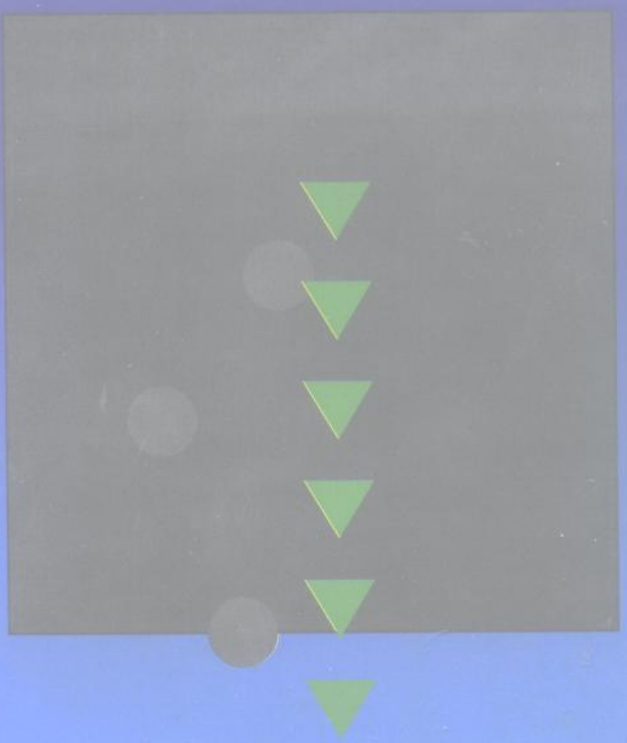
对象技术丛书

Smalltalk 程序设计导论



IBM

[美] IBM公司 著

科学出版社

Smalltalk程序设计导论
[美] IBM公司 著
科学出版社

对象技术丛书

Smalltalk 程序设计导论

〔美〕IBM 公司 著

梅　宏　黄柏素 译

科 学 出 版 社

1998

内 容 简 介

本书介绍一种面向对象程序设计语言——Smalltalk，旨在向不熟悉Smalltalk的开发者介绍Smalltalk编程方法，描述关键的面向对象概念，解释Smalltalk如何支持这些概念的实现，并给出一些用Smalltalk编程的例子。

本书可作为面向对象技术的培训教材，特别适合于新接触Smalltalk及面向对象程序设计的读者。

图书在版编目（CIP）数据

Smalltalk程序设计导论/〔美〕IBM公司著；梅宏，黄柏素译，－北京：科学出版社，1998.7
（对象技术丛书）
书名原文：Introduction to Object-Oriented Programming with IBM Smalltalk
ISBN 7-03-006364-3
Ⅰ.S… Ⅱ.①I… ②梅… ③黄… Ⅲ.面向对象语言，Smalltalk-程序设计 Ⅳ.TP312
中国版本图书馆CIP数据核字（98）第25291号

图字：01-98-0553号

科学出版社出版
北京东黄城根北街16号
邮政编码：100717
北京双青印刷厂 印刷
新华书店北京发行所发行 各地新华书店经售
*
1998年7月第 一 版 开本：787×1092 1/16
1998年7月第一次印刷 印张：9 1/2
印数：1—3 000 字数：210 000

定价：19.00元

《对象技术丛书》前言

国内有关面向对象语言和技术的书籍已经出版不少，为什么还要再出这样一套丛书？道理很简单，这是社会的需要，既是高等学校进行软件工程教学和实践的需要，也是广大从事软件开发和应用的工程技术人员的需要。有人认为，能够用 C++ 编写程序就是掌握了对象技术。这是一种误解。当代软件工程发展正面临着从传统的结构化范型到面向对象范型的转移，这需要有新的语言、新的系统和新的方法学的支持，对象技术就是这种新范型的核心技术。对象技术正在发展成为当代乃至面向下一世纪软件工程发展的主流技术。只有真正深刻理解对象技术的内涵，才能在实践活动中运用自如，进入一个新的境界。本丛书旨在向国内读者全面和深刻地介绍面向对象技术，包括面向对象的语言、方法学、体系结构、技术标准和产品应用等。本丛书部分专题由国内有关对象技术专家撰写，部分专题经 IBM 授权同意，由该公司的技术资料翻译而成。由于对象技术尚在蓬勃发展阶段，各种对象技术的新系统和新产品不断涌现。本丛书没有预先确定一套完整的书目清单，根据先求实不求全的原则，按照对象技术的发展和国内读者的需要，将陆续选定出版的专题，丛书编委会负责指导和组织专题的编写和翻译工作。

希望这套丛书能对我国从事对象技术教学和研究开发的科技人员有所帮助，并能为我国对象技术的应用和软件产业的进步起到推动作用。一本书、一套书的作用总是很有限的，但是它所激发的社会响应却可能很大。这正是我们所追求的。

《对象技术丛书》编委会

1997 年 12 月

导　言

面向对象语言和工具正在变成软件开发者主要的开发环境。特别是 Smalltalk，由于其支持应用的快速开发且易于修改、易于理解，被认为是主要的开发语言之一。

很多正转向 Smalltalk 的开发者对该语言和面向对象概念并不了解。本书旨在向不熟悉 Smalltalk 的开发者介绍 Smalltalk，标识和描述主要的面向对象概念，解释 Smalltalk 如何支持这些概念的实现，描述 Smalltalk 语法，以及给出一些 Smalltalk 程序的例子。

本书的读者

本书面向新接触 Smalltalk 及面向对象程序设计的用户。读完本书后，读者将知道如何构造简单的 Smalltalk 应用程序，并可以容易地转向 Smalltalk 开发环境和更高级的 Smalltalk开发技术(这些内容在“IBM Smalltalk and VisualAge Development Guide 和 IBM Smalltalk Programmer's Reference”一书中描述)。

本书覆盖的内容

本书从第一章的基础介绍开始到第十三章的 Smalltalk 应用示例结束，每章均建立在前面章节的基础上，因此，本书适当的阅读顺序是非常重要的。

每章都有一些例子，其中有些例子可以在 Smalltalk 环境中运行，这需要读者已安装 Smalltalk环境并可正常运行。

本书未涉及的内容

本书不是一本关于 Smalltalk 环境和工具的教材。其讨论的内容仅限于对构造简单的应用所必需的基本 Smalltalk 概念。关于高级应用构造的高级论题，包括用户界面，在“IBM Smalltalk Programmer's Reference”一书中讲述。

本书不是关于操作系统及其操作的教材，我们假定读者已经知道如何打开和关闭窗口，如何使用鼠标，以及如何选择菜单项等。

目　录

第一章　Smalltalk 开发环境

新的术语，新的工具，新的程序设计方式，这些似乎把人弄混了（见图 1.1）。如何学习面向对象的开发？这是否值得努力？为什么选择 Smalltalk？

图 1.1

本书并不试图去说服人们相信面向对象开发或 Smalltalk 的益处，大量已转向面向对象开发的开发者均声称没有比面向对象更好的开发方法已足以说明这一点。这些开发者中的很多人还将告诉你，Smalltalk 是很好的语言选择。传统语言，即使用面向对象的概念改良过，也不能使你达到采用 Smalltalk 编程所能达到的容易程度。

§1.1　Smalltalk 是什么

很多开发者仅把 Smalltalk 视为一种语言，然而，Smalltalk 远远不止于此，它是一个完整的实现面向对象应用的开发环境。Smalltalk 由以下部件组成（见图 1.2）：

- 一种语言
- 一个定义对象的对象模型
- 一组成熟的开发工具
- 大量可复用对象
- 一个支持对象的运行时环境

Smalltalk 中一切都是对象，这也许是关于 Smalltalk 最难于理解的概念，但是，一旦理解了，也正是这个概念使得 Smalltalk 如此容易使用。幸运的是，用户需要理解的 Smalltalk 基本概念并不多，Smalltalk 的其他东西均建立在这些概念之上。

Smalltalk 支持试验并修改的原则。开发环境和运行环境的集成是无缝的。Smalltalk 中修改非常容易，甚至可以对运行中的程序进行修改，修改后的效果也可以在应用中立即看到。

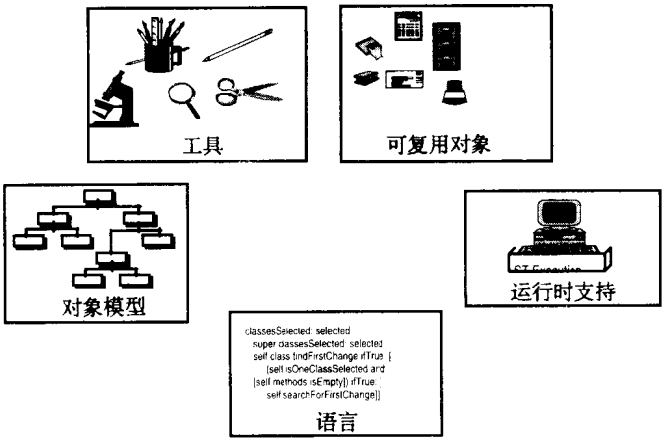


图 1.2

Smalltalk 支持高度的复用。它具有丰富的对象集，可以直接使用或简单地修改以符合应用的需要，用户也可以加入新对象，这些对象可在一个或多个应用中复用。复用将改善生产率和质量。

因此，学习 Smalltalk 应从理解对象概念开始。

§1.2 对象——Smalltalk 的基础

对象是 Smalltalk 开发中的基本建造块，一个 Smalltalk 应用中的所有代码和数据均归属于某对象。应用开发包括开发应用对象和使这些对象进行必要的相互交互以完成应用的任务。

对象是指对外展示行为(behavior)的任何事物，在现实生活中，具有行为的事物通常是名词，如：汽车、动物、机器等。

在应用中同样如此，当查看以 Smalltalk 方式开发的应用时，用户通常会看见很多应用中的名词，如表 1.1 所示。

表 1.1

应　用	对象的例子
银行业务	银行、帐号、存款、顾客
拍卖	拍卖、出价人、拍卖物项、人、地址
订货业务	顾客、顾客订单、产品、销售商
画图	圆、矩形、正文、线

Smalltalk 对象命名约定

在 Smalltalk 中，这些应用对象的名字均以大写字母开头，其他是字母和数字的组合。在上面的应用例子中，我们有对象：Auction, Bank, Customer, Vender, Product, Circle, Rectangle, 等等。

有时需要多词组成的对象名，如"checking account"，然而，对象名不能包含空格。在这种情况下，对象名采用所有单词无空格的组合，每个单词均以大写字母开头，如 checking account 对象的名字是 CheckingAccount, customer order 对象的名字是 CustomerOrder。

每个单词以大写字母开头不是必须的，但它可以改善可读性，如 CustomerOrder 比 Customerorder 更容易读。

注：很多语言用下划线来组合名字的多个部分，如 customer order 命名为 Customer_Order。Smalltalk 也接受这种命名方式。然而，这不是 Smalltalk 开发中被广泛使用的方式，本书将不使用。

Smalltalk 中的名字是区分大小写的，比如可以有两个对象，一个命名为 Customerorder，另一个命名为 CustomerOrder，二者均是可接受的并代表不同的对象。

§1.3 对象的执行(Execute)、显示(Display)和检查(Inspect)

与传统语言不同的是，Smalltalk 不是按照"编辑－编译－连接"的周期来创建程序。在 Smalltalk 中，代码当其被存储或执行时被编译，对象当它们互相传递消息时被连接在一起，此即所谓动态绑定，从而使得 Smalltalk 代码的修改和测试更为容易。

Smalltalk 代码可在正文窗口中输入、编译并立即执行，不需将其存入对象或文件。这个技术可用于执行本书中讨论的例子，当读者涉及这些例子时会对该技术有更清楚的了解。

如果要观察执行(Execute)、显示(Display)和检查(Inspect)技术的工作程序，用户可在正文窗口中输入代码。为了打开一个正文窗口，需先启动 Smalltalk，Smalltalk 的主窗口是 Transcript 窗口，从该窗口，用户可以打开和关闭其他窗口。如果关闭 Transcript 窗口，将关闭所有其他 Smalltalk 窗口并退出 Smalltalk 开发环境。

§1.4 例子——在菜单条上寻找 Execute(执行)、Display(显示)和 Inspect(检查)

激活 Transcript 窗口，打开菜单条上的 File 菜单，选择菜单项 New。这将打开一个正文窗口 Workspace，可在此窗口中输入代码。用户可同时打开多个 Workspace 窗口，但对本书例子来说，一个 Workspace 就足够了。

为了运行代码，应该在 Workspace 的正文窗口中输入程序文本，选择该程序文本，然后从弹出式(pop－up)菜单选取 Execute, Display 或 Inspect 三个命令之一，这些命令要求

Smalltalk 执行被选择的代码，执行方式为：

· Execute 命令执行被选择的代码。

· Display 命令执行被选择的代码，并将执行结果显示在 Workspace 中，结果紧随在被执行代码之后显示并呈被选中状态。

· Inspect 命令执行被选择的代码，并打开一个 Inspector 窗口，该窗口显示代码执行的结果并在需要时查看关于结果的附加信息。

注：Inspect 命令被视为高级功能，因为它需要关于被检查对象的知识。为此，本书中限制该命令的使用。关于 Inspect 窗口的详细讨论参见“IBM Smalltalk and VisualAge Development Guide”一书。

如果被选择的代码有语法错误，在出错的地方将出现一条错误消息，该消息以被选中的正文形式出现。用户必须解决该错误并删去插入的错误消息正文，然后再重新选择该代码并重新选择命令。

注：最好在错误消息仍处于选中态时删去它，这将非常容易做。读完消息但在移动光标前，简单地使用 Delete 或 Backspace 键即可，因为该消息被选中，这将从文本中删除它。另一个方法是用鼠标右键选择 Cut 命令。

如果代码有执行错误，将出现一个 Debug 窗口，用户可使用该窗口去检查和纠正代码中的问题。现在，我们只需简单地关闭该窗口，检查输入的代码以保证没有任何输入错误，最常见的问题是拼写错误或缺少合适的标点符号。

Execute，Display 和 Inspect 命令出现在两个不同的菜单上。你可以打开 Context 菜单，通过将鼠标放在正文区的任何地方并按下鼠标右键；你也可以从菜单条上选择 Edit，将其弹出后选择合适的命令。用户可以选择自己认为最合适的方式。

让我们试一试，打开一个 Workspace 窗口，将鼠标放在 Workspace 的正文窗口中，按下右键，你将看见一个 Context 菜单出现，上面有三个菜单项：Execute，Display 和 Inspect。

让我们看一个例子，在正文窗口中，输入下述语句：

```
Date today
```

确认 Date 中 D 必须大写，Date 和 today 间需留有空格。选择该语句（即显著标记它），然后从菜单上选择 Display，执行的结果为显示今天日期的正文文字，显示在正文窗口中。

注：本书中，术语 Execute，Display 和 Inspect 表示你应该在 Workspace 正文窗口中输入代码，选择该代码，然后选择 Execute，Display 或 Inspect 三个命令之一。

§1.5 什么是对象

前面我们提到，Smalltalk 开发涉及应用中对象的标识。如何标识对象？一旦一个对象被标识，又如何使用它？为了完整地理解对象，你需要理解对象的定义。

第二章　对象的定义

对象的定义包括对象的名字和对象的行为。

第一章中讨论了如何命名对象，记住：一个对象名以大写字母开头，其余是字母和数字的组合，并且不含有空格。对象名的例子有：Customer，CustomerOrder，Rectangle。

名字仅是对象描述的简单部分，最复杂的部分是确定对象的行为。一个对象具有行为的含义是什么？如何在 Smalltalk 中定义对象的行为？为了理解对这些问题的回答，我们需要先弄清对象行为的定义。

一个对象的行为定义有三个部分：接口、代码和数据，下面将详细讨论这三部分。我们也将讨论如何将对象定义组织为类并讨论其他影响对象及其行为定义的关键概念。

§2.1　对象的接口

一个对象是一个可完成一组活动的"东西"，这个活动集合定义了对象的行为。

例如，一个 CheckingAccount 对象可告诉你余款、或加入存款、或处理取款；一个 CustomerOrder 对象可以告诉你一个定单中的产品列表、或在定单中加入一个产品、或发出定单；一个 Customer 对象可以告诉你其名字和地址；一个 Rectangle 对象可以告诉你其长和宽。

对象的接口是一个命令集合，每个命令完成一项特定的动作。一个对象可通过向另一个对象发出消息而请求它完成一个命令，请求对象是发送对象，接收请求的对象是接收对象，也可称为发送者和接收者(见图 2.1)。

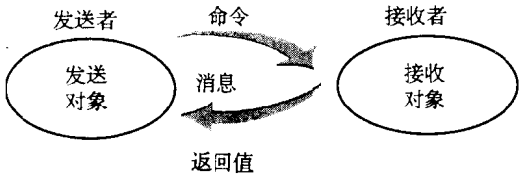


图 2.1

接收对象将拥有控制权直到它完成命令，然后将控制权还给发送对象。

例如，一个 Teller 对象向一个 Customer 对象发送需要其地址的消息，接收者 Customer 对象将其地址返回给发送对象(见图 2.2)。

消息也可以包含发送对象准备传送给接收对象的信息，这种信息称为消息的参数。消息总有一个返回值返回给发送对象，因为 Smalltalk 中一切均为对象，返回值也是对象。对发送对象而言，返回值可能有用也可能无用。

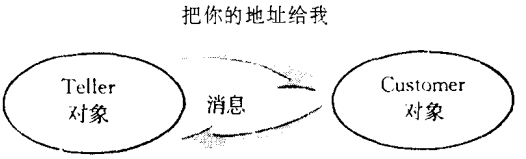


图 2.2

例如，Teller 对象现在想修改顾客的地址，它向 Customer 对象发送一个消息，将其地址设为新值，新地址作为消息的参数传送，在此情况下，Teller 对象并不关心消息的返回值(见图 2.3)。

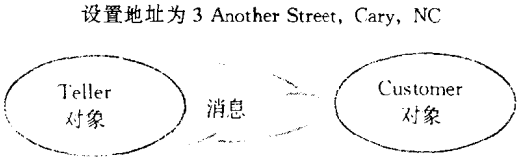


图 2.3

发送给某对象的消息的返回值是由代码的书写者决定的。大多数情况下，返回值是什么是明显的(如在请求告知地址时)，有些情况下，它并不是这样明显。消息的文档将清楚地说明返回值将是什么。

注：消息将总会有返回值，缺省返回值是接收对象自己(除非该消息将其显式地设置为其他值)。在 Smalltalk 中，消息的形式定义并未含有返回值声明，这点和 C++语言不同。Smalltalk 的习惯是通过在代码中加入注释而描述返回值，在本章后部，我们可以看到消息的形式定义都由什么部分构成。

2.1.1 Smalltalk 消息命名约定

上面例子中，Teller 对象通过发送两条消息和 Customer 对象通信：一条是返回顾客的地址，另一条是设置顾客的地址。所有消息被附上名字(也称为选择子)，正是该名字被发送给接收对象。

Smalltalk 中消息名字以小写字母开头，其他为字母和数字的组合，不允许嵌入空格。如果消息名由多个单词组成，则除第一个单词外，其他单词均以大写字母开头，如：phoneNumber。

注：消息名以小写字母开头，并且多字组合的消息名中的单词使用大写字母开头，从而改善了 Smalltalk 代码的可读性。然而，这仅仅是习惯，Smalltalk 也接受以大写字母开头的消息名。

2.1.2 获取和设置信息

一个消息如“give me your address”可命名为 giveMeYourAddress。然而，这个名字太长，可以简单地缩短为 address。

传送信息的消息如“set the address to this address”，采用相同的命名约定，但名字必须以冒号(:)结束，该消息名为 address:。后面我们会讨论消息如何传送此类信息。

2.1.3　Smalltalk 消息的类型

不传送信息的消息是单元消息，如：name，address。

传送信息的消息是关键字消息，如：name:，address:。

一个消息可以包含多个参数，对消息中每一个参数都有对应关键字。如，可以在一条消息中同时设置名字和地址，这就需要消息名具有两个关键字，如：name:address:。这个消息的名字是两个关键字没有嵌入空格的组合。

还有一类消息，称为二元消息，用于刻划算术、比较或逻辑操作。一个二元消息可以是一个或两个字符，它可能包含下面特殊字符的任意组合：

，　+　/　\　*　~　>　<　=　@　%　|　&　?　!

下面是使用二元消息的 Smalltalk 表达式例子：

a + b　　返回 a 和 b 相加的结果

a | b　　返回 a 和 b 相或的结果

a >= b　　比较 a 是否大于或等于 b，返回 true 或 false

所有二元消息自动包含一个参数。在第五章中，我们将更详细地讨论几个二元消息。

注：所有 Smalltalk 名字应该是自标识的，即仅仅通过看名字就可以得到一些关于其目的的线索。这将大大改善语言特别是 Smalltalk 这类消息传递语言的可读性，但是，输入长的名字也是烦琐的，应该在长度和可读性间适当权衡。

在 Smalltalk 中，经常会看到消息名指出了某类被请求或设置的信息，如获取或设置一个 Rectangle 对象的高度和宽度的消息名是：height，width，height: 和 width:。这将使 Smalltalk 代码易读易理解。

2.1.4　消息传递和程序控制流

为了完成任务，接收对象发送其他消息给自己或其他对象是很常见的事，这是一个连续的操作系列，直到所有的其他消息均完成后控制权才返回给最源头的发送对象。例如，在下面的图中，对象 A 发送一个消息给对象 B，对象 B 为了处理这个消息，发送一个消息给对象 C，类似地，对象 C 发送消息给对象 D，对象 D 返回给对象 C，然后返回给对象 B，最后返回给对象 A。直到所有其他消息均完成后，控制权才返回到对象 A（见图 2.4）。

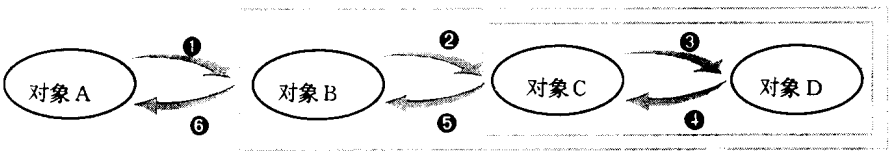


图 2.4

§2.2　对象的逻辑

每个消息均有代码和其相关联，只要一个对象接收到消息，相应的代码将被执行。对

象的消息定义了对象的行为，对象的代码确定了对象如何处理每一个消息。

和每一个消息关联的代码是一个方法。当一个对象接收到一个消息，对象确定该消息请求的返回并将控制权传给和该消息相关联的方法。由于消息和方法间的紧密耦合，消息名也称为方法名。

注： 方法类似于如 C, PASCAL, COBOL 等过程型语言中的子例程、过程或函数。如方法名等价于子例程名，方法代码等价于子例程中的代码，向一个对象发消息类似于调用子例程。

操作于特定对象的方法称为实例方法，激活实例方法的消息称为实例消息，本书将在后面更清楚地阐述有关的概念。

在前面的例子中，name, name:, address, address:和 name:address:是 Customer 对象的方法名(见图 2.5)。

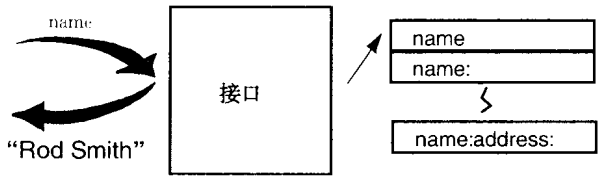


图 2.5

图 2.5 示意向 Customer 对象发送 name 消息，该消息将控制权传给定义在 Customer 中的 name 方法。

§2.3 对象的数据

一个对象为完成其定义的行为需保持一定信息，例如，Rectangle 对象必须含有关于其形状的信息以返回宽度和高度信息。

大多数对象包含有支持它们行为的变量，这些变量称为实例变量。这类数据存在于对象的生命期并定义对象的状态。只有对象的实例方法可以引用和改变存储在实例变量中的值，其他对象的实例方法不能引用本对象的数据，对象只能通过发送消息而访问另一个对象的数据。

在图 2.6 中，实例变量有 variableOne 到 variableX，实例方法是 methodOne 到 methodN。发送者不能直接引用实例变量，只能通过使用方法 methodOne 到 methodN 来访问对象的数据。

注： 过程型程序设计经常使用公共数据区来共享信息。对这些数据区域的引用是由需要查看或设置这些信息的程序直接进行的。

面向对象程序设计不鼓励对公共数据的直接访问(除了全局变量的使用)，而是所有的数据应归某对象拥有，只有该对象可以访问或改变其数据，其他对象通过向数据的拥有者发送消息而访问或设置该数据。

读者可能注意到实例变量名和方法名可以是相同的，例如，Customer 有方法 name、