《设计模式》

图书基本信息

书名：《设计模式》

13位ISBN编号：9787111095071

10位ISBN编号：7111095073

出版时间：2002-3-1

出版社：机械工业出版社

作者：伽玛

页数：424

《设计模式》

内容概要

本书内容全面，讲解鞭辟入里，独具特色，读者必将在计算机科学的宫殿中的由登堂而入室。

《设计模式》

作者简介

作者：(美国)伽玛 等

《设计模式》

书籍目录

《设计模式》

编辑推荐

《设计模式:可复用面向对象软件的基础》(英文版)是经典原版书库中的设计模式可复用面向对象软件的基础分册。

精彩短评

《设计模式》

精彩书评

1、久仰这本书的大名,虽然以前扫过<大话设计模式>和Head First的设计模式,但老实说我真的没耐心看完这两本书.我想或许是我的功力不够,所以没耐心继续往下看吧.不过后来看到不少说道模式的牛文,里面引用了GoF的Pattern Intent,忽然我发现其实在自己的设计中已经不知不觉地用到了很多里面的思想.之前没有看这本书,老实说是因为看到不少书评说这本书的语言太"学术",不够"亲近".为此我也做好了心理准备,最后一下决心,还是买了这本书.先是顺序地看完了Chapter I和II,然后按照II说到得顺序看了后面模式的具体介绍,也忍不住地看了那些介绍的Related Pattern. 实话说,我不觉得这本书的语言有什么太过学术的地方,相反,我觉得倒是很精确和简洁,很值得品味.这个或许和个人的习惯有关系,一直以来都认为所谓高手,一定能把自己熟悉的领域用简洁扼要的语言表达出来,而不乏可读性.这个地方的可读性不能以大众来定,所以我仍然承认这本书是不适合初学者看的,换句话说,值得大多数人细细品味,读上多遍.对于设计模式而言,每次看的感觉和悟出都不太一样,而这次读这本书,是有一种返璞归真的感觉,这也正说明了这本书的经典.要说感悟的话,这里或许说不出那么多.不过个人认为需要澄清的是:1.这本书的语言并不学术化,相反是简洁扼要2.语言难度而言,印象中只出现了个位数的GRE级别的词汇3.这本书抽象,也不抽象,设计模式很大的程度上说明了如何抽象,说以不可能不抽象,而里面是不乏例子的4.里面有C++的Sample Code,如果觉得看不懂,我认为是C++水平不够,里面只有一处稍微有那么一点点tricky,P240的typedef void (Receiver::* Action();其他地方(在语言方面)都非常非常基础.因此,还是强烈推荐.相信大家读这本书一定会很开心的.

2、由李英军同志翻译的中文版我是无论如何读不懂。本来这本书GOF就是以学术专著的形式写成，所以晦涩啊，艰深啊，再加上中文翻译的不准确性，就造就了该书中文版被芸芸众生顶礼膜拜为天书的壮观景象。不过还好，机械工业出版社自己都觉得问心有愧，遂直接出版『影印版』，于是真迹终于面世。不过话说回来，该书不适合初学者，功力未到，勿念此书。推荐《Design Patterns Explained 2nd》《Head First Design Pattern》作为入门。

3、如书中所说“设计面向对象软件比较困难，而设计可复用的面向对象软件就更加困难”，很多人在设计面向对象软件时往往无从下手，勉强设计的结果也很难扩展和维护，这本书告诉你，大师是怎么设计的。本书以一个文档编辑器的示例开始引入常见的设计模式，并做了简单的说明。后面各章按类别：创建型、结构型、行为，详细分析了各个模式，并配以样例代码。本书有一点比较好的是，他把几个有关的模式放在一起，对比他们的特点并说明他们之间的关系，这样一来，即使你对某个模式只有一点模糊的概念，完全可以把这本书当字典，查到你想使用的模式和相关的模式。当然有一点觉得不大友好，为啥样例代码是 C++ 和 smalltalk？C++ 是一个主流的编程语言，用这个说的过去，虽然我对 C++ 一点都不熟悉，smalltalk 虽然有点名气，但是毕竟用的不广泛，估计没多少人能看懂。我还是建议使用 Java，一门语言就足够了。

4、一遍一遍的读，闲来没事就翻一翻，已经快沦为厕所读物了！：）现在我的感受是——举头四顾，不见模式！其实模块化，封装，信息隐蔽，高内聚低耦合，这些才是程序设计的精髓。不管是设计模式，还是KISS，归根到底都是为了实现上述目标，所以不能为了设计模式而使用设计模式。

5、可复用的软件，好！面向对象的软件，好！我的代码应该尽可能的用设计模式，嗯，等等。之所以出此感慨是因为最近一次的code review我用了state pattern，结果reviewer说不直观。我郁闷了一会儿，试图教育他什么是state pattern，为什么好。不过最后还是改回了if...else来实现。又看了公司（Google）的一个Web Application Framework，文档里说这是用了Command Pattern。我一看，好，用了模式的肯定是好东东。现在我的感觉是如果是写Library、Framework什么的，应该用设计模式。写Application，理解设计模式可以帮助学习Framework。不过Application本身的代码未必要用设计模式，因为设计模式是适合需要重用的代码，很多Application的代码并不好复用。

6、因为可以买到这本书的地方和版本这么多。所以这个质量纯粹是指的外在质量（内在质量不用我介绍了吧？其实我觉得设计模式读一两本书就够了，最重要的还是应用吧）。不知道为什么，电力出版社出版的一套开本和印刷都让人读起来挺舒服的深蓝色封面的特辑（原版风暴系列）里没有它(可是有design pattern explained)。机械工业出版社的这个系列都是小开本，翻起来费劲多了。但是这恐怕也是世界上最便宜的Design Patterns了！

7、两年前的面向对象课程,要求我们读这本书,当时没太看懂,只是简单的应付了一下作业,将自己要讲的3个模式囫囵吞枣地看了.当时讲composite模式的时候，还记得被黄震春老师批评，我正好理解反了

。。今年暑假我再次看此书，多了两年的编程经历，一年的实习经历，忽然觉得简单了很多。在邮箱找到了两年前我当时在台上讲composite模式的ppt，看了几页就立马发现，我确实搞反了。这本20年前的神作，虽然很多人说晦涩难懂，我现在觉得其实还好，没想像中的那么难理解，也没想像中的那么枯燥无味，认真看还是能看的。书中的各种模式可能还需要自己在接下来的编程中逐渐去融会贯通。但是这几天读完了全书，一本两年前看不懂，一直令我感到畏惧的书，今天终于被我拿下了，还是难以隐藏心中的喜悦之情，所以短短写一两百字纪念一下~
8、这本书的中文版2000上年大学时就看过，后来机工出了影印版就买了一本。说实话这不是一本看完一遍就可以扔掉的书，它需要你不断实践，每次翻一下都有新的收获。
9、因为看书之前我已经有了不少经验，所以只是草草翻了一遍，基本上自习看的只有自己没有实现过的FlyWeight，所以对书里面的内容也没有太多的评论,不过书里的代码清晰，基本上看每个设计模式的名称，再看看代码就能明白模式是如何实现的。书本身不错的，但是光看书不行，为模式而模式更不行。很多模式都有其特定的应用背景和面对的实际问题。在看书之前很多模式我自己就实现过。为模式而模式其实挺傻的。尤其是看了书之后就特别想把这些实际运用，往往导致“过度设计”。还是建议自己实际做一些比较困难的项目之后，再回过头来看这本书，如果你在项目中已经自己重新发现了书里的模式，你会有“果然如此”的感觉，而如果没有，那就有“原来如此”的收获。
10、这应该算是学术著作——从最后一章写道的编著历史看，也确实如此（最先起源于Erich的博士论文）。恐怕大部分读者都会对这样的书籍感到枯燥。不过倒挺适合我的口味:-D。由于写作年代久远等缘故，书中的23种设计模式都是“古典”设计模式了。从现在的角度看，这些都只是设计模式中的冰山一角。书中所举的应用例子也都是很传统的软件应用。不过读下来，感觉多态、复用的思想贯穿全书，基本上每种设计模式都对应着一种要复用的东西（可能需要第二次阅读以确认），这也符合本书的副标题。这个思想应该对当今大多数设计模式仍然适用。

《设计模式》

章节试读

1、《设计模式》的笔记-第77页

刚读过第2章。这一章以设计一个所见即所得的文档编辑器为例，将第3章起的很多设计模式的应用场景都串了起来。不过还是有几处让我困惑的地方：

1. Composite Pattern完全就是在实现一个树结构。只要是需要用树来作为数据结构的地方，肯定都会采用这种设计模式。干脆直接叫Tree Pattern好了。我都读过Composite Pattern那一章了，还不太清楚这跟实现一颗普通的树有啥区别。可能这个模式强调的是，树中的所有节点对外都是一个统一的接口，每个节点内部的实现可以各自不同。

2. Strategy Pattern和Bridge Pattern在这一章里给我感觉十分相似。2.9节小结里写道，Strategy的作用是“ allow different formatting algorithms ”，Bridge的作用是“ allow multiple windowing platforms ”。其实都可归结为“ allow different XXX ”。不同的视窗平台蕴含着的其实就是不同的绘制实现，2.6节的代码例子也说明了这点。实际上2.3节的Strategy UML图和2.6节的Bridge UML图也非常相似。不过在后面章节里，这两个模式却很不一样，而且Strategy被归到了行为模式，Bridge被归到了结构模式。希望读到后面时能找到答案。

2、《设计模式》的笔记-第151页

151-161 Bridge
Decouple an abstraction from its implementation so that the two can vary independently.

I want to say I like the original English book.

3、《设计模式》的笔记-第255页

Interpreter Pattern（解释器模式）是我读本书到现在最难理解的模式。可能是因为阅读过程中我一直分不清interpret（解释）和parse（解析）的区别。Interpreter Pattern是不管parse的。Interpret就是evaluate，这是求值的过程；Parse则是构建可用于求值的语法树的过程；Parse是interpret的先行步骤。

以书中的Sample Code为例：

1）给定一个布尔表达式（书中的C++示例），我要求它的最终值是真还是假——这是interpret；但是给定一行文本，我要读懂这行文本是一个什么样的布尔表达式——这是parse；只有我读懂了这个布尔表达式，我才能对它求值。

2）给定一个正则表达式（书中的Smalltalk示例），再给定一行文本，我要知道这行文本是否匹配这个正则表达式，或者再给定一个文本文件，我要找出文件中匹配该正则表达式的文本——这是interpret；但是给定一行文本，我要读懂这行文本描述的是怎样一个正则表达式——这是parse；只有我读懂了这个正则表达式，我才能用它去匹配任意文本。

所以parse是“ 读懂 ”，interpret是用读懂的东西去求值。可能有的时候，这两者的区别并不是那么清晰。

另外，Interpret Pattern和Composite Pattern的确很相似，它们之间有很紧密的联系。两者的类图甚至是一模一样的，只是类的名字有所不同。这更加印证了我读本书过程中一个越来越强的体会：设计模式

的目的十分重要；理解一个设计模式时，不要光看它的类图，仅仅知道模式中有哪些类哪些方法是不够的。

4、《设计模式》的笔记-第95页

之前读第2章A Case Study的时候就已经基本理解了抽象工厂模式的思想。这一章对我而言比较新的内容倒是Smalltalk利用自身的类为一等公民和动态类型的特性，使用原型（Prototype）的思路来实现抽象工厂的方法。

Known Uses这一节提到，ET++使用了抽象工厂模式来实现不同窗口系统（如X Windows和SunView）之间的可移植性。不过在第2章2.6节里，作者却是使用桥接模式（Bridge Pattern）来实现同样目的的，理由是不同窗口系统之间的编程接口往往难以兼容。

5、《设计模式》的笔记-第126页

我把这一章读了两遍才感觉理解了原型模式的含义，可能是因为实践中没怎么碰到过这种模式。不过Consequences一节写道的原型模式的其中两点好处还是挺让我糊涂：Specifying new objects by varying values和Specifying new objects by varying structure。要是能就这两点举个代码上的例子就好了。可惜篇幅有限。

6、《设计模式》的笔记-第327页

Template methods lead to an inverted control structure that's sometimes referred to as "the Hollywood principle," that is, "Don't call us, we'll call you" [Swe85]. This refers to how a parent class calls the operations of a subclass and not the other way around.
这里提到了 IoC 概念，可以看出这些概念在成书时还不成熟。它和 DI 有区别，意义也较模糊，看起来跟多态一样。

《设计模式》

版权说明