16 | 设计模式基础：不会灵活应用设计模式，你就没有掌握面向对象编程

2019-12-27 李智慧

后端技术面试38讲 进入课程 >

讲述：李智慧
时长 11:07 大小 10.19M

我在面试的时候，喜欢问一个问题：“你比较熟悉哪些设计模式？”得到的回答很多时候是“单例”和“工厂”。老实说，这个回答不能让人满意。因为在我看来，单例和工厂固然是两种经典的设计模式，但是，这些创建类的设计模式并不能代表设计模式的精髓。

设计模式的精髓在于对面向对象编程特性之一——多态的灵活应用，而多态正是面向对象编程的本质所在。

面向对象编程的本质是多态

我在面试时，有时候会问“什么是对象”，得到的回答各种各样：“对象是数据与方法的组合。”“对象是领域的抽象。”“一切都是对象。”“对象的特性就是封装、继承、多态。”

这是一个比较开放的问题，这些回答可以说都是对的，都描述了对象的某个方面。那么，面向对象的本质是什么？面向对象编程和此前的面向过程编程的核心区别是什么？

我们常说，面向对象编程的主要特性是封装、继承和多态。那么这三个特性是否是面向对象编程区别于其他编程技术的关键呢？

我们先看封装，面向对象编程语言都提供了类的定义。通过类，我们可以将类的成员变量和成员方法封装起来，还可以通过访问控制符，private、protected、public 控制成员变量和成员方法的可见性。

面向对象设计最基本的设计粒度就是类。类通过封装数据和方法，构成一个相对独立的实体。类之间通过访问控制的约束互相调用，这样就完成了面向对象的编程。但是，封装并不是面向对象编程语言独有的。面向过程的编程语言，比如 C 语言，也可以实现封装特性，在头文件.h 里面定义方法，而在实现文件.c 文件里定义具体的结构体和方法实现，从而使依赖.h 头文件的外部程序只能够访问头文件里定义过的方法，这样同样实现了变量和函数的封装，以及访问权限的控制。

继承似乎是面向对象编程语言才有的特性，事实上，C 语言也可以实现继承。如果 A 结构体包含 B 结构体的定义，那么就可以理解成 A 继承了 B，定义在 B 结构上的方法可以直接（通过强制类型转换）执行 A 结构体的数据。

作为一种编程技巧，这种通过定义结构体从而实现继承特性的方法，在面向对象编程语言出现以前就已经经常被开发者使用了。

我们再来看多态，因为有指向函数的指针，多态事实上在 C 语言中也可以实现。但是使用指向函数的指针实现多态是非常危险的，因为这种多态没有语法和编译方面的约束，只能靠程序员之间约定，一旦出现 bug，调试非常痛苦。因此在面向过程语言的开发中，这种多态并不能频繁使用。

而在面向对象的编程语言中，多态非常简单：子类实现父类或者接口的抽象方法，程序使用抽象父类或者接口编程，运行期注入不同的子类，程序就表现出不同的形态，是为多态。

这样做最大的好处就是软件编程时的实现无关性，程序针对接口和抽象类编程，而不需要关心具体实现是什么。你应该还记得我在第 10 篇中讲到的案例：对于一个从输入设备拷贝字符到输出设备的程序，如果具体的设备实现和拷贝程序是耦合在一起的，那么当我们想要增加任何输入设备或者输出设备的时候，都必须要修改程序代码，最后这个拷贝程序将会变得越来越复杂、难于使用和理解。

而通过使用接口，我们定义了 Reader 和 Writer 两个接口，分别描述输入设备和输出设备，拷贝程序只需要针对这两个接口编程，而无需关心具体设备是什么，程序可以保持稳定，并且易于复用。具体设备在程序运行期创建，然后传给拷贝程序，传入什么具体设备，就在什么具体设备上操作拷贝逻辑，具体设备可以像插件一样，灵活插拔，使程序呈现多态的特性。

多态还颠覆了程序模块间的依赖关系。在习惯的编程思维中，如果 A 模块调用 B 模块，那么 A 模块必须依赖 B 模块，也就是说，在 A 模块的代码中必须 import 或者 using B 模块的代码。但是通过使用多态的特性，我们可以将这个依赖关系倒置，也就是：A 模块调用 B 模块，A 模块却可以不依赖 B 模块，反而是 B 模块依赖 A 模块。

这就是我在第 12 篇中提到的依赖倒置原则。准确地说，B 模块也没有依赖 A 模块，而是依赖 A 模块定义的抽象接口。A 模块针对抽象接口编程，调用抽象接口，B 模块实现抽象接口。在程序运行期将 B 模块注入 A 模块，就使得 A 模块调用 B 模块，却没有依赖 B 模块。

多态常常使面向对象编程表现出神奇的特性，而多态正是面向对象编程的本质所在。正是多态，使得面向对象编程和以往的编程方式有了巨大的不同。

设计模式的精髓是对多态的使用

但是就算知道了面向对象编程的多态特性，也很难利用好多态的特性，开发出强大的面向对象程序。到底如何利用好多态特性呢？人们通过不断的编程实践，总结了一系列的设计原则和设计模式。

我们前面几篇文章都是讨论设计原则的：

1. 开闭原则：软件类、模块应该是对修改关闭的，而对扩展是开放的。通俗地说，就是要不修改代码就是实现需求的变更。
2. 依赖倒置原则：高层模块不应该依赖低层模块，低层模块也不应该依赖高层模块，他们应该都依赖抽象，而这个抽象是高层定义的，逻辑上属于高层。
3. 里氏替换原则：所有能够使用父类的地方，应该都可以用它的子类替换。但要注意的是，能不能替换是要看应用场景的，所以在设计继承的时候就要考虑运行期的场景，而不是仅仅考虑父类和子类的静态关系。
4. 单一职责原则：一个类应该只有一个引起它变化的原因。实践中，就是类文件尽量不要太大，最好不要超过一屏。
5. 接口隔离原则：不要强迫调用者依赖他们不需要的方法。方法主要是通过对接口的多重继承，一个类实现多个接口，不同接口服务不同调用者，不同调用者看到不同方法。

这些设计原则大部分都是和多态有关的，不过这些设计原则更多时候是具有指导性，编程的时候还需要依赖更具体的编程设计方法，这些方法就是设计模式。

模式是可重复的解决方案，人们在编程实践中发现，有些问题是重复出现的，虽然场景各有不同，但是问题的本质是一样的，而解决这些问题的方法也是可以重复使用的。人们把这些可以重复使用的编程方法称为设计模式。**设计模式的精髓就是对多态的灵活应用**。

我们以装饰模式为例，看一下如何灵活应用多态特性。我们先定义一个接口 AnyThing，包含一个 exe 方法。

复制代码

```
public interface AnyThing {
  void exe();
}
```

然后多个类实现这个接口，装饰模式最大的特点是，通过类的构造函数传入一个同类对象，也就是每个类实现的接口和构造函数传入的对象是同一个接口。我们创建了三个类，如下：

复制代码

```
public class Moon implements AnyThing {
  private AnyThing a;
  public Moon(AnyThing a) {
```

```
    this.a = a;
  }
  public void exe() {
    System.out.print(" 明月装饰了 ");
    a.exe();
  }
}

public class Dream implements AnyThing {
  private AnyThing a;
  public Dream(AnyThing a) {
    this.a=a;
  }
  public void exe() {
    System.out.print(" 梦装饰了 ");
    a.exe();
  }
}

public class You implements AnyThing {
  private AnyThing a;
  public You(AnyThing a) {
    this.a = a;
  }
  public void exe() {
    System.out.print(" 你 ");
  }
}
```

设计这个几个类的时候，它们之间没有任何耦合，但是在创建对象的时候，我们通过构造函数的不同次序，可以使这几个类互相调用，从而呈现不同的装饰结果。

复制代码

```
AnyThing t = new Moon(new Dream(new You(null)));
t.exe();

输出：明月装饰了梦装饰了你


AnyThing t = new Dream(new Moon(new You(null)));
t.exe();

输出：梦装饰了明月装饰了你
```

多态的迷人之处就在于，你单独看类的代码的时候，这些代码似乎平淡无奇，但是一旦运行起来，就会表现出纷繁复杂的特性。所以多态有时候也会带来一些代码阅读方面的困扰，让面向对象编程的新手望而却步，这也正是设计模式的作用，这时候你仅仅通过类的名字，比如 Observer、Adapter，你就能知道设计者在使用什么模式，从而更快速理解代码。

小结

如果你只是使用面向对象编程语言进行编程，其实并不能说明你就掌握了面向对象编程。只有灵活应用设计模式，使程序呈现多态的特性，进而使程序健壮、灵活、清晰、易于维护和复用，这才是真正掌握了面向对象编程。

所以，下次再有面试官让你“聊聊设计模式”，也许你可以这样回答：“除了单例和工厂，我更喜欢适配器和观察者，还有，组合模式在处理树形结构的时候也非常有用。”适配器和观察者模式我在前面已经讲到。

设计模式是一个非常注重实践的编程技能，通过学习设计模式，我们可以体会到面向对象编程的种种精妙。真正掌握设计模式，需要在实践中不断使用它，让自己的程序更加健壮、灵活、清晰、易于复用和扩展。这个时候，面试聊设计模式更好的回答是：“我在工作中比较喜欢用模板模式和策略模式，上个项目中，为了解决不同用户使用不同推荐算法的问题，我……”

事实上，设计模式不仅仅包括《设计模式》这本书里讲到的 23 种设计模式，只要可重复用于解决某个问题场景的设计方案都可以被称为设计模式。关于设计模式还有一句很著名的话“精通设计模式，就是忘了设计模式”，有点像张无忌学太极。如果真正对设计模式融会贯通，你的程序中无处不是设计模式，也许你在三五行代码里，就用了两三个设计模式。你自己就是设计模式的大师，甚至还可以创建一些自己的设计模式。这个时候，再去面试的时候，面试官也不会再问你设计模式的问题了，如果问了，那么你说什么都是对的。

思考题

我在第 2 篇文章和本篇中都提到了可以使用组合模式遍历树，那么如何用组合模式遍历树呢？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　15 | 软件设计的接口隔离原则：如何对类的调用者隐藏类的公有方法?

下一篇　17 | 设计模式应用：编程框架中的设计模式

精选留言 (3)

写留言

蓝魔丶
2019-12-27

每个子类构造函数传入父类实现组合，通过递归获取父类

1

Jesse
2019-12-30

使用组合模式遍历树：我的想法是，每棵树都可以看成是根节点和子树（左子树，右子树）的组合。因此在定义节点Node的时候，就可以持有两个子树根节点的的引用（left,right），这也是一般链式结构树的实现，用到的就是组合模式。而树的遍历，就可以递归的遍历所有的子树了。

展开

作者回复: 问题是：不用递归如何遍历树?

尹宗昌

2019-12-27

无招胜有招

展开