<<大话设计模式>>

图书基本信息

书名：<<大话设计模式>>

13位ISBN编号：9787302162063

10位ISBN编号：7302162069

出版时间：2007-12

出版时间：清华大学出版社

作者：程杰

页数：368

前言

本书是一本程序集？
NO。
本书是一本故事集？
NO。
本书是一本通过故事讲述程序如何设计的方法集。
本书是给连Hello World都没写过的非程序员看的书吗？
NO。
本书是给玩过穿孔纸带（0/1）、写过汇编、BASIC、C、C++、Delphi、Java、C#等语言，开发过覆盖全球、使用人数过亿、数百万行代码等大型系统的骨灰级程序员看的书吗？
NO。
本书希望能给渴望了解OO世界的初学者、困惑于僵硬、脆弱、无法复用的代码编程体验者、一直打着OO编程的旗号，做着过程式开发的基于对象的编程实践者一些好的建议和提示。
本书起因写本书源于我一次做培训的经历，学生大多是计算机专业的学生或有过一定经验的在职开发者。
他们都知道类、方法、构造方法、甚至抽象类、接口等概念，并用Visual Studio写过不少的Windows或Web程序，可是当我提问为什么要用面向对象，它的好处在哪里时，却没有人能完整地讲得出来，多数人的反应是，概念知道的，就是表达不清楚。
针对于此，我就举了中国古代的四大发明中活字印刷的例子（见第1章），通过一个虚构的三国曹操做诗的情景，把面向对象的几大好处讲解了一下，学生普遍都感觉通俗易懂，觉得这样的教学比直接告诉面向对象有什么好处要更加容易理解和记忆。
这就使得我不断地思考这样一个问题，学一门技术是否需要趣味性、通俗性的引导。
我在思考中发现，看小说时，一般情况下我都可以完整地读完它，而阅读技术方面的图书，却很少有真正的每章每页的仔细阅读。
尽管这两者是有很大区别，技术书中可能有不少知识是已经学会或暂时用不上的内容，但也不得不承认，小说之所以可以坚持读完是因为对它感兴趣，作者的文字吸引你。
而有些技术书的枯燥乏味使得阅读产生了困难，通常读个前几章就留待以后再说了。
技术课的教学同样如此，除非学生是抱着极大的学习动机来参与其中，否则照本宣科的教学、枯燥乏味的讲解，学生一定会被庞杂的概念和复杂的逻辑搅晕了头脑，致使效果大打折扣。
也正因为此，往往造成部分学生，学了四年的计算机编程，却可能连面向对象有什么好处都还说不清。
为什么不可以让技术书带点趣味性呢，哪怕这些趣味性与所讲的技术并不十分贴切，只要不是影响技术核心的本质，不产生重大的错误，让读者能轻松阅读它，并且有了一定的了解和感悟，这要比一本书写得高深无比，却被长期束之高阁要好得多。
也正是这个原因，本人开始了关于设计模式的趣味性写作的尝试。
本书读者显然本书不是给无任何编程经验的人看的，对于想入这一行的朋友来说，找一门编程语言，从头开始或许才是正道。
而本书也不太适合有了多年面向对象开发经验，对常用的设计模式了如指掌的人看的。
毕竟这里更多的是一些基础性的东西。
我时常拿程序员的成长与足球运动员的成长做对比。
GoF的《设计模式》好比是世界顶级足球射门集锦，《重构》、《敏捷软件开发》、《设计模式解析》好比是一场场最精彩的足球比赛。
我为之疯狂，为之着迷。
可是我并不只是想做一个球迷（软件使用者），而是更希望自己能成为一个足球运动员（软件设计编程者），能够亲自上场比赛，并且最终能成为球星（软件架构师）。
我仔细地阅读这些被誉为经典的著作，认真地实践其中代码，但是我总是半途而废、坚持不下去，我

痛恨自己意志力的薄弱、憎恶自己无端地放弃，难道我真的就是那么的笨？
痛定思痛，反思悔过。
我终于发现，贝利、马拉多纳不管老、胖是用来敬仰的，贝克汉姆、罗纳尔迪尼奥不管美、丑是用来欣赏的，但他们的球技……嗨，客气地说，是不容易学会的，客观地说，是不可能学得会的。
为什么会这样？
原来，我学习中缺了一个很重要的环节，我们在看到了精彩的球赛，欣赏球星高超球技的同时，却忽略了球星的成长过程。
他们尽管有一定天分，但却也是从最底层通过努力一点一点慢慢显露出来的，我们需要的不仅仅是世界杯上的那定乾坤的一脚，更需要这一脚之前是如何练出那种神奇的方法，对于程序员来讲，精彩的代码是如何想出来的，要比看到精彩的代码更加令人期待。
本书显然不是培养足球明星（软件架构师）的俱乐部，而是训练足球基本功的学校，培训的是初学足球的小球员（面向对象的程序员），本书希望的是读者阅读后可以打好面向对象的基础，从而更加容易并深入的去理解和感受GoF的《设计模式》以及其他大师作品的魅力。
本书定位本书是在学习众多大师智慧结晶的图书作品、分享了网上多位朋友的实践经验的基础上，加之自己的编程感受写出来的。
正如牛顿有句名言：“如果说我比别人看得更远些，那是因为我站在了巨人的肩上。
”但显然，本书并没有创造或发现什么模式，因此谈不上站在巨人肩膀上看得更远。
所以作者更希望本书能成为一些准备攀登面向对象编程高峰的朋友的登山引路人、提携者，在您登山途中迷路时给予指引，在您峭壁攀岩摔跤时给予保护。
本书特色本书有两个特色，第一特色是重视过程。
看了太多的计算机编程类的图书，大多数书籍都是集中在讲授优秀的解决方案或者一个完美的程序样例，但对这些解决方案和程序的演变过程却重视不够，好书之所以好，就是因为作者可以站在学习者的角度去讲解问题所在，让学习门槛降低。
《重构与模式》中有一句经典之语：“如果想成为一名更优秀的软件设计师，了解优秀软件设计的演变过程比学习优秀设计本身更有价值，因为设计的演变过程中蕴藏着大智慧。
”本人就希望能通过小菜与大鸟的对话，在不断地提问与回答过程中，在程序的不断重构演变中，把设计模式的学习门槛降低，让初学者可以更加容易地理解，为什么这样设计才是好，是如何想到这样设计的。
本书的第二个特色就是贴近生活。
尽管编程是严谨的，不容大话和戏说。
但生活却是多姿多彩的，而设计模式也不是完全孤立于现实世界而凭空想出来的理论。
事实上所有的模式都可以在生活中找到对应。
因此，通过主人公小菜和大鸟的对话，将求职、面试、工作、交友、投资、兼职、办公室文化、生活百味等等非常接近程序员生活原貌的场景写到了书中，用一个个小故事来引出模式，会让读者相对轻松地进入学习设计模式的状态。
当然，此举的最大目的还是为了深入浅出，而非纯粹噱头。
本书内容本书通篇都是以情景对话的形式，用一个又一个的小故事或编程示例来组织的。
共分为四个部分。
第一部分是面向对象的意义和好处以及几个重要的设计原则，通过小菜面试的失败引出；第二部分是详细讲解23个设计模式；第三部分是对设计模式的总结，利用小菜梦到的超级模式大赛的场景，把所有的面向对象和模式概念都拟人化来趣味性的总结设计模式之间的异同和关键点。
第四部分是附录，主要是针对对面向对象不熟悉读者的一个补充，通过一个例子的演变介绍了类、封装、继承、多态、接口、事件等概念。
本书人物及背景小菜：原名蔡遥，22岁，上海人，上海某大学计算机专业大学四年级学生，成绩一般，考研刚结束，即将毕业，正求职找工作。
大鸟：原名李大辽，29岁，小菜的表哥，云南昆明人，毕业后长期从事软件开发和管理工作，近期到上海发展，借住小菜家在宝山的空套房内。

小菜以向大鸟学习为由，也从市区父母家搬到宝山与大鸟同住。
本书研读方法本书建议按顺序阅读，如果您感觉由于面向对象知识的匮乏，例如对继承、多态、接口、抽象类的理解不足，造成阅读上的困难，不妨先阅读附录一的“培训实习生——面向对象基础”部分，然后再从第1章开始阅读。
如果您已经对不少设计模式熟悉，也不妨挑选不熟悉的模式章节阅读。
尽管本书中的代码都提供下载，但不经过读者的自己手动输入过程，其实阅读的效果是大打折扣的。强烈建议读者根据样例自己写程序，只有在运行出错，达不到预期效果时再查看本书提供的源程序，这样或许才是最好的学习方法。
有问题可及时与我联系。
我的电子邮箱是chengjielong@163.com，博客是 http://cj723.cnblogs.com/。
本书中的很多精华都来自许多大师作品，建议读者通过笔记形式记录，这将有助于您的记忆和理解设计模式，增强最终的读书效果。
本书中出现的“[ ]”是表示句子摘自某书。
例如，“策略模式（Strategy）：它定义了算法家族，分别封装起来，让它们之间可以互相替换，此模式让算法的变化不会影响到使用算法的客户[DP]。
”其中“[DP]”表示此名摘自《设计模式：可复用面向对象软件的基础》，详细摘要说明请参看附录二。
本书中29章中的虚拟人物姓名都是软件编程中的专业术语，因此凡是专业术语被指向人物姓名的都用斜体字表示，以和实际术语区分。
例如，“第一位是我们OOTV创始人，面向对象先生”，这里的斜体字面向对象指人名。
关于本书学习的疑问解答看本书需要什么基础？
主要是C#或其他编程语言的基础知识，如变量、分支判断、循环、函数等编程基础，关于面向对象基础可参看本书的附录一。
设计模式是否有必要全部学一遍？
答案是，Yes！
别被那些说什么设计模式大多用不上，根本不用全学的舆论所左右。
尽管现在设计模式远远不止23种，对所有都有研究是不太容易的，但就像作者本人一样，在学习GoF总结的23个设计模式过程中，你会被那些编程大师们进行伟大的技术思想洗礼，不断增加自己对面向对象的深入理解，从而更好的把这种思想发扬光大。
这就如同高中时学立体几何感觉没用，但当你装修好房子购买家俱时才知道，有空间感，懂得空间计算是如何的重要，你完全可能遇到买了一个大号的冰箱却放不进厨房，或买了开关门的衣橱（移门不占空间）却因床在旁边堵住了门而打不开的尴尬。
重要的不是你将来会不会用到这些模式，而是通过这些模式让你找到“封装变化”、“对象间松散耦合”、“针对接口编程”的感觉，从而设计出易维护、易扩展、易复用、灵活性好的程序。
成为诗人后可能不需要刻意地按照某种模式去创作，但成为诗人前他们一定是认真地研究过成百上千的唐诗宋词、古今名句。
如果说，数学是思维的体操，那设计模式，就是面向对象编程思维的体操。
我学了设计模式后时常会过度设计，如何办？
作者建议，暂时现象，继续努力。
设计模式有四境界：1．没学前是一点不懂，根本想不到用设计模式，设计的代码很糟糕；2．学了几个模式后，很开心，于是到处想着要用自己学过的模式，于是时常造成误用模式而不 自知；3．学完全部模式时，感觉诸多模式极其相似，无法分清模式之间的差异，有困惑，但深知误用之害，应用之时有所犹豫；4．灵活应用模式，甚至不应用具体的某种模式也能设计出非常优秀的代码，以达到无剑胜有剑的境界。
从作者本人的观点来说，不会用设计模式的人要远远超过过度使用设计模式的人，从这个角度讲，因为怕过度设计而不用设计模式显然是因噎废食。
当你认识到自己有过度使用模式的时候，那就证明你已意识到问题的存在，只有通过不断的钻研和努

力，你才能突破“不识庐山真面目，只缘身在此山中”的瓶颈，达到“会当凌绝顶，一览众山小”的境界。
编程语言的差异本书讲的是面向对象设计模式，是用.NET中的C#语言编写，但本书并不是主要讲解C#语言或.NET框架的图书，因此本书同样适合Java、VB.NET、C++等其他一些面向对象语言的读者阅读来学习设计模式。
就Java而言，主要差异来自C#对于子类继承父类或实现接口用的都是“：”，而Java中两者是有区别的。
当Cat继承抽象类Animal时，Java语法是public class Cat extends Animal当Superman实现接口IFly时，Java语法是public class Superman implements IFly然后Java中所有的方法都是虚拟的，因此不用指定new或是override修饰符。
还有一些其他差异，但基本都不影响本书的阅读。
对于VB.NET的程序员，如果阅读困难，不妨去网上查找关于转换C#与VB.Net语言的工具，比如http://www.kamalpatel.net/ConvertCSharp2VB.aspx，将下载本书的源代码转换后再进行阅读。
C++的程序员，可能在语言上会有些差异，不过本书应该不会因为语言造成对面向对象思想的误读。
不是一个人在战斗首先要感谢我的妻子李秀芳对我写作本书期间的全力支持，没有她的理解和鼓励，就不可能有本书的出版。
而我们的宝宝也将在2008年初出生，希望等宝宝懂事后能知道，在宝宝的母亲怀胎过程中，宝宝的父亲也在为书的诞生而努力。
也希望本书成为赠送给他或者她的最好的礼物。
父母的养育才有作者本人的今天，本书的出版，寻根溯源，也是父母用心教育的结果。
养育之恩，没齿难忘。
本书起源于本人在“博客园”网站的博客http://cj723.cnblogs.com/中的一个连载文章《小菜编程成长记》。
没想到连载引起了不小的反应，网友们普遍认为本人的这种技术写作方式新颖、有趣、喜欢看。
正是因为众多网友的支持，本人有了要把GoF的23种设计模式全部成文的冲动。
非常感谢这些在博客回复中鼓励我的朋友。
这里需要特别提及洪立人先生，他是本人在写书期间共同为理想奋斗的战友，写作也得到了他的大力支持和帮助，我写作的不少妙句也来自我们俩共同合作的网站http://www.miaoju.net。
在此对两位表示衷心的感谢。
写作过程中，本人参考了许多国内外大师的设计模式的著作。
尤其是《设计模式》（作者：简称GoF的Erich Gamm，Richard Helm，Ralph Johnson，John Vlissides）、《设计模式解析》（作者：Alan Shalloway，James R. Trott）、《敏捷软件开发：原则、模式与实践》（作者：Robert C.Martin）、《重构——改善既有代码的设计》（作者：Martin Fowler）、《重构与模式》（作者：Joshua Kerievsky）、《Java与模式》（作者：阎宏等等，没有他们的贡献，就没有本书的出版。
也希望本书能成为更好阅读他们这些大师作品的前期读物。
写作过程中，本人还参考了http://www.dofactory.com/ 关于23个设计模式的讲解，并引用了他们的结构图和基本代码。
在博客园中的许多朋友，比如张逸、吕震宇、李会军、idior、Allen Lee的博文，MSDN SmartCast中李建忠的讲座，CSDN博客中的大卫、ai92的博文，网站J道www.jdon.com 的版主banq的文章都给本人的写作提供了非常大的指引和帮助，在此表示感谢。
另外博客园的双鱼座先生还对本人的部分代码提出了整改意见，也表示衷心的谢意。
详细参考资料与网站链接，见附录二。
事实上，由于本人长期有看书记读书笔记的习惯，所以书中引用笔记的内容，也极有可能是来自某本书或者某个朋友的博客、某个网站的文章。
而本人已经无法一一说出其引用的地址，但这些作者的智慧同样对本书的写作带来了帮助，在此只能说声谢谢。

最后，对本书的责任编辑陈冰先生及清华大学出版社的相关工作人员，表示由衷的感谢。
本书的出版离不开陈先生的指导和其他工作人员的辛勤工作。
程 杰2007年7月序这本书最初起源于作者程杰在其博客中所写的连载文章——《小菜编程成长记》。
随着文章的一篇篇发布，这些文章新颖的表现形式和独特的风格受到了众多读者的关注和喜爱，很多人在博客中留下了评语。
有些虽然只有短短的一句话，但也可以看出是对作者由衷的感谢。
作为本书的策划编辑，最初我也是在博客园中浏览博文时阅读到这些文章的，我的直觉和网友们热情洋溢的评语告诉我，这些文章有作为一部书出版的价值，于是我就联系了程杰。
几个月后，我拿到了这部书的初稿。
初审后，我发现书稿中存在一些问题。
比如，当时书稿中还没有对UML类图进行讲解的内容，这会导致初学者学习后面的内容时感到理解困难，于是我请作者在第1 章中增加了UML类图这一节，这是简洁却精彩的一节；另外，当时作者为了便于表达某些举例的含义，有相当数量的代码都是用中文编写的，虽然中文代码看似易懂，但却会令绝大多数早已熟悉了英文代码的程序员们感到困惑和难以阅读，所以我请作者把代码改回为程序员们所熟悉的英文代码，并同时添加了更详细的中文注释。
经过几番认真和辛苦的修改与调整，现在，这本书在你的手中了。
对于这本书，我想说的是，其中的很多篇章非常的精彩，会令你禁不住叫好，但也有一些篇章会显得有些拖沓，或者是有些牵强，然而，随着你读过那些精彩的段落，读过那些不那么精彩的段落，最终，你会读到书的最后一页（很多书不能使你做到这一点），当你读完全书时，你会发现，你的心情很愉快，很平静，即使是那些当时看起来不那么精彩的段落，现在也都成为了这温馨故事的一部分。
你会记得书中那个好学、天真、而又执著的小菜，也会记得那个善于启发，经验老道的大鸟。
下面这些是来自作者博客的网友评论，看完这些热情洋溢的评论，就和作者一起，进入设计模式的大话境界吧。
本书策划编辑 陈冰2007年10月18日网友评论daigua：看到这篇精彩的成长记，我连饭都不想吃了，什么事都不想做，就想把它看完。
写得太好了！
是啊，现在很多教材都太枯燥了，不好理解。
其实书的意义就在于让人学到知识，而不在于用什么方式，为什么一定要那么教条呢，只要能让人比较容易地学到书里的知识就是一本好书。
谢谢你啊，给了我很大的信心。
我现在很有信心把编程进行到底，哈哈。
光头小松鼠：绝对经典！
一篇小故事，把程序的灵活性、可扩展性、可维护、可复用等说得怎一个妙字了得！
沉默天蝎：感激，让我这个菜鸟顿悟。
这样的写法太好了，如果老大你出书，我肯定购买！
碳碳：这种学习的方式真的很神奇，尽管每个人都能想到，但不是每个人都能做到。
或许可以把系列文章归档出书，说不定会收到追捧，呵呵。
Bryant：真的是太棒了！
我原来看过一些有关设计模式的书，都觉得太抽象，根本就不能理解，也不知道啥时候能用上。
看过你写的这些文章，才知道了应该怎样在实际中运用这些模式，而且文笔非常的幽默，享受！
Thx ^_^ 支持！
有个建议，最好慢慢地把所有的设计模式都聊聊！
Bryant：不错，楼主说的非常幽默，通俗，把我们一步一步带入面向对象的世界 thx ^_^Bryant：太棒了，我正是这样初学设计模式的小菜，需要这样的文章，谢谢楼主！
菜鸟飞：楼主，加油，支持你。
在这里献上崇高的敬意，不管你有没有感受到我挚热的目光。
请你相信，有这样一些人一直在默默地关注着你，期待着你。

wdx2008：非常好！
！
！
幽默，搞笑，易懂，真神人也，鬼神不可测！
支持楼主！
！
空明流转：呵呵，楼主说得蛮好。
国外的文章好就好在有例子，“废话”多，所以比较好理解。
至于行文风格嘛，这个倒是因人而异的。
我个人就偏向于论文式的行文风格，逻辑严密，层层递进，阐述也很清晰。
就有点像有序数组，二分法就能轻松查找到自己想要的东西，但国内的那种论文式的文章，呵呵，我看是卖弄的成分居多，实作的成分偏少，所以才那么难读的吧。
Char：现在的大学就缺少这种既通俗易懂，又有内容的东西。
Apple：不错，学习了。
希望博主能再接再厉多写点，看了很多书都没有看你的文章明白得快。
SnowDoggie：呵呵，挺好的。
其实要想找个绝对没有漏洞的例子是很辛苦的，关键在于文章本身能说明问题，能体现作者的意图就足够了。
昨天和朋友一起爬山的时候还讨论了你的文章风格，其实最有用的还是你这种寓教于乐，步步深入的风格，阳春白雪的经典虽然是经典，大众却不见得喜欢。
Jerry：不错的文章，简单明了，又不乏趣味，好的文章就得顶下。
izhizhe2000：很好，整个系列写完之后可以出书了，保证受大学生的广泛欢迎！
mekong：很是欣赏这样幽默风趣又不失睿智深刻的文字。
Wuyisky：呵呵，楼主不仅程序写得好，而且还有文学天赋。
佩服！
Jack：真正的高手是用最生动的语言，最简单的例子，这才是真正的“深入浅出”。
赞！
！
！
老兄，加油，继续哟。
BoyLee：人才，爱死你了。
做了一年外包，没技术含量。
正打算从头学习这些东西，这样的方式我最喜欢了。
Leoxu：很不错，对正在找工作的我有很大的帮助。
以后会多来光顾。
Ame：写得承上启下，始终有一主干线贯穿，作者的文字功底很强啊！
Artech：我很喜欢你的写作风格！
以一种调侃的方式讲明一个深奥的问题。
我一直在尝试如何以一种让每个人都懂得的语言来向大家分享我所理解的.NET。
你给了我一个启发。
8：醍醐灌顶！
感谢，领悟了不少东西！
！
！
Yufengly：真是太容易理解了，而且看后印象深刻，继续努力！
期待下文……支持作者！
Sopper：支持，例子举得很形象，写得很棒，以后会常来关注。

d：会技术的高人有很多，但能把技术讲得如此通俗易懂的高人并不多，你是一个，谢谢～～
～white.wu：非常喜欢您这种授人以“ 渔 ”的文章。
Answer：强啊，本菜鸟受益很大，谢谢。
Hanlei：强，很受益啊，感谢楼主，写出这么好的文章来。
金色海洋（jyk）：继续呀，我们期待中……，写得很好，一看就懂。
DSharp：看博客园这么久了，终于看到一篇有中国特色的好文。

<<大话设计模式>>

内容概要

本书通篇都是以情景对话的形式，用多个小故事或编程示例来组织讲解GoF（设计模式的经典名著——Design Patterns：Elements of Reusable Object-Oriented Software，中译本名为《设计模式——可复用面向对象软件的基础》的四位作者Erich Gamma、Richard Helm、Ralph Johnson，以及John Vlissides，这四人常被称为Gang of Four，即四人组，简称GoF）总结的23个设计模式。
本书共分为29章。
其中，第1、3、4、5章着重讲解了面向对象的意义、好处以及几个重要的设计原则；第2章，以及第6到第28章详细讲解了23个设计模式；第29章是对设计模式的全面总结。
附录部分是通过一个例子的演变为初学者介绍了面向对象的基本概念。
本书的特色是通过小菜与大鸟的趣味问答，在讲解程序的不断重构和演变过程中，把设计模式的学习门槛降低，让初学者可以更加容易地理解——为什么这样设计才是好的？
是怎样想到这样设计的？
以达到不但授之以“鱼”，还授之以“渔”的目的。
引导读者体会设计演变过程中蕴藏的大智慧。
本书适合编程初学者或希望在面向对象编程上有所提高的开发人员阅读。

作者简介

程杰，高级软件工程师＆高级培训讲师。
从事软件开发一线工作近八年时间。
曾在申银万国证券公司、上海杨浦区政府、朝华集团下属网游公司、香港晨兴集团等多行业项目开发中担任主程及项目负责人，有丰富的大中型软件开发经验，以及多年的软件设计与项目管理经验。
曾任加拿大慧

<<大话设计模式>>

书籍目录

章节摘录

插图：

媒体关注与评论

前言本书是一本程序集？
NO。
本书是一本故事集？
NO。
本书是一本通过故事讲述程序如何设计的方法集。
本书是给连Hello World都没写过的非程序员看的书吗？
NO。
本书是给玩过穿孔纸带（0/1）、写过汇编、BASIC、C、C++、Delphi、Java、C#等语言，开发过覆盖全球、使用人数过亿、数百万行代码等大型系统的骨灰级程序员看的书吗？
NO。
本书希望能给渴望了解OO世界的初学者、困惑于僵硬、脆弱、无法复用的代码编程体验者、一直打着OO编程的旗号，做着过程式开发的基于对象的编程实践者一些好的建议和提示。
本书起因写本书源于我一次做培训的经历，学生大多是计算机专业的学生或有过一定经验的在职开发者。
他们都知道类、方法、构造方法、甚至抽象类、接口等概念，并用Visual Studio写过不少的Windows或Web程序，可是当我提问为什么要用面向对象，它的好处在哪里时，却没有人能完整地讲得出来，多数人的反应是，概念知道的，就是表达不清楚。
针对于此，我就举了中国古代的四大发明中活字印刷的例子（见第1章），通过一个虚构的三国曹操做诗的情景，把面向对象的几大好处讲解了一下，学生普遍都感觉通俗易懂，觉得这样的教学比直接告诉面向对象有什么好处要更加容易理解和记忆。
这就使得我不断地思考这样一个问题，学一门技术是否需要趣味性、通俗性的引导。
我在思考中发现，看小说时，一般情况下我都可以完整地读完它，而阅读技术方面的图书，却很少有真正的每章每页的仔细阅读。
尽管这两者是有很大区别，技术书中可能有不少知识是已经学会或暂时用不上的内容，但也不得不承认，小说之所以可以坚持读完是因为对它感兴趣，作者的文字吸引你。
而有些技术书的枯燥乏味使得阅读产生了困难，通常读个前几章就留待以后再说了。
技术课的教学同样如此，除非学生是抱着极大的学习动机来参与其中，否则照本宣科的教学、枯燥乏味的讲解，学生一定会被庞杂的概念和复杂的逻辑搅晕了头脑，致使效果大打折扣。
也正因为此，往往造成部分学生，学了四年的计算机编程，却可能连面向对象有什么好处都还说不清。
为什么不可以让技术书带点趣味性呢，哪怕这些趣味性与所讲的技术并不十分贴切，只要不是影响技术核心的本质，不产生重大的错误，让读者能轻松阅读它，并且有了一定的了解和感悟，这要比一本书写得高深无比，却被长期束之高阁要好得多。
也正是这个原因，本人开始了关于设计模式的趣味性写作的尝试。
本书读者显然本书不是给无任何编程经验的人看的，对于想入这一行的朋友来说，找一门编程语言，从头开始或许才是正道。
而本书也不太适合有了多年面向对象开发经验，对常用的设计模式了如指掌的人看的。
毕竟这里更多的是一些基础性的东西。
我时常拿程序员的成长与足球运动员的成长做对比。
GoF的《设计模式》好比是世界顶级足球射门集锦，《重构》、《敏捷软件开发》、《设计模式解析》好比是一场场最精彩的足球比赛。
我为之疯狂，为之着迷。
可是我并不只是想做一个球迷（软件使用者），而是更希望自己能成为一个足球运动员（软件设计编程者），能够亲自上场比赛，并且最终能成为球星（软件架构师）。
我仔细地阅读这些被誉为经典的著作，认真地实践其中代码，但是我总是半途而废、坚持不下去，我

痛恨自己意志力的薄弱、憎恶自己无端地放弃，难道我真的就是那么的笨？
痛定思痛，反思悔过。
我终于发现，贝利、马拉多纳不管老、胖是用来敬仰的，贝克汉姆、罗纳尔迪尼奥不管美、丑是用来欣赏的，但他们的球技……嗨，客气地说，是不容易学会的，客观地说，是不可能学得会的。
为什么会这样？
原来，我学习中缺了一个很重要的环节，我们在看到了精彩的球赛，欣赏球星高超球技的同时，却忽略了球星的成长过程。
他们尽管有一定天分，但却也是从最底层通过努力一点一点慢慢显露出来的，我们需要的不仅仅是世界杯上的那定乾坤的一脚，更需要这一脚之前是如何练出那种神奇的方法，对于程序员来讲，精彩的代码是如何想出来的，要比看到精彩的代码更加令人期待。
本书显然不是培养足球明星（软件架构师）的俱乐部，而是训练足球基本功的学校，培训的是初学足球的小球员（面向对象的程序员），本书希望的是读者阅读后可以打好面向对象的基础，从而更加容易并深入的去理解和感受GoF的《设计模式》以及其他大师作品的魅力。
本书定位本书是在学习众多大师智慧结晶的图书作品、分享了网上多位朋友的实践经验的基础上，加之自己的编程感受写出来的。
正如牛顿有句名言：“如果说我比别人看得更远些，那是因为我站在了巨人的肩上。
”但显然，本书并没有创造或发现什么模式，因此谈不上站在巨人肩膀上看得更远。
所以作者更希望本书能成为一些准备攀登面向对象编程高峰的朋友的登山引路人、提携者，在您登山途中迷路时给予指引，在您峭壁攀岩摔跤时给予保护。
本书特色本书有两个特色，第一特色是重视过程。
看了太多的计算机编程类的图书，大多数书籍都是集中在讲授优秀的解决方案或者一个完美的程序样例，但对这些解决方案和程序的演变过程却重视不够，好书之所以好，就是因为作者可以站在学习者的角度去讲解问题所在，让学习门槛降低。
《重构与模式》中有一句经典之语：“如果想成为一名更优秀的软件设计师，了解优秀软件设计的演变过程比学习优秀设计本身更有价值，因为设计的演变过程中蕴藏着大智慧。
”本人就希望能通过小菜与大鸟的对话，在不断地提问与回答过程中，在程序的不断重构演变中，把设计模式的学习门槛降低，让初学者可以更加容易地理解，为什么这样设计才是好，是如何想到这样设计的。
本书的第二个特色就是贴近生活。
尽管编程是严谨的，不容大话和戏说。
但生活却是多姿多彩的，而设计模式也不是完全孤立于现实世界而凭空想出来的理论。
事实上所有的模式都可以在生活中找到对应。
因此，通过主人公小菜和大鸟的对话，将求职、面试、工作、交友、投资、兼职、办公室文化、生活百味等等非常接近程序员生活原貌的场景写到了书中，用一个个小故事来引出模式，会让读者相对轻松地进入学习设计模式的状态。
当然，此举的最大目的还是为了深入浅出，而非纯粹噱头。
本书内容本书通篇都是以情景对话的形式，用一个又一个的小故事或编程示例来组织的。
共分为四个部分。
第一部分是面向对象的意义和好处以及几个重要的设计原则，通过小菜面试的失败引出；第二部分是详细讲解23个设计模式；第三部分是对设计模式的总结，利用小菜梦到的超级模式大赛的场景，把所有的面向对象和模式概念都拟人化来趣味性的总结设计模式之间的异同和关键点。
第四部分是附录，主要是针对对面向对象不熟悉读者的一个补充，通过一个例子的演变介绍了类、封装、继承、多态、接口、事件等概念。
本书人物及背景小菜：原名蔡遥，22岁，上海人，上海某大学计算机专业大学四年级学生，成绩一般，考研刚结束，即将毕业，正求职找工作。
大鸟：原名李大辽，29岁，小菜的表哥，云南昆明人，毕业后长期从事软件开发和管理工作，近期到上海发展，借住小菜家在宝山的空套房内。

小菜以向大鸟学习为由，也从市区父母家搬到宝山与大鸟同住。
本书研读方法本书建议按顺序阅读，如果您感觉由于面向对象知识的匮乏，例如对继承、多态、接口、抽象类的理解不足，造成阅读上的困难，不妨先阅读附录一的“培训实习生——面向对象基础”部分，然后再从第1章开始阅读。
如果您已经对不少设计模式熟悉，也不妨挑选不熟悉的模式章节阅读。
尽管本书中的代码都提供下载，但不经过读者的自己手动输入过程，其实阅读的效果是大打折扣的。强烈建议读者根据样例自己写程序，只有在运行出错，达不到预期效果时再查看本书提供的源程序，这样或许才是最好的学习方法。
有问题可及时与我联系。
我的电子邮箱是chengjielong@163.com，博客是http://cj723.cnblogs.com/。
本书中的很多精华都来自许多大师作品，建议读者通过笔记形式记录，这将有助于您的记忆和理解设计模式，增强最终的读书效果。
本书中出现的“[]”是表示句子摘自某书。
例如，“策略模式（Strategy）：它定义了算法家族，分别封装起来，让它们之间可以互相替换，此模式让算法的变化不会影响到使用算法的客户[DP]。
”其中“[DP]”表示此名摘自《设计模式：可复用面向对象软件的基础》，详细摘要说明请参看附录二。
本书中29章中的虚拟人物姓名都是软件编程中的专业术语，因此凡是专业术语被指向人物姓名的都用斜体字表示，以和实际术语区分。
例如，“第一位是我们OOTV创始人，面向对象先生”，这里的斜体字面向对象指人名。
关于本书学习的疑问解答看本书需要什么基础？
主要是C#或其他编程语言的基础知识，如变量、分支判断、循环、函数等编程基础，关于面向对象基础可参看本书的附录一。
设计模式是否有必要全部学一遍？
答案是，Yes！
别被那些说什么设计模式大多用不上，根本不用全学的舆论所左右。
尽管现在设计模式远远不止23种，对所有都有研究是不太容易的，但就像作者本人一样，在学习GoF总结的23个设计模式过程中，你会被那些编程大师们进行伟大的技术思想洗礼，不断增加自己对面向对象的深入理解，从而更好的把这种思想发扬光大。
这就如同高中时学立体几何感觉没用，但当你装修好房子购买家俱时才知道，有空间感，懂得空间计算是如何的重要，你完全可能遇到买了一个大号的冰箱却放不进厨房，或买了开关门的衣橱（移门不占空间）却因床在旁边堵住了门而打不开的尴尬。
重要的不是你将来会不会用到这些模式，而是通过这些模式让你找到“封装变化”、“对象间松散耦合”、“针对接口编程”的感觉，从而设计出易维护、易扩展、易复用、灵活性好的程序。
成为诗人后可能不需要刻意地按照某种模式去创作，但成为诗人前他们一定是认真地研究过成百上千的唐诗宋词、古今名句。
如果说，数学是思维的体操，那设计模式，就是面向对象编程思维的体操。
我学了设计模式后时常会过度设计，如何办？
作者建议，暂时现象，继续努力。
设计模式有四境界：1．没学前是一点不懂，根本想不到用设计模式，设计的代码很糟糕；2．学了几个模式后，很开心，于是到处想着要用自己学过的模式，于是时常造成误用模式而不　自知；3．学完全部模式时，感觉诸多模式极其相似，无法分清模式之间的差异，有困惑，但深知误用之害，应用之时有所犹豫；4．灵活应用模式，甚至不应用具体的某种模式也能设计出非常优秀的代码，以达到无剑胜有剑的境界。
从作者本人的观点来说，不会用设计模式的人要远远超过过度使用设计模式的人，从这个角度讲，因为怕过度设计而不用设计模式显然是因噎废食。
当你认识到自己有过度使用模式的时候，那就证明你已意识到问题的存在，只有通过不断的钻研和努

力，你才能突破“不识庐山真面目，只缘身在此山中”的瓶颈，达到“会当凌绝顶，一览众山小”的境界。
编程语言的差异本书讲的是面向对象设计模式，是用.NET中的C#语言编写，但本书并不是主要讲解C#语言或.NET框架的图书，因此本书同样适合Java、VB.NET、C++等其他一些面向对象语言的读者阅读来学习设计模式。
就Java而言，主要差异来自C#对于子类继承父类或实现接口用的都是“：”，而Java中两者是有区别的。
当Cat继承抽象类Animal时，Java语法是public class Cat extends Animal当Superman实现接口IFly时，Java语法是public class Superman implements IFly然后Java中所有的方法都是虚拟的，因此不用指定new或是override修饰符。
还有一些其他差异，但基本都不影响本书的阅读。
对于VB.NET的程序员，如果阅读困难，不妨去网上查找关于转换C#与VB.Net语言的工具，比如http://www.kamalpatel.net/ConvertCSharp2VB.aspx，将下载本书的源代码转换后再进行阅读。
C++的程序员，可能在语言上会有些差异，不过本书应该不会因为语言造成对面向对象思想的误读。
不是一个人在战斗首先要感谢我的妻子李秀芳对我写作本书期间的全力支持，没有她的理解和鼓励，就不可能有本书的出版。
而我们的宝宝也将在2008年初出生，希望等宝宝懂事后能知道，在宝宝的母亲怀胎过程中，宝宝的父亲也在为书的诞生而努力。
也希望本书成为赠送给他或者她的最好的礼物。
父母的养育才有作者本人的今天，本书的出版，寻根溯源，也是父母用心教育的结果。
养育之恩，没齿难忘。
本书起源于本人在“博客园”网站的博客http://cj723.cnblogs.com/中的一个连载文章《小菜编程成长记》。
没想到连载引起了不小的反应，网友们普遍认为本人的这种技术写作方式新颖、有趣、喜欢看。
正是因为众多网友的支持，本人有了要把GoF的23种设计模式全部成文的冲动。
非常感谢这些在博客回复中鼓励我的朋友。
这里需要特别提及洪立人先生，他是本人在写书期间共同为理想奋斗的战友，写作也得到了他的大力支持和帮助，我写作的不少妙句也来自我们俩共同合作的网站http://www.miaoju.net。
在此对两位表示衷心的感谢。
写作过程中，本人参考了许多国内外大师的设计模式的著作。
尤其是《设计模式》（作者：简称GoF的Erich Gamm，Richard Helm，Ralph Johnson，John Vlissides）、《设计模式解析》（作者：Alan Shalloway，James R. Trott）、《敏捷软件开发：原则、模式与实践》（作者：Robert C.Martin）、《重构——改善既有代码的设计》（作者：Martin Fowler）、《重构与模式》（作者：Joshua Kerievsky）、《Java与模式》（作者：阎宏等等，没有他们的贡献，就没有本书的出版。
也希望本书能成为更好阅读他们这些大师作品的前期读物。
写作过程中，本人还参考了http://www.dofactory.com/ 关于23个设计模式的讲解，并引用了他们的结构图和基本代码。
在博客园中的许多朋友，比如张逸、吕震宇、李会军、idior、Allen Lee的博文，MSDN SmartCast中李建忠的讲座，CSDN博客中的大卫、ai92的博文，网站J道www.jdon.com 的版主banq的文章都给本人的写作提供了非常大的指引和帮助，在此表示感谢。
另外博客园的双鱼座先生还对本人的部分代码提出了整改意见，也表示衷心的谢意。
详细参考资料与网站链接，见附录二。
事实上，由于本人长期有看书记读书笔记的习惯，所以书中引用笔记的内容，也极有可能是来自某本书或者某个朋友的博客、某个网站的文章。
而本人已经无法一一说出其引用的地址，但这些作者的智慧同样对本书的写作带来了帮助，在此只能说声谢谢。

最后，对本书的责任编辑陈冰先生及清华大学出版社的相关工作人员，表示由衷的感谢。
本书的出版离不开陈先生的指导和其他工作人员的辛勤工作。
程 杰 2007年7月序这本书最初起源于作者程杰在其博客中所写的连载文章——《小菜编程成长记》。随着文章的一篇篇发布，这些文章新颖的表现形式和独特的风格受到了众多读者的关注和喜爱，很多人在博客中留下了评语。
有些虽然只有短短的一句话，但也可以看出是对作者由衷的感谢。
作为本书的策划编辑，最初我也是在博客园中浏览博文时阅读到这些文章的，我的直觉和网友们热情洋溢的评语告诉我，这些文章有作为一部书出版的价值，于是我就联系了程杰。
几个月后，我拿到了这部书的初稿。
初审后，我发现书稿中存在一些问题。
比如，当时书稿中还没有对UML类图进行讲解的内容，这会导致初学者学习后面的内容时感到理解困难，于是我请作者在第1章中增加了UML类图这一节，这是简洁却精彩的一节；另外，当时作者为了便于表达某些举例的含义，有相当数量的代码都是用中文编写的，虽然中文代码看似易懂，但却会令绝大多数早已熟悉了英文代码的程序员们感到困惑和难以阅读，所以我请作者把代码改回为程序员们所熟悉的英文代码，并同时添加了更详细的中文注释。
经过几番认真和辛苦的修改与调整，现在，这本书在你的手中了。
对于这本书，我想说的是，其中的很多篇章非常的精彩，会令你禁不住叫好，但也有一些篇章会显得有些拖沓，或者是有些牵强，然而，随着你读过那些精彩的段落，读过那些不那么精彩的段落，最终，你会读到书的最后一页（很多书不能使你做到这一点），当你读完全书时，你会发现，你的心情很愉快，很平静，即使是那些当时看起来不那么精彩的段落，现在也都成为了这温馨故事的一部分。
你会记得书中那个好学、天真、而又执著的小菜，也会记得那个善于启发，经验老道的大鸟。
下面这些是来自作者博客的网友评论，看完这些热情洋溢的评论，就和作者一起，进入设计模式的大话境界吧。
本书策划编辑 陈冰2007年10月18日网友评论daigua：看到这篇精彩的成长记，我连饭都不想吃了，什么事都不想做，就想把它看完。
写得太好了！
是啊，现在很多教材都太枯燥了，不好理解。
其实书的意义就在于让人学到知识，而不在于用什么方式，为什么一定要那么教条呢，只要能让人比较容易地学到书里的知识就是一本好书。
谢谢你啊，给了我很大的信心。
我现在很有信心把编程进行到底，哈哈。
光头小松鼠：绝对经典！
一篇小故事，把程序的灵活性、可扩展性、可维护、可复用等说得怎一个妙字了得！
沉默天蝎：感激，让我这个菜鸟顿悟。
这样的写法太好了，如果老大你出书，我肯定购买！
碳碳：这种学习的方式真的很神奇，尽管每个人都能想到，但不是每个人都能做到。
或许可以把系列文章归档出书，说不定会收到追捧，呵呵。
Bryant：真的是太棒了！
我原来看过一些有关设计模式的书，都觉得太抽象，根本就不能理解，也不知道啥时候能用上。
看过你写的这些文章，才知道了应该怎样在实际中运用这些模式，而且文笔非常的幽默，享受！
Thx ^_^ 支持！
有个建议，最好慢慢地把所有的设计模式都聊聊！
Bryant：不错，楼主说的非常幽默，通俗，把我们一步一步带入面向对象的世界 thx ^_^Bryant：太棒了，我正是这样初学设计模式的小菜，需要这样的文章，谢谢楼主！
菜鸟飞：楼主，加油，支持你。
在这里献上崇高的敬意，不管你有没有感受到我挚热的目光。
请你相信，有这样一些人一直在默默地关注着你，期待着你。

wdx2008：非常好！
！
！
幽默，搞笑，易懂，真神人也，鬼神不可测！
支持楼主！
！
空明流转：呵呵，楼主说得蛮好。
国外的文章好就好在有例子，“废话”多，所以比较好理解。
至于行文风格嘛，这个倒是因人而异的。
我个人就偏向于论文式的行文风格，逻辑严密，层层递进，阐述也很清晰。
就有点像有序数组，二分法就能轻松查找到自己想要的东西，但国内的那种论文式的文章，呵呵，我看是卖弄的成分居多，实作的成分偏少，所以才那么难读的吧。
Char：现在的大学就缺少这种既通俗易懂，又有内容的东西。
Apple：不错，学习了。
希望博主能再接再厉多写点，看了很多书都没有看你的文章明白得快。
SnowDoggie：呵呵，挺好的。
其实要想找个绝对没有漏洞的例子是很辛苦的，关键在于文章本身能说明问题，能体现作者的意图就足够了。
昨天和朋友一起爬山的时候还讨论了你的文章风格，其实最有用的还是你这种寓教于乐，步步深入的风格，阳春白雪的经典虽然是经典，大众却不见得喜欢。
Jerry：不错的文章，简单明了，又不乏趣味，好的文章就得顶下。
izhizhe2000：很好，整个系列写完之后可以出书了，保证受大学生的广泛欢迎！
mekong：很是欣赏这样幽默风趣又不失睿智深刻的文字。
Wuyisky：呵呵，楼主不仅程序写得好，而且还有文学天赋。
佩服！
Jack：真正的高手是用最生动的语言，最简单的例子，这才是真正的“深入浅出”。
赞！
！
！
老兄，加油，继续哟。
BoyLee：人才，爱死你了。
做了一年外包，没技术含量。
正打算从头学习这些东西，这样的方式我最喜欢了。
Leoxu：很不错，对正在找工作的我有很大的帮助。
以后会多来光顾。
Ame：写得承上启下，始终有一主干线贯穿，作者的文字功底很强啊！
Artech：我很喜欢你的写作风格！
以一种调侃的方式讲明一个深奥的问题。
我一直在尝试如何以一种让每个人都懂得的语言来向大家分享我所理解的.NET。
你给了我一个启发。
8：醍醐灌顶！
感谢，领悟了不少东西！
！
！
Yufengly：真是太容易理解了，而且看后印象深刻，继续努力！
期待下文……支持作者！
Sopper：支持，例子举得很形象，写得很棒，以后会常来关注。

d：会技术的高人有很多，但能把技术讲得如此通俗易懂的高人并不多，你是一个，谢谢～～
～white.wu：非常喜欢您这种授人以“ 渔 ”的文章。
Answer：强啊，本菜鸟受益很大，谢谢。
Hanlei：强，很受益啊，感谢楼主，写出这么好的文章来。
金色海洋（jyk）：继续呀，我们期待中……，写得很好，一看就懂。
DSharp：看博客园这么久了，终于看到一篇有中国特色的好文。

<<大话设计模式>>

编辑推荐

《大话设计模式》是准备攀登面向对象编程高峰朋友们的引路人和提携者；《大话设计模式》是学习、体会和领悟了众多大师智慧结晶后的图书作品；《大话设计模式》是你深入理解和感受GoF的《设计模式》及其它大师作品的必备书籍；《大话设计模式》授之以“鱼”，更授之以“渔”。
感受设计演变过程中所蕴含的大智慧，体会乐与怒的程序人生中值得回味的一幕幕。
设计模式的趣味解读，面向对象的深入剖析。
在诙谐与温馨中做一次面向对象编程思维的体操。

版权说明