重点大学软件工程规划系列教材

UML 2面向对象分析与设计

谭火彬　编著

清华大学出版社

重点大学软件工程规划系列教材

UML 2面向对象分析与设计

谭火彬　编著

清华大学出版社
北京

内 容 简 介

分析和设计是软件开发中至关重要的一环，面向对象的方法是主流的软件开发方法，UML 是用于面向对象分析设计的标准化建模语言。本书围绕这三个方面展开，以论述分析设计建模过程为最终目标，以面向对象方法作为建模的理论基础，以 UML 作为建模支撑语言。全书从面向对象和 UML 的基本概念入手，循序渐进地讲解业务建模、需求建模、需求分析、设计原则和模式、架构设计、构件设计和代码生成等分析设计期间的各个知识点，并通过多个贯穿全书的案例将各个知识点串联起来，形成一套完整的面向对象分析设计方法论。

本书是作者多年从事软件工程教学和软件项目开发实践的总结，书中并没有太多抽象的概念，主要关注实际软件开发中所需要的知识和实践技能，力求做到通俗易懂。

本书可作为高等院校软件工程专业及计算机相关专业的高年级本科生或研究生教材，也可供软件开发人员阅读和参考。

图书在版编目(CIP)数据

UML 2 面向对象分析与设计/谭火彬编著. --北京：清华大学出版社，2013.5
重点大学软件工程规划系列教材
ISBN 978-7-302-30788-4

Ⅰ. ①U…　Ⅱ. ①谭…　Ⅲ. ①面向对象语言－程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 287041 号

责任编辑：魏江江　王冰飞
封面设计：傅瑞学
责任校对：时翠兰
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn,010-62795954

印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.25　　**字　　数**：518 千字
版　　次：2013 年 5 月第 1 版　　**印　　次**：2013 年 5 月第1 次印刷
印　　数：1～3000
定　　价：35.00 元

产品编号：050122-01

FOREWORD

前　言

毋庸置疑，面向对象的方法已成为现代软件开发中最主流的方法，即使是最新的 SOA、云计算等概念也都是建立在面向对象方法基础之上的进一步抽象。与此同时，自 1997 年 UML 正式诞生，到 2011 年发布的 UML 2.4.1，经历了多个版本的发展和完善，UML 已成为建模语言的国际标准（ISO 19501 和 ISO 19505），基于 UML 的面向对象分析设计方法也日益成熟。然而，由于 UML 只是提供了一种标准的表示法，在分析设计过程中，什么时候以什么方式使用什么 UML 模型等具体的建模实践并没有在 UML 中定义，而这才是广大软件开发人员所要掌握的实践技能，也是本书所关注的内容。

本书目标

本书系统地介绍了利用最新的 UML 2 进行面向对象分析与设计的过程，全书的主要目标包括以下三个方面。

- OO（面向对象）：建立对象的思维方式，对面向对象思想和理论有深入的理解；
- UML（统一建模语言）：能够熟练地使用 UML 表达面向对象的设计思想；
- Model（建模）：运用面向对象的一般原则和模式进行应用系统的分析和设计建模。

组织结构

本书总体结构可以分为三大部分。第一部分为基础概念，包括第 1 章和第 2 章。其中，第 1 章为上升到面向对象，通过案例引出面向对象的方法，并重点介绍了对象技术中的几个核心概念。第 2 章为可视化建模技术基础，全面介绍了有关 UML 2 的组织结构和内容。这些基础概念将在后续的分析设计中被广泛使用。

第二部分为面向对象的分析，包括第 3、4、5 章。第 3 章为业务建模，原始业务是需求分析的出发点，本章简要地介绍了业务建模的基本概念和方法，并提供了一些实践指南。第 4 章为用例建模，系统地介绍了利用 UML 用例模型进行需求定义的过程和实践。第 5 章为

用例分析，介绍了如何围绕第 4 章所建立的用例模型进行面向对象分析的方法和实践。

第三部分为面向对象的设计，包括第 6、7、8、9、10 章。其中，第 6 和 7 章为设计基础，分别介绍了有关面向对象设计的基本原则和模式，这些原则和模式将有效地指导后续的设计过程。第 8 章为架构设计，介绍了如何在系统的全局范围内，基于分析活动的成果定义设计元素、设计机制等内容，从而构造系统的组织结构。第 9 章为构件设计，介绍了如何在系统的局部设计各个细节，包括用例设计、子系统设计、类设计和数据库设计等方面的内容。第 10 章为从模型到代码，简单地介绍了设计模型和代码之间的映射，为后续编码做准备。

在案例设计方面，本书设计了两个贯穿全书的案例：旅店预订系统和旅游业务申请系统。这两个案例各有侧重，通过它们，读者不仅可以掌握 UML 建模的基本方法，还可以全面了解在整个系统开发过程中从分析模型到设计模型不断演化的过程。此外，在各个章节中，针对一些特定的知识点，设计了各种小的案例进行阐述，这些案例包括第 1 章开篇的素数问题、第 2 章的图书馆管理系统、第 3 章的饭店系统、第 6 章的咖啡机系统、第 7 章的可复用按钮等。

有关 UML 内容，本书从两个层面进行介绍：首先在第 2 章对 UML 基本概念、组织结构和各种模型进行了系统、初步的介绍；之后则在后续的分析设计实践中针对一些重点 UML 模型的使用进行详细、深入的论述，使读者在掌握 UML 基本概念之后，能够在需要的地方进行应用，有关各章节中涉及的 UML 模型和核心概念如下表所示。

章节	章节名称	章节性质	UML 模型	核心概念
第 1 章	上升到面向对象	基础		对象技术、类、对象
第 2 章	可视化建模技术	基础	*（全部）	UML 组织结构
第 3 章	业务建模	业务	活动图	业务参与者、业务用例、活动
第 4 章	用例建模	需求	用例图	用例、参与者、用例关系
第 5 章	用例分析	分析	顺序图、类图	用例实现，分析类（边界类、控制类和实体类）
第 6 章	面向对象的设计原则	设计基础	通信图	设计原则
第 7 章	面向对象的设计模式	设计基础		模式、设计模式、GRASP
第 8 章	架构设计	设计	包图、部署图	构架、设计元素、设计机制、进程、线程
第 9 章	构件设计	设计	类图、状态机图、构件图	接口、子系统、组合结构操作、方法、状态、关系
第 10 章	从模型到代码	实现		正向工程、逆向工程

有关 UML 工具的选择，也是读者所关心的问题。市面上有很多商业的或开源的 UML 工具，这些工具各有特点，但其核心建模功能都相差不大。工具本身只是一种实现手段，选择哪款 UML 工具，并不影响对本书概念的理解和实践。本书中的 UML 模型主要采用 IBM Rational Rose 2003 和 Sparx Systems Enterprise Architect 7.5，都不算最新的工具，够用即可。选择 Rose 是因为笔者从最早学习 UML 开始就一直使用该工具，虽然有点老，但能满足大部分建模需求；不过由于 Rose 2003 不支持最新的 UML 2，所以针对 UML 2 中的新概念选择了 Enterprise Architect（没有选择 IBM Rational 的后续版本 RSA 是因为这个工具集过于庞大，更倾向于一个集成开发平台，不适合作为一个普通的 UML 工具进行介

绍；此外，其默认的图形样式颜色较淡，不适合放在书中展示）。当然，这两个工具也无法覆盖到所有的 UML 概念，因此书中有些模型是选择其他的 UML 工具或一些绘图工具完成的。

致谢

本书是笔者在北航软件学院近 10 年的研究生面向对象分析与设计课程教学基础上编写而成的，非常感谢软件学院的领导、教师和学生提供这样的交流平台，书中大部分内容和案例都是通过此课程的教学不断形成和完善的。还要特别感谢在课程教学和教材编写过程中提供帮助的姚淑珍教授、林广艳副教授、杨文龙教授、孟岩老师等学院老师和企业导师。此外在这期间，笔者还曾参加过 IBM 组织的面向对象分析与设计课程培训、UML China 公开课等，也曾为一些软件企业进行过有关方面的培训，这些来自企业的经历也充实了课程和教材内容，在此一并表示感谢。最后，还需要感谢我的家人，正是因为她们无私的奉献才使得我有更多的精力投入到课程教学和教材编写中。

编 者

2012 年 10 月

CONTENTS

目　录

CHAPTER 1

第 1 章

上升到面向对象

从早期的手工艺开发阶段，到软件工程的出现；从传统的结构化开发方法，到面向对象的方法；软件开发方法正逐渐扮演着更加重要的角色。而面向对象的方法也已取代了传统的软件工程方法，成为软件开发方法的主流。对象、类、封装、继承和多态等概念也已被广泛接受。

本章从一个简单案例入手，分析面向对象方法的特点，并对对象技术中的各类关键技术进行详细介绍，从而帮助读者建立面向对象的思维方式，为后续的分析和设计课程打下理论基础。

本章目标

本章是基础章，通过对本章的学习，读者能够快速地掌握面向对象领域的核心概念，了解面向对象技术、系统分析与设计、UML 之间的关系，并建立面向对象的思维方式。

主要内容

(1) 从结构化到面向对象：理解传统结构化方法和面向对象方法之间的思维差异，掌握它们在具体应用中的区别和联系。

(2) 对象技术：掌握对象技术的定义，了解面向对象技术的发展历史，对面向对象技术的优势要有一定的认识。

(3) 对象和类：掌握并理解对象和类的定义以及它们之间的关系。

(4) 对象技术相关原则：掌握抽象、封装、模块化、分层面向对象的基本原则，掌握并理解泛化和多态机制的作用。

(5) 上升到面向对象：掌握面向对象、建模和 UML 之间的关系；对面向对象的建模要有一定的认识。

1.1 从素数问题看面向对象

随着 C++、Java、C# 等面向对象的编程语言的日益普及，面向对象技术已经得到了广泛的应用。从面向对象的编程语言到面向对象软件工程方法也日益被人们重视起来。关于面向对象的更进一步的应用(如面向构件、面向服务、面向模式等)也逐步发展起来，而这些新

方法都是建立在面向对象的思维方式上的。由此可见，深入理解面向对象的思维方式不仅可以帮助我们理解目前面临的应用模式，还是我们进一步学习和发展的必经之路。

很多人都经历过传统的结构化思维方式，让我们先通过一个经典的数据结构中的算法问题来探讨如何走出传统的思维模式。利用面向对象的思维方式来解决问题，进而理解到底什么是面向对象思维方式，为后续的对象建模热身。

1.1.1 问题的提出

我们所面临的是一个求素数的问题，素数(也叫质数)是指除了1与本身之外，不能被其他正整数整除的数。按照习惯规定，1不算素数，最小的素数是2，其余的有3、5、7、11、13、17、19……

根据上面的定义可以推导出判断素数的算法：对于数 n，判断 n 能否被 i($i=2,3,4,5\cdots n-1$)整除，如果全部不能被整除，则 n 是素数，只要有一个能除尽，则 n 不是素数。事实上，为了压缩循环次数，可将判断范围从 $2 \sim n-1$ 改为 $2 \sim \text{sqrt}(n)$。

筛选法是一个经典的求素数的算法，它的作用并不是判断某个数是否是素数，而是求小于数 n 的所有素数，下面通过一个简单的例子来说明筛选法的求解过程。

我们需要求解50以内的所有素数 i($2<i<n$)，为此需要进行如下的筛选(见图1-1)。

筛掉2的倍数：2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 …
筛掉3的倍数：2 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 …
筛掉5的倍数：2 3 5 7 11 13 17 19 23 25 29 31 35 37 41 43 …
筛掉7的倍数：2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 49
留下素数序列：2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

图1-1 筛选法求素数示意图

为了利用程序实现此算法，需要进行算法设计。考虑分别采用结构化方法和面向对象的方法实现此算法，并将其设计方案以合适的方式记录下来。下面将通过对两种不同方法的比较，讲解结构化和面向对象方法在本质上的区别。

1.1.2 传统的结构化解决方案

按照传统的结构化方法，算法的执行过程如下所示。

(1) 首先，以当前最小的数(即2)作为因子，将后面所有可以被2整除的数去掉(因为它们肯定不是素数，参见图1-1的第1行，去掉后面的4、6、8……剩余结果见第2行)。

(2) 之后，取剩余序列中第二小的数(即3)作为因子，将后面所有可以被3整除的数去掉(参见图1-1中的第2行，去掉后面的9、15、21……剩余结果见第3行)。

(3) 如此继续，直到所取得最小数大于 sqrt(n)(图1-1中第4行为最后一次筛选，此时的因子为7，因为下一个因子即为11(大于 sqrt(50)))。

(4) 剩余的序列即为 n 以内的所有素数。

为了更清楚地描述该算法，可以采用流程图来阐述算法流程，该算法的流程图如图1-2所示。

需要说明的是，上述的流程图和前面描述的算法有所出入。在具体实现时，考虑到算法的执行效率，并没有将当前因子的倍数直接删除(因为，如果采用数组存储当前数字序列，则

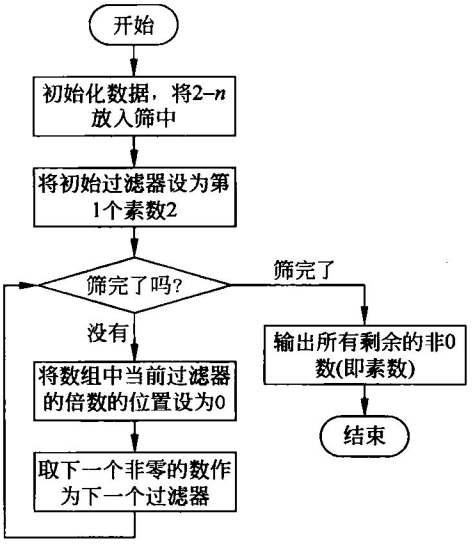

图 1-2 筛选法求素数流程图

删除过程的算法复杂度都为 O(n)),而是将相应的位置零,表明该位置已经没有数据了。

设计出这样一个算法后,后面的实现过程就是水到渠成了,可以选择一种合适的编程语言来实现。下面给出了该算法的 C 语言来实现[①]。

```
void main(){
    int * sieve, n;
     int iCounter = 2, iMax, i;
     printf("Please input max number:");
     scanf(" % d", &n);
     sieve = malloc((n - 1) * sizeof(int))
     for(i = 0;i < n - 1;i++) { sieve[i] = i + 2; }
     iMax = sqrt(n);
     while (iCounter < = iMax) {
         for (i = 2 * iCounter - 2; i < n - 1; i += iCounter)
             sieve[i] = 0;
         iCounter++; }
     for(i = 0; i < n - 1; i++)
         if sieve[i]! = 0 printf(" % d ",sieve[i]);
}
```

1.1.3 面向对象的解决方案

在上面的问题中,可以很容易地从算法描述中构造目标程序,很自然,也似乎很符合人们的思维习惯。那么这种方法是面向对象的方法吗?也许大家都会说不是!因为很明显,

① 需要说明的是,本书提供的代码主要是便于读者理解设计方案,多数代码只是一个片断,并不完整,而且很多示例性的代码语法也很不严格。另外,代码多数采用了 C++或 Java 语法表示。

我们采用的是C语言实现的，而C语言显然是结构化的。什么才算是面向对象的方法呢？如果我现在需要大家用面向对象的方法去解决这个问题，那么大家又应该怎么做呢？

有人也许会说，这很简单，Java是一门真正的面向对象的语言，我用Java去实现这个算法是不是就是面向对象呢？那么，下面就看看该算法的Java实现。

```
import java.lang.Math;
public class PrimerNumber{
    public static void main(String args[]) {
        int n = 50;
        int sieve[] = new int[n - 1];
        int iCounter = 2, iMax, i;
        for(i = 0;i < n - 1;i++) {sieve[i] = i + 2;}
        iMax = (int)Math.sqrt(n);
        while(iCounter < = iMax){
            for (i = 2 * iCounter - 2; i < n - 1; i += iCounter)
                sieve[i] = 0;
            iCounter++;
        }
        for(i = 0; i < n - 1; i++)
        if (sieve[i]! = 0) System.out.println(sieve[i]);
    }
}
```

在这个程序中，我们看到一个面向对象的关键特征——类。为了能够使程序正确地通过编译并运行，我们需要利用Java的关键字class定一个类，并为该类定义相应的成员函数(main函数)，这不就是面向对象吗？原来就这么简单。

真的是这样的吗？我们再仔细看看程序的内部实现，怎么这么面熟？这不是和前面的C程序很类似吗？定义数组、利用for循环初始化，利用while循环控制因子；只不过将语法从C换成了Java，换汤不换药。整个算法实现的思维方式完全没有变化，这显然不是面向对象，这只不过是披着“面向对象皮”的结构化程序，可把它称为“伪面向对象”。

那么怎样才算面向对象的思维方式呢？到底要怎样去做才是一个面向对象的程序呢？在这里先不讨论面向对象的概念或理论(我们在后续章节中会陆续介绍这些内容)，先来看看这个例子如何通过面向对象的方法实现。

我们都知道，在面向对象的方法中，最重要的概念是类和对象，这些算法所要求的功能是通过各个对象之间的交互实现的(这就像我们日常生活一样，为了完成某一件事，需要和各种不同的人、物打交道，这些人和物就是对象，而打交道的过程就是交互)。因此在面向对象的思维方法中，我们并不是关注算法本身，而需要关注为了完成这个算法，需要什么样的“人”和“物”(即对象)，之后再定义这些“人”和“物”之间的关系，从而明确它们是如何“打交道”(即交互)的。把这个过程明确后，事就自然办成了(即实现了算法)。

按照这种思维模式，再来看前面的筛选法求素数的问题是怎样的一个过程。在这个算法中，我们看到了什么？

(1) 首先，看到了一堆需要处理的整数，这些整数构成数据源，这个数据源就是一个待处理对象，针对这个对象即可抽象出一个类：筛子Seive(存储数据源)。

(2) 其次,看到了一个过滤因子,通过这个因子筛选后面的数,这个因子也是一个对象,针对这个对象即可抽象出一个类:过滤器 Factor(记录当前的过滤因子)。

(3) 此外,还看到了一些东西,比如为了能够对数据源进行遍历,需要一个计数器记录当前正在访问的数据值,这个计数器对象即可抽象成类:计数器 Counter(记录当前正在筛选的数据)。

到此为止,我们就从业务场景中找出了为了实现该算法所需要的对象,这些对象就可以满足基本的业务需求。然而,这还不是面向对象方法的全部,还需要进行进一步的抽象。

如果仅仅找到具体的业务对象,还不算真正的面向对象,只不过是"基于对象"罢了。要做到真正的面向对象,还需要执行最关键的一步:抽象。这一步是面向对象领域最难的一步,只有做好了这一步,程序(或软件)才会获得那些面向对象"广告语"中所谓的面向对象的各种好处(比如稳定性、复用性等),否则这一切都是空谈。至于如何对这个例子进行抽象,则是一个非常复杂的问题,在这里并不展开讨论(关于抽象,后面会有专门的章节进行论述)。让我们直接看结果,图 1-3 即为筛选法求素数问题的类图。

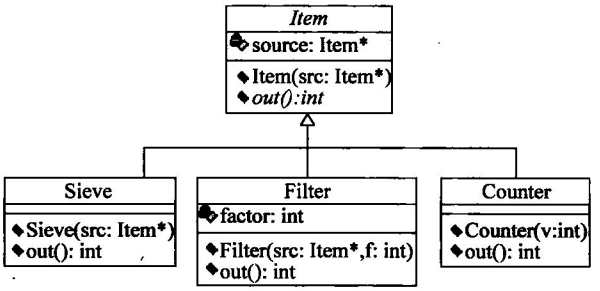

图 1-3 筛选法求素数的类图

从类图中可以看出,除了前面所找出的三个对象的类之外,还定义了一个新的抽象类 Item(在图中类名用斜体字表示),作为这三个类的公有基类。通过该抽象层次,为成员函数 out()提供了多态(在三个子类中分别定义不同的实现)。

类图用于描述系统所需要的类以及它们之间的静态关系。而为了实现具体的算法,还需要通过 UML 中的交互图描述它们之间的交互过程(关于交互过程的描述,后面章节会详细论述),从而实现算法所需要的操作。下面给出了该算法的面向对象的实现(采用 C++语法)。

```
//基类: Item
class Item{
public:
    Item * source;
    Item (Item * src) {source = src;}
    virtual int out() {return 0;}
};
//计数器类: Counter
class Counter: public Item{
    int value;
public:
    int out() {return value++;}
```