20个设计模式和软件设计面试问题

2014/08/13 | 分类： [基础技术](http://www.importnew.com/cat/basic) | [2 条评论](http://www.importnew.com/12649.html#comments) | 标签： [设计模式](http://www.importnew.com/tag/%e8%ae%be%e8%ae%a1%e6%a8%a1%e5%bc%8f), [面试问题](http://www.importnew.com/tag/%e9%9d%a2%e8%af%95%e9%97%ae%e9%a2%98)

分享到：**105**

本文由 [ImportNew](http://www.importnew.com/) - [陈 晓舜](http://www.importnew.com/author/chenxiaoshun) 翻译自 [javarevisited](http://javarevisited.blogspot.tw/2012/06/20-design-pattern-and-software-design.html)。欢迎加入[翻译小组](http://group.jobbole.com/category/feedback/trans-team/)。转载请见文末要求。

不管是参加Java面试还是C#面试，[**设计模式**](http://www.amazon.cn/gp/product/B001130JN8/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&tag=importnew-23&linkCode=as2&camp=536&creative=3200&creativeASIN=B001130JN8)**和软件设计**都是任何编程面试中的必问问题。实际上，编程能力和设计技巧是对彼此很好的补充。一个好的程序员通常都是一个好的软件设计人员。他们知道怎么把一个问题分割成一段段代码或者软件设计，但这些能力和技巧并不能凭空而来。你需要持续做大型、小型系统的设计和编码，并且不断从错误中学习。从[面向对象设计原则](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html) 开始是一个不错的选择。好吧，这篇文章是关于一些在各种面试中频繁问到的一些设计问题。考虑到针对性和困难度，我把它们分成两个种类，分别针对初学者和有经验的人。

对中高级别的设计模式面试问题

这是一些和设计模式还有软件设计都相关的问题。这些问题需要一些思考和经验来回答。在大多数情况下，面试官并不是需要一个确切的答案，而是希望听到你的想法，你对这个问题是怎么考虑的，你能不能想通这个问题，能不能挖掘一些没有告诉你的潜在信息。在解决一个问题时你应该考虑什么等等可以使你经验大涨。总的来说，这些设计问题会让你动起脑来。有时面试官也会让你写代码，所以还是准备一下吧。如果你知道编程和设计技巧中的概念，例子和程序，你应该可以在这些问题中有突出的表现。

**1. 举出一个例子，在这种情况你会更倾向于使用抽象类，而不是接口？**

这是很常用但又是很难回答的设计面试问题。接口和抽象类都遵循”面向接口而不是实现编码”设计原则，它可以增加代码的灵活性，可以适应不断变化的需求。下面有几个点可以帮助你回答这个问题：

1. 在Java中，你只能继承一个类，但可以实现多个接口。所以一旦你继承了一个类，你就失去了继承其他类的机会了。
2. 接口通常被用来表示附属描述或行为如：Runnable、Clonable、Serializable等等，因此当你使用抽象类来表示行为时，你的类就不能同时是Runnable和Clonable(译者注：这里的意思是指如果把Runnable等实现为抽象类的情况)，因为在Java中你不能继承两个类，但当你使用接口时，你的类就可以同时拥有多个不同的行为。
3. 在一些对时间要求比较高的应用中，倾向于使用抽象类，它会比接口稍快一点。
4. 如果希望把一系列行为都规范在类继承层次内，并且可以更好地在同一个地方进行编码，那么抽象类是一个更好的选择。有时，接口和抽象类可以一起使用，接口中定义函数，而在抽象类中定义默认的实现。

希望了解更多关于Java接口的，可以看我的文章[Java接口需要知道的10件事](http://javarevisited.blogspot.com/2012/04/10-points-on-interface-in-java-with.html)。

**2. 设计一个贩卖机，可以接收不同的货币，出售不同的产品？**

这是一个开放设计问题，你可以作为练习，尝试着写出设计文档、代码和JUnit测试而不是只是解决这个问题，看看它花了你多少时间得到解决方案和得到需要的原形。理想情况下，这个问题应该可以在3个小时内解决，至少应该得到一个可以运行的版本。

**3. 你有一个Smartphone类，可以派生如IPhone、AndroidPhone、WindowsMobilePhone类**

它还可以是一些有着品牌的手机名称，你会怎么设计这个类系统呢。

这是另外一个设计模式练习，你可以应用你的面向对象设计技巧来得到一个设计，这个设计需要足够灵活能够支持未来产品的扩展，足够稳定能够支持在现有模型进行修改。

**4. 在Java中，什么时候用重载，什么时候用重写？**

对有经验的Java设计师来说，这是一个相当简单的问题。如果你看到一个类的不同实现有着不同的方式来做同一件事，那么就应该用重写（overriding），而重载（overloading）是用不同的输入做同一件事。在Java中，重载的方法签名不同，而重写并不是。

**5. 设计一个ATM机？**

我们所有人都使用*ATM(自动柜员机)*。想想你会怎么设计一个ATM？就设计金融系统来说，必须知道它们应该在任何情况下都能够如期工作。不管是断电还是其他情况，ATM应该保持 **正确的状态（事务）** , 想想 **加锁（locking）、事务（transaction）、错误条件（error condition）、边界条件（boundary condition）** 等等。尽管你不能想到具体的设计，但如果你可以指出非功能性需求，提出一些问题，想到关于边界条件，这些都会是很好的一步。

**6. 你正在写一些类提供市场数据，你知道你可以不定时切换不同的厂商如Reuters、wombat或者直接的批发商， 你会如何设计你的市场数据系统。**

这是一个非常有趣的设计面试问题，并且真的在一家大的投资银行问到过，如果你是用Java编码的话这是一个相当平常的场景。最主要的一点是你要有一个MarketData接口，它会有调用端需要的方法如：getBid()、getPrice()、getLevel()等等，而MarketData应该由一个MarketDataProvider通过 **依赖注入（dependency injection）** 组成。因此，当你修改你的MarketData 提供器（MarketDataProvider）时，调用端不会受影响，因为它们是通过MarketData接口或类的方法来访问的。

**7. 在Java中，为什么不允许从静态方法中访问非静态变量？**

你在Java中不能从静态上下文访问非静态数据只是因为非静态变量是跟具体的对象实例关联的，而静态的却没有和任何实例关联。你可以看我的文章[为什么在静态上下文中不能访问非静态变量](http://javarevisited.blogspot.com/2012/02/why-non-static-variable-cannot-be.html)查看详细的讨论。

**8. 在Java中设计一个并发规则的pipeline？**

**并发编程**或**并发设计**这些天很火，它可以充分利用现在不断提升的高级处理器的处理能力，而Java成为一个多线程语言也从这种情况获益良多。设计一个并发系统需要记住的最关键的点是[线程安全](http://javarevisited.blogspot.com/2012/01/how-to-write-thread-safe-code-in-java.html)，不可变性，本地变量和避免使用static或者[类变量（instance variables）](http://javarevisited.blogspot.com/2012/02/difference-between-instance-class-and.html)。你只需要想着每一类都可以同时被多个线程同时执行，所以最好的做法就是每一个线程都处理自己的数据 ，不跟其他数据交互，并且运行时只需要最小的同步保证。这个问题可以涉及到从最初的讨论到完整的类和接口编码，但只要你记住并发中最重要的点和问题如，[竞争条件（race condition）](http://javarevisited.blogspot.com/2012/02/difference-between-instance-class-and.html)、[死锁（deadlock）](http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html)、内存交互问题（memory interference）、原子性、[ThreadLocal变量](http://javarevisited.blogspot.com/2012/05/how-to-use-threadlocal-in-java-benefits.html)等，你都可以回答它。

给初学者的设计模式面试问题

这些软件设计和设计模式问题大多在初学者层次时被问起，目的只是了解一下候选人(应聘者)对设计模式知道多少，如， **设计模式是什么** 或者 **一个特定的设计模式做什么** ?这些问题通过简单地记忆概念就可以回答，但就信息和知识而言还是有价值的。

**1. 什么是设计模式？你是否在你的代码里面使用过任何设计模式？**

设计模式是世界上各种各样程序员用来解决特定设计问题的尝试和测试的方法。设计模式是代码可用性的延伸。

**2. 你可以说出几个在JDK库中使用的设计模式吗？**

[装饰器设计模式（Decorator design pattern）](http://javarevisited.blogspot.com/2011/11/decorator-design-pattern-java-example.html)被用于多个Java IO类中。单例模式（Singleton pattern）用于Runtime，Calendar和其他的一些类中。工厂模式（Factory pattern）被用于各种不可变的类如Boolean，像Boolean.valueOf，观察者模式（Observer pattern）被用于Swing和很多的事件监听中。

**3. Java中什么是单例设计模式？用Java写出线程安全的单例**

单例模式重点在于在整个系统上共享一些创建时较耗资源的对象。整个应用中只维护一个特定类实例，它被所有组件共同使用。Java.lang.Runtime是单例模式的经典例子。你可以在我的文章[Java单例模式的10个问题](http://javarevisited.blogspot.com/2011/03/10-interview-questions-on-singleton.html)看到更多的问题和讨论。从Java 5开始你可以使用[枚举（enum）](http://javarevisited.blogspot.com/2011/08/enum-in-java-example-tutorial.html)来实现线程安全的单例。

**4. 使用工厂模式最主要的好处是什么？你在哪里使用？**

工厂模式的最大好处是增加了创建对象时的封装层次。如果 你使用工厂来创建对象，之后你可以使用更高级和更高性能的实现来替换原始的产品实现或类，这不需要在调用层做任何修改。可以看我的文章[工厂模式](http://javarevisited.blogspot.com/2011/12/factory-design-pattern-java-example.html)得更详细的解释和和了解更多的好处。

**5. 在Java中，什么叫观察者设计模式（observer design pattern）**？

观察者模式是基于对象的状态变化和观察者的通讯，以便他们作出相应的操作。简单的例子就是一个天气系统，当天气变化时必须在展示给公众的视图中进行反映。这个视图对象是一个主体，而不同的视图是观察者。可以在[这篇文章](http://javarevisited.blogspot.sg/2011/12/observer-design-pattern-java-example.html)中看到Java观察者模式的完整例子。

**6. 举一个用Java实现的装饰模式(decorator design pattern)？它是作用于对象层次还是类层次？**

装饰模式增加强了单个对象的能力。Java IO到处都使用了装饰模式，经典的例子就是Buffered系列类如BufferedReader和BufferedWriter，它们增强了Reader和Writer对象，以实现提升性能的Buffer层次的读取和写入。可以看[这篇文章](http://javarevisited.blogspot.com/2011/11/decorator-design-pattern-java-example.html)了解更多。

**7. 什么是MVC设计模式？举一个MVC设计模式的例子？**

**8, Java中什么是表示层设计模式（FrontController design pattern）？举一个使用表示层设计模式（front controller pattern）的例子？**

**9. 什么是责任链模式（Chain of Responsibility）？**

**10. 什么是适配器模式？举用Java实现适配器模式的例子？**

这些留给你自己做练习，作为面试准备，试着去找出这些设计模式的答案。

这些是我在很多面试中都看到的设计模式问题，当然，在[google面试](http://javarevisited.blogspot.com/2012/01/google-interview-questions-answers-top.html)和各种各样的公司如Amzone、Microsoft等等还有很多重要的专业软件设计问题。如果你遇到一些有趣的值得分享的设计问题，不妨分享出来。

原文链接： [javarevisited](http://javarevisited.blogspot.tw/2012/06/20-design-pattern-and-software-design.html) 翻译： [ImportNew.com](http://www.importnew.com/)- [陈 晓舜](http://www.importnew.com/author/chenxiaoshun)  
译文链接： <http://www.importnew.com/12649.html>