
hq.judge (new MathTeacher (" 李明" ," 成都七中")) ;
hq.judge (new PhysicsTeacher (" 张三" ," 川师附中")) ;
hq.judge (new EnglishTeacher (“李四" ," 北大附中")) ;
}
}
```

从此例可以看出，使用多态之后，当需要增加新的子类类型时，无需更改 Education 类，这样大大增强了程序的可扩展性及可维护性。

3.2 接口

首先了解什么是接口：例如电脑主板上的 PCI 插槽的规范就类似于 Java 接口，PCI 插槽上可以插上声卡、网卡、显卡，每种卡的内部结构都不相同，可以把声卡、网卡、显卡都插在 PCI 插槽上，而不用担心哪个插槽是专门插哪个卡的。在这里 PCI 插槽作为接口只提供一个标准（即各种卡 I/O 接口的标准），只要符合此标准的电子配件都能连接上去，连接不同的电子配件，此 PCI 插槽就具有不同的功能，而不要关心其电子配件内部功能的实现原理，这就是使用接口技术实现多态性的机制。那么这里的 PCI 插槽就是接口（提供标准），各种卡是实现了此接口的类（必须符合接口所定义的标准）。

用程序的方式来描述此问题：

```
/** 这是 Java 接口，相当于主板上的 PCI 插槽的规范，是一些方法特征的集合，但没有方法的实现，interface 为定义接口的关键字 */
public interface PCI {
public void start () ;
public void stop () ;
}
/*Java 接口中定义的方法在不同的地方被实现，可以具有完全不同的行为，这里的 SoundCard、NetworkCard 就是这样，他们都遵守了 PCI 插槽的规范，但行为完全不同 implements 为类实现接口标准的关键字 */
class SoundCard implements PCI {
public void start () {
System.out.println (" Du du...") ;
}
public void stop () {
System.out.println (" Sound stop!") ;
}
}
class NetworkCard implements PCI {
public void start () {
System.out.println (" Send...") ;
}
public void stop () {
System.out.println (" Network stop!") ;
}
}
public class mainboard {
/* 可以使用 Java 接口类型作为方法形参，运行时，根据实际传递的对象类型（此对象必须符合形参中 Java 接口的标准，即实现了此 Java 接口）调用相应的方法实现 */
public void install (PCI card) {
card.start () ;
card.stop () ;
}
public static void main (String [] args) {
Mainboard mb=new mainboard () ;
mb. Install (new NetworkCard ()) ;
mb. Install (new SoundCard ()) ;
}
}
```

在这个示例中，SoundCard 与 NetworkCard 存在继承关系吗？无法通过“is-a”找到一个合适的父类，显然不存在继承关系，所以只能使用接口技术来实现多态。可以看出在使用接口技术实现多态时，接口相当于继承关系中的父类的地位，因此得首先抽象出接口来代替父类在继承关系中的地位，另外某个方法得使用接口类型作为形参，并且此方法体内调用了接口中申明的方法，那么可以传递实现了此接口的任何类的对象给它，这样，根据传递的不同类型的对象执行不同的方法实现，以实现多态。

参考文献

[1] 孙卫琴. Java 面向对象编程. 电子工业出版社，2006.

[2] 甘玲，等. 解析 Java 程序设计. 清华大学出版社，2006.

[3] 秦凤梅，林旭东. Java 程序设计实用教程. 西南师范大学出版社，2006.