Java 面向对象程序设计

MIANXIANG DUIXIANG CHENGXU SHEJI

李素若　陈万华　张　牧　编著

化学工业出版社

Java 面向对象程序设计

李素若　陈万华　张牧　编著

化学工业出版社
·北京·

本书主要讲述 Java 程序设计的基础知识，以及面向对象程序设计的基本思想及主要特点。全书内容丰富、生动活泼、由浅入深，特别注重实用性。书中包含大量精心设计并调试通过的编程实例，方便初学者学习。

全书共有 12 章，主要内容包括：Java 概述、Java 语言基础、类与对象、继承与多态、接口与内部类、异常处理与输入/输出、图形用户界面设计、Swing 组件、Java Applet 程序、Java 网络编程、Java 高级应用和上机实验题。

本书适合作为普通高等院校计算机科学与技术专业教材，也可作为高职高专计算机专业教材，并可供相关工程技术人员参考。

图书在版编目(CIP)数据

Java 面向对象程序设计/李素若，陈万华，张牧编著
北京：化学工业出版社，2010.10
ISBN 978-7-122-09356-1

Ⅰ.J… Ⅱ.①李… ②陈… ③张… Ⅲ.JAVA 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2010）第 164644 号

责任编辑：王听讲　　装帧设计：刘丽华
责任校对：陈　静

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号　邮政编码 100011）
印　　刷：北京永鑫印刷有限责任公司
装　　订：三河市万龙印装有限公司
787mm×1092mm　1/16　印张 21¾　字数 564 千字　　2010 年 10 月北京第 1 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686）　售后服务：010-64518899
网　　址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：38.00 元

前　言

Java 语言经过近 20 年的发展完善，其功能日益强大，应用的领域越来越广。为什么 Java 能以极快的速度推广应用？首先，Java 是面向对象的语言。随着软件工程技术的不断发展，面向对象编程技术已经成为当前软件开发的主要手段之一。其次，Java 语言是随着 Internet 的广泛应用而发展起来的。Java 的跨平台特性非常适合于在 Internet 上应用，它已经成为网络编程的首选语言。另外，Java 也能胜任科学计算和工程模拟方面的应用。它与传统的 Fortran77、Fortran90 以及 C++比较，并不逊色。

在本书的编写过程中，编者结合自己的教学和编程实践经验，力图用生动、通俗易懂的语言，并结合编程实例来讲解各个知识点，便于读者理解和掌握。全书共 12 章，第 1 章 Java 概述，主要讲述 Java 的发展历史，其特点和开发环境，读者可以了解到 Java 程序是如何做到跨平台程序设计的。第 2 章介绍了 Java 语言的基本语法，读者可以了解到 Java 程序的基本结构和结构化编程的方法。第 3 章介绍了类与对象的基本概念、类的声明、对象的生成与销毁、类的组织。第 4 章介绍了与类的继承与多态性有关的内容，包括类的继承、多态性概念及实现方法、Object 类、抽象类和最终类介绍，以及 Java 包的应用。第 5 章介绍了 Java 接口与内部类的基本概念和用法。第 6 章介绍了异常概念、Java 的异常处理类、异常处理机制，如何创建和使用用户自己定义的异常类；还介绍了 I/O 流的概念，并以此详细介绍了 Java 字节流类、字符流类和文件类。第 7 章介绍了图形用户界面设计的基本概念、Java 的 AWT 事件处理机制及 AWT 图形设计。第 8 章介绍了 Swing 基本组件及应用。第 9 章介绍了 Java Applet 的概念、特点及应用。第 10 章介绍了 Java 的网络编程的基本知识。第 11 章介绍了 Java 的多线程的概念及应用，同时介绍了在 Java 程序中如何实现对数据库的操作。

本书特别强调应用，教师在教学中可以通过实例讲解各种概念和基本应用，使学生一目了然，而且能加深理解，以激发学生学习兴趣；课后通过每章小结和习题，便于读者掌握各章的重点和难点，并进行必要的训练；为了方便学生上机实践，本书专门设计了 8 套上机实验题，供学生在每章学习过后上机练习。

本书第 1～4、7～8、12 章由李素若编写，第 5～6、11 章由张牧编写，第 9～10 章由陈万华编写，全书由李素若负责审核和统稿。参加本书编写大纲讨论的教师还有游明坤、胡玉荣、任正云、严永松、武永成等。

由于编者水平有限，加之时间仓促，书中难免有疏漏之处，敬请广大读者批评指正，以使本书质量得到进一步提高。

编著者

2010 年 8 月

目　录

第 1 章　Java 概述 ····· 1
1.1　面向对象的程序设计 ····· 1
1.1.1　什么是面向对象程序设计 ····· 1
1.1.2　面向对象程序设计的基本概念 ····· 2
1.1.3　传统程序设计方法的局限性 ····· 4
1.1.4　面向对象程序设计的主要优点 ····· 5
1.2　Java 概述 ····· 6
1.2.1　Java 的起源与发展 ····· 6
1.2.2　Java 语言的特点 ····· 7
1.3　Java 与 C/C++比较 ····· 9
1.4　Java 虚拟机工作原理 ····· 11
1.5　Java 的开发和运行环境 ····· 12
1.5.1　JDK 三种形式 ····· 12
1.5.2　J2SE 的主要内容 ····· 13
1.5.3　JDK 下载、安装和配置 ····· 13
1.6　开发和运行 Java 程序的步骤 ····· 15
1.6.1　选择编辑工具 ····· 15
1.6.2　编译和运行 Java 程序 ····· 16
小结 ····· 16
习题 ····· 16
第 2 章　Java 基础 ····· 18
2.1　Java 的基本组成 ····· 18
2.1.1　标识符 ····· 18
2.1.2　关键字 ····· 18
2.1.3　数据 ····· 18
2.1.4　运算符 ····· 20
2.1.5　分隔符 ····· 20
2.2　Java 基本数据类型 ····· 21
2.2.1　整数类型 ····· 21
2.2.2　浮点类型 ····· 21
2.2.3　逻辑类型 ····· 21
2.2.4　字符类型 ····· 22
2.3　Java 运算符与表达式 ····· 22
2.3.1　运算符的优先级 ····· 22

2.3.2 算术运算符……23
2.3.3 赋值运算符……23
2.3.4 关系运算符……24
2.3.5 逻辑运算符……24
2.3.6 位运算符……25
2.3.7 条件运算符……26
2.3.8 类型转换……26
2.4 基本输入输出语句……27
2.5 结构化程序设计……28
2.5.1 顺序结构……29
2.5.2 选择结构……29
2.5.3 循环结构……32
2.5.4 跳转语句……34
2.6 数组……36
2.6.1 数组的声明与创建……37
2.6.2 数组的初始化……37
2.6.3 数组的引用……38
2.6.4 多维数组……38
2.7 方法……41
2.7.1 方法声明……41
2.7.2 方法调用……42
2.7.3 方法重载……44
2.7.4 参数传递……45
2.8 字符串处理……46
2.8.1 字符数组与字符串……46
2.8.2 字符串……47
2.8.3 字符串操作……48
2.8.4 字符串数组……50
小结……50
习题……51
第3章 类与对象……53
3.1 类的定义……53
3.1.1 类的声明……53
3.1.2 类体……54
3.1.3 构造方法……56
3.2 对象……57
3.2.1 对象的创建……57
3.2.2 对象的使用……58
3.2.3 对象销毁……60
3.2.4 对象初始化……61
3.3 访问属性控制……62

3.3.1 默认访问属性 62
3.3.2 public 63
3.3.3 private 64
3.3.4 protected 66
3.4 静态成员 66
3.4.1 静态成员变量 66
3.4.2 静态成员方法 68
3.4.3 静态代码块 69
3.4.4 main()方法 70
3.5 final、this 和 null 71
3.5.1 final 71
3.5.2 this 72
3.5.3 null 73
3.6 包 73
3.6.1 包的概念 74
3.6.2 包的声明 74
3.6.3 包的使用 75
3.6.4 常用系统包简介 77
小结 78
习题 79
第 4 章 继承与多态 82
4.1 类的继承 82
4.1.1 继承的基本概念 82
4.1.2 继承实现 84
4.1.3 子类的构造方法 86
4.2 类成员的隐藏与重载 87
4.2.1 成员变量的继承和隐藏 87
4.2.2 方法的继承、重载和覆盖 88
4.2.3 this 和 super 92
4.2.4 构造方法的继承和重载 96
4.2.5 父对象与子对象的转换 98
4.3 多态性 99
4.3.1 多态性概述 100
4.3.2 参数多态性示例 101
4.4 抽象类和最终类 102
4.4.1 抽象类 102
4.4.2 最终类 104
4.5 Object 类 104
4.5.1 Object 概述 104
4.5.2 Object 常用方法 105
小结 108
习题 108

第 5 章　接口与内部类 ····· 111
5.1　接口能够解决的问题 ····· 111
5.1.1　接口的概念 ····· 111
5.1.2　定义接口 ····· 113
5.1.3　接口的特点和实现 ····· 115
5.2　Comparable 接口 ····· 121
5.3　回调 ····· 124
5.3.1　回调的概念 ····· 124
5.3.2　Java 语言的回调（callback）机制 ····· 124
5.4　内部类 ····· 126
5.4.1　内部类的概念和使用 ····· 126
5.4.2　成员内部类 ····· 127
5.4.3　局部内部类 ····· 129
5.4.4　静态内部类 ····· 130
5.4.5　匿名内部类 ····· 131
小结 ····· 132
习题 ····· 132
第 6 章　异常处理与输入/输出 ····· 134
6.1　异常处理 ····· 134
6.1.1　异常的概述 ····· 134
6.1.2　Java 中的异常类 ····· 135
6.1.3　异常处理机制 ····· 136
6.1.4　异常的捕获与处理 ····· 137
6.1.5　throw 和 throws 语句 ····· 139
6.1.6　定义自己的异常类 ····· 142
6.2　输入/输出流的基本概念 ····· 144
6.3　输入/输出类 ····· 145
6.3.1　字节流 InputStream 类和 OutputStream 类 ····· 146
6.3.2　字符流 Reader 类和 Writer 类 ····· 147
6.3.3　标准输入/输出 ····· 148
6.4　文件的顺序访问 ····· 149
6.4.1　输入/输出流操作的一般步骤 ····· 149
6.4.2　字节流类 ····· 149
6.4.3　字符流（Reader 类和 Writer 类） ····· 155
6.5　文件的随机访问 ····· 158
6.5.1　建立随机访问文件流对象 ····· 158
6.5.2　读/写随机访问文件方法 ····· 158
6.5.3　文件指针及相关方法 ····· 159
6.6　目录和文件管理 ····· 160
6.6.1　目录管理 ····· 160
6.6.2　文件管理 ····· 160

小结 …… 161
习题 …… 162
第 7 章　图形用户界面设计 …… 164
7.1 Java 图形用户界面概述 …… 164
7.1.1 AWT …… 164
7.1.2 Swing …… 165
7.2 AWT 组件概述 …… 165
7.3 文本编辑组件 …… 171
7.3.1 文本行 …… 171
7.3.2 文本区 …… 173
7.4 布局管理 …… 174
7.4.1 边界布局（BorderLayout） …… 174
7.4.2 流式布局（FlowLayout） …… 176
7.4.3 网格布局（GridLayout） …… 177
7.4.4 卡片布局（CardLayout） …… 178
7.5 事件处理 …… 179
7.5.1 Java 事件处理基本概念 …… 179
7.5.2 Java 委托事件处理机制 …… 180
7.5.3 事件监听器类编写要点 …… 181
7.5.4 事件类和监听器接口 …… 182
7.5.5 处理 ActionEvent 事件 …… 185
7.5.6 处理 ItemEvent 事件 …… 186
7.5.7 处理 TextEvent 事件 …… 187
7.5.8 处理 KeyEvent 事件 …… 188
7.5.9 处理 MouseEvent 事件 …… 190
7.5.10 处理 WindowEvent 事件 …… 193
7.6 绘图 …… 195
小结 …… 197
习题 …… 198
第 8 章　Swing 组件 …… 199
8.1 Swing 组件的概述 …… 199
8.2 使用 Swing 的基本规则 …… 200
8.3 标签和按钮 …… 202
8.4 文本编辑组件 …… 202
8.4.1 文本行 JTextField …… 203
8.4.2 密码行 JPasswordField …… 203
8.4.3 文本区 JTextArea …… 203
8.5 选择组件 …… 204
8.5.1 单选按钮 …… 204
8.5.2 复选框 …… 206
8.5.3 列表框 …… 209

8.5.4 组合框 211
8.6 菜单 213
8.6.1 菜单栏 213
8.6.2 菜单类 213
8.6.3 菜单项 214
小结 216
习题 216
第 9 章 Java Applet 程序 217
9.1 Java Applet 基础 217
9.1.1 Applet 类的继承关系 218
9.1.2 Applet 的创建 219
9.1.3 Applet 生命周期 221
9.1.4 Applet 类的显示方法 224
9.2 Applet 标记 225
9.2.1 Applet 定位属性 226
9.2.2 Applet 代码属性 226
9.2.3 用于非 Java 兼容浏览器的 Applet 属性 227
9.2.4 向 Applet 传递消息的参数属性 227
9.3 在 Applet 程序中添加组件 227
9.4 Applet 通信 229
9.4.1 同页 Applet 间的通信 229
9.4.2 Applet 与浏览器之间的通信 231
9.5 Applet 与 Application 234
9.5.1 将应用程序转化为 Applet 234
9.5.2 将 Applet 转化为应用程序 236
9.6 Applet 安全机制 236
9.7 Applet 的多媒体支持 237
9.7.1 图像 237
9.7.2 声音 242
9.7.3 动画 246
小结 249
习题 250
第 10 章 Java 网络编程 251
10.1 Java 网络编程概述 251
10.1.1 Java 与网络编程 251
10.1.2 TCP/IP 与 UDP 252
10.2 基于 URL 的网络编程 253
10.2.1 URL 基础知识 254
10.2.2 URL 类 254
10.2.3 URL 的创建 257
10.2.4 使用 URL 获取网络资源 257

10.3 InetAddress 类 …… 264
10.4 基于 Socket 的网络编程 …… 265
10.4.1 Socket 通信的一般结构 …… 266
10.4.2 TCP Socket 编程 …… 267
10.4.3 UDP Socket 编程 …… 274
10.5 综合示例：聊天室程序 …… 279
小结 …… 282
习题 …… 283
第 11 章 Java 的高级应用 …… 284
11.1 线程概述 …… 284
11.1.1 进程与线程 …… 285
11.1.2 线程的概念模型 …… 285
11.1.3 实现线程的类(Thread 类) …… 285
11.1.4 线程的状态 …… 286
11.2 创建和启动线程 …… 287
11.2.1 创建线程 …… 288
11.2.2 启动线程 …… 290
11.3 JDBC 基础 …… 292
11.3.1 JDBC Driver …… 293
11.3.2 JDBC API …… 294
11.3.3 创建 JDBC 应用 …… 298
11.4 JDBC 应用实例 …… 303
11.4.1 建立数据源 …… 303
11.4.2 数据库编程实例 …… 306
小结 …… 312
习题 …… 312
第 12 章 上机实验题 …… 313
12.1 实验一 Java 的安装、配置与运行 …… 313
12.2 实验二 Java 基本语法练习 …… 314
12.3 实验三 面向对象编程练习 …… 319
12.4 实验四 Java 包、接口和异常的使用 …… 323
12.5 实验五 Java 图形用户界面 …… 327
12.6 实验六 Java Applet 技术 …… 328
12.7 实验七 Java 网络编程应用 …… 330
12.8 实验八 Java 高级应用 …… 332
参考文献 …… 336

第 1 章　Java 概述

Java 语言是由 Sun 公司于 1995 年 5 月 23 日正式推出的纯面向对象的程序设计语言，是当前最流行的一种网络编程语言，推出不久，就获得了极大的成功。这是因为 Java 与传统的计算机语言相比，更简洁，更安全，易学易用，并独具特色，本章首先介绍面向对象程序设计的基本概念，然后介绍 Java 语言的起源与发展及其基本特点、Java 与 C/C++比较，Java 与 Internet 及 HTML 的关系、Java 虚拟机工作原理和 Java 的开发工具包 JDK。

1.1　面向对象的程序设计

Java 语言是面向对象的程序设计语言，因此在系统学习 Java 语言前，首先需要对面向对象的程序设计思想有一个了解，并在以后的学习过程中不断加深理解并掌握面向对象的方法。

1.1.1　什么是面向对象程序设计

面向对象程序设计是一种新的程序设计范型（Paradigm）。程序设计范型是指设计程序的规范、模型和风格，是一类程序设计语言的基础。一种程序设计范型体现了一类程序设计语言的主要特征，这些特征能用以支持应用领域内所希望的设计风格。不同的程序设计范型有不同的程序设计技术和方法学。

面向过程程序设计范型是使用较广泛的程序设计范型，这种范型的主要特征是，程序由过程定义和过程调用组成，即程序=过程+调用。基于面向过程程序设计范型的语言称为面向过程性语言，如 C、PASCAL、Ada 等都是典型的面向过程性语言。函数式程序设计范型也是较为流行的程序设计范型，其主要特征是，程序被看做“描述输入与输出之间关系”的数学函数。LISP 是支持这种范型的典型语言。除了面向过程程序设计范型和函数式程序设计范型外，还有许多其他的程序设计范型，如模块程序设计范型（典型语言是 Modula）、逻辑式程序设计范型（典型的语言是 PROLOG）、进程式程序设计范型、类型系统程序设计范型、事件程序设计范型、数据流程程序设计范型等。

面向对象程序设计是一种新型的程序设计范型。这种范型的主要特征是：

程序=对象+消息

面向对象程序的基本元素是对象，面向对象程序的主要结构特点是：第一，程序一般由类的定义和类的使用两部分组成，在程序中定义各个对象并规定它们之间传递消息的规律。第二，程序中的一切操作都是通过向对象发送消息来实现的，对象接收到消息后，启动有关方法来完成相应的操作。一个程序中涉及的类，可以由程序设计者自己定义，也可以使用现成的类（包括类库中为用户提供的类和他人已构建好的）。尽量使用现成的类，是面向对象程序设计范型所倡导的程序设计风格。

需要说明的是，某一种程序设计语言不一定与一种程序设计范型相对应。实际上存在有具备两种范型或多种范型的程序设计语言，即混合型语言。例如 C++就不是纯粹的面向对象程序设计范型，而是面向过程程序设计范型和面向对象程序设计范型的混合型程序设计语言。

1.1.2　面向对象程序设计的基本概念

1. 对象

首先需要搞清楚的问题是：什么是对象？对象具有两方面的含义，即在现实生活中的含义和在计算机世界中的含义。

在我们所生活的现实世界中，“对象”无处不在。在我们身边存在的一切事物都是对象，例如一粒米、一本书、一个人、一所学校，甚至一个地球，这些都是对象。除去这些可以触及的事物是对象外，还有一些无法整体触及的抽象事件，例如一次演出、一场球赛、一次借书等也是对象。

一个对象既可以非常简单，又可以非常复杂，复杂的对象往往是由若干个简单对象组合而成的。

所有这些对象，除去它们都是现实世界中所存在的事物外，它们都还是有各自不同的特征。例如一粒米，首先是一粒米这样一个客观存在。再例如一个人，首先是一个客观实体，具有一个名字来标识，其次具有性别、年龄、身高、体重等这些体现其自身状态的特征；另外还具有一些技能，例如会说英语、会修电器等。

通过上面的这些举例我们可以对“对象”下一个定义，即对象是现实世界中的一个实体，具有如下特性：

- 有一个名字以区别于其他对象；
- 有一个状态用来描述其某些特征；
- 有一组操作，每一个操作决定对象的一种功能或行为；
- 对象的操作可分为两类：一类是自身承受的操作，一类是施加于其他对象的操作。

在面向对象程序设计中，对象是描述其属性的数据以及对这些数据施加的一组操作封装在一起构成的统一体。对象可以被认为是：数据+操作。对象所能完成的操作表示其动态行为，通常也把操作称为方法。

为了帮助读者理解对象的概念，图 1-1 形象地描述了具有 3 个操作的对象。

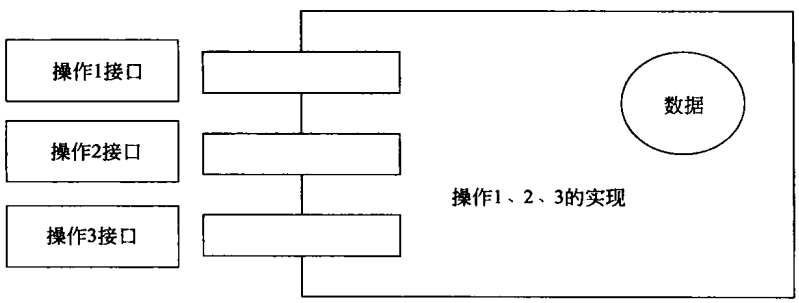

图 1-1　3 个操作对象

下面我们用一台录音机比喻一个对象，通俗地说明对象的某些特点。

录音机上有若干按键，如 Play（播放）、Rec（录音）、Stop（停止）、Rew（倒带）等，当人们使用录音机时，只要根据自己的需要如放音、录音、停止、倒带等按下与之对应的键，录音机就会完成相应的工作。这些按键安装在录音机的表面，人们通过它们与录音机交互。我们无法操作录音机的内部电路，因为电路被装在机壳里，录音机的内部情况对于用户来说是隐蔽的，不可见的。

一个对象很像一台录音机，当在软件中使用一个对象时，只能通过对象与外界的接口操作它。对象与外界的接口也就是该对象向公众开放操作。使用对象向公众开放的操作就好像

使用录音机的按键，只需知道该操作的名字（如录音机的键名）和所需要的参数（用于提供附加信息或设置状态，好像听录音前先装录音带并将录音带转到指定位置），根本无需知道实现这些操作的方法。事实上，实现对象操作的代码和数据是隐藏在对象内部的，一个对象好像是一个黑盒子，表示它内部的数据和实现各个操作的代码，都被封装在这个黑盒子里，在外面是看不见的，更不能从外面去访问或修改这些数据或代码。

使用对象时只需知道对象向外界提供的接口形式而无需知道其内部的实现算法，不仅使得对象的使用变得非常简单、方便，而且具有很高的安全性和可靠性。可见面向对象程序设计中的对象来源于现实世界，更接近人的思维。

2. 类

在现实世界中，类是一组相同属性和行为的对象的抽象。例如，张三、李四、王五……虽然每个人的性格、爱好、职业、特长等各不相同，但是基本特征是相似的，都具有相同的生理构造，都能吃饭、说话、走路等，于是把他们统称为“人”，而具体的每一个人是人类的一个实例，也就是一个对象。

类和对象之间的关系是抽象和具体的关系。类是多个对象进行综合抽象的结果，一个对象是类的一个实例。例如“狗”是一个类，它是由千千万万个具体的不同的狗抽象而来的一般概念。同理，鸡、鸭、牛、羊等都是类。

类在现实世界中并不真正存在。例如，在地球上并没有抽象的“人”，只有一个个具体的人，如张三、李四、王五……同样，世界上没有抽象的“学生”，只有一个个具体的学生。

面向对象程序设计中，类就是具有相同的数据和相同的操作的一组对象的集合，即类是对具有相同数据结构和相同操作的一类对象描述。例如，“学生”类可以由学号、姓名、性别、成绩等表示其属性的数据项和对这些数据的录入、修改和显示等操作组成。在 Java 语言中把类中的数据称为数据成员。

在面向对象中，总是先声明类，再由类生成对象。类是建立对象的“模板”，按照这个模板所建立的一个个具体的对象，就是类的实际例子，通常称为实例。例如，手工制作月饼时，先雕刻一个有凹下图案的木模，然后在木模上抹油，接着将事先揉好的面塞进木模里，用力挤压后，将木模反扣在桌上，一个漂亮的图案就会出现在月饼上了。这样一个接着一个的，就可以制造出外形一模一样的月饼。这个木模就好比是“类”。制造出来的糕点好比是“对象”。

3. 消息

现实世界中的对象不是孤立存在的实体，而是相互间存在着各种各样的联系，正是它们之间的相互作用、联系和连接，才构成了世间各种不同的系统。同样，在面向对象程序设计中，对象之间也需要联系，称为对象的交互。面向对象程序设计必须提供一种机制允许一个对象与另一个对象的交互。这种机制称为消息传递。即对象之间进行通信的结构称为消息。在对象的操作中，当一个消息发送给某个对象时，消息包含接收对象去执行某种操作的信息。发送一条消息至少要包括说明接受消息的对象名、发送给该对象的消息名（即对象名、方法名）。一般还要对参数加以说明，参数可以是认识该消息的对象所知道的变量名，或者是所有对象都知道的全局变量名。

在面向对象程序设计中的消息传递实际是对现实世界中的信息传递的直接模拟。以实际生活为例，我们每一个人可以为他人服务，也可以要求他人为自己服务。当我们需要他人为自己服务时，必须告诉他们我们需要的是什么服务，也就是说，要向其他对象提出请求，其他对象接到请求后，才会提供相应的服务。

一般情况下，我们称发送消息的对象为发送者或请求者，接收消息的对象为接收者或目标对象。对象中的联系只能通过消息传递来进行。接收对象只有在接收到消息时，才能被激活，被激活的对象会根据消息的要求完成相应的功能。

消息具有三个性质。

- 同一对象可接收不同形式的多个消息，产生不同的响应；
- 相同形式的消息可以送给不同的对象，所产生的响应可以是截然不同的；
- 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以对消息不予理会，对消息的响应并不是必需的。

在面向对象程序设计中，消息分为两类，即公有消息和私有消息。到底哪些消息是公有消息，哪些消息是私有消息，需要有一个明确的规定。若有一批消息同属于一个对象，其中有一部分是由外界对象直接发送的，称为公有（public）消息；还有一部分则是它自己向本身发送的，这些消息是不对外开放的，外界不必了解它，称为私有（private）消息。

4. 方法

在面向对象程序设计中，要求某一个对象做操作时，就向该对象发送一个相应的消息，当对象接收到发向它的消息时，就调用有关方法，执行相应的操作。方法就是对象所能执行的操作。方法包括界面和方法体两部分。方法的界面也就是消息的模式，给出方法的调用协议；方法体则是实现某种操作的一系列计算步骤，也就是一段程序。消息和方法的关系是：对象根据接收到消息，调用相应的方法；反过来，有了方法，对象才能响应相应的消息。所以消息模式与方法界面应该是一致的。同时，只要方法界面保持不变，方法体的改动不会影响方法的调用。在 Java 语言中方法是通过函数来实现的，称为成员函数。

1.1.3　传统程序设计方法的局限性

当今社会是信息社会，信息社会的灵魂是作为“信息处理机”的电子计算机，从 1946 年第一台计算机 ENIAC 问世到今天的“深蓝”，电子计算机的硬件得到突飞猛进的发展，程序设计的方法也随之不断的进步。20 世纪 70 年代以前，程序设计方法主要采用流程图，结构化设计（Structure Programming，SP）也趋成熟，整个 20 世纪 80 年代 SP 是主要的程序设计方法。然而，随着信息系统的加速发展，应用程序日趋复杂化和大型化。传统的软件开发技术难以满足发展的新要求。

1. 传统程序设计开发软件的生产效率低下

早期计算机存储器的容量非常小，人们设计程序时首先考虑的问题是如何减少存储器的开销，硬件的限制不容许人们考虑如何组织数据与逻辑，程序本身短小，逻辑简单，也无需人们考虑程序设计方法问题。与其说程序设计是一项工作，倒不如说是程序员的个人技艺。但是，随着大容量存储器的出现及计算机技术的广泛应用，程序编写越来越困难，程序的大小以算术基数递增，而程序的逻辑控制难度则以几何基数递增，人们不得不考虑程序设计的方法。相对于硬件的快速发展，软件的生产能力还比较低下，开发这周期长、效率低、费用不断上升，以至出现了所谓的“软件危机”。

然而，尽管传统的程序设计语言经历了第一代语言、第二代语言以及第三代语言的发展过程，但是其编制程序的主要工作还是围绕着设计解题过程来进行的，故称为面向过程的程序设计，传统的程序设计语言为过程性语言。这种传统程序设计的生产方式仍是采用较原始的方式进行，程序设计基本上还是从语句一级开始。软件的生产缺乏大粒度、可重用的构件，软件的重用问题没有得到很好解决，从而导致软件生产的工程化和自动化屡屡受阻。

影响软件生产效率低的另一个原因是问题的复杂性。随着计算机技术的大规模推广，软件的应用范围越来越广，软件的规模越来越大，要解决的问题越来越复杂。传统程序设计的特点是数据与其操作分离，而且对同一数据的操作往往分散在程序的不同的地方。这样，如果一个或多个数据的结构发生了变化，那么这种变化将涉及程序的很多部分甚至遍及整个程序，致使许多函数和过程必须重写，严重时会导致整个软件结构崩溃。这就是说，传统程序的复杂性控制是一个很棘手的问题，这也是传统程序难以重用的一个重要原因。

维护是软件生命周期中的最后一个环节，也是非常重要的一个环节。传统程序设计是面向过程的，其数据和操作分离的结构，使得维护数据和处理数据的操作过程要花费大量的精力和时间，严重地影响了软件的生产效率。

总之，要提高软件生产的效率，就必须很好地解决软件的重用性、复杂性和可维护性问题。但是传统的程序设计是难以解决这些问题的。

2. 传统程序设计难以应付日益庞大的信息量和多样性的信息类型

随着计算机科学与技术的飞速发展和计算机应用的普及，当代计算机的应用领域已从数值计算扩展到了人类社会的各个方面，所处理的数据已从简单数字和字符，发展为具有多种格式的多媒体数据，如文本、图形、图像、影像、声音等，描述的问题从单纯的计算机问题到仿真复杂的自然现象和社会现象。于是，计算机处理的信息量与信息类型迅速增加，程序的规模日益庞大，复杂度不断增加。这些都要求程序设计语言有更大的信息处理能力。然而，面对这些庞大的信息量和多样的信息格式，传统程序设计方法是无法应付的。

3. 传统的程序设计难以适应各种新环境

当前，并行处理、分布式、网络和多机系统等，已经或将是程序运行的主要方式和主流环境。这些环境的一个共同的特点是都具有一些有独立处理能力的节点，节点之间有通信机制，即以消息传递进行联络。显然传统的程序设计技术很难适应这些新环境。

综上所述，传统的程序设计不能够满足计算机技术的迅猛发展的需要，软件开发迫切需要一种新的程序设计范型的支持。那么，面向对象程序设计是否能担当此任呢？下面我们再分析一下面向对象程序设计的一些优点。

1.1.4 面向对象程序设计的主要优点

从面向过程程序设计到面向对象程序设计是软件开发史上的一个里程碑。面向对象程序设计主要将精力集中处理对象的设计和研究上，从而改变了过去人们设计软件的思维方式，即程序员主要设计和研究的是数据格式和过程，这样极大地减少了软件开发的复杂性，提高了软件开发的效率。面向对象程序设计主要具有以下优点：

1. 可提高程序的重用性

重用是提高软件开发效率的最主要的方法，传统程序设计的重用技术是利用标准函数库，但是标准函数库缺乏必要的“柔性”，不能适应不同场合的不同需要，库函数往往仅提高最基本的、最常用的功能，在开发一个新的软件系统时，通常大部分函数仍需要程序员自己编写，甚至绝大部分函数是新编的。

面向对象程序设计能比较好地解决软件重用的问题。对象所固有的封装性和信息隐蔽等机制，使得对象内部的实现与外界隔离，具有较强的独立性，可以作为一个大粒度的程序构件，供同类程序直接使用。

有论证方法可以重复使用一个对象类：一种方法是建立在各种环境下都能使用的类库对象集，供相关程序直接使用；另一种方法是从其派生出一个满足当前需要的新类。继承性机制使得子类可以重用其父类的数据和程序代码，而且可以在父类代码的基础上方便地修改和扩充，这种修改并不影响对原有类的使用。由于可以像使用集成电路（IC）构建计算机硬件那样，比较方便地重用对象类来构造软件系统，因此有人把对象称为“软件 IC”。

2. 可控制程序的复杂性

传统的程序设计方法忽略了数据和操作之间的内在联系，把数据与其操作分离，于是存在使用错误的数据调用正确的程序模块，或使用正确的数据调用错误程序模块的危险。使数据和操作保持一致，控制程序的复杂性，是程序员的一个沉重的负担。面向对象程序设计采用了数据抽象和信息隐蔽技术，把数据及对数据的操作放在一个类中，作为相互依存、不可

分割的整体来处理。这样，在程序中任何要访问这些数据的地方都只需简单地通过传递消息和调用方法来进行，这就有效控制了程序的复杂性。

3. 可改善程序的可维护性

用传统程序设计语言开发出来的软件很难维护，是长期困扰人们的一个严重问题，是软件危机的突出表现。但面向对象程序设计方法所开发的软件可维护性较好。在面向对象程序设计中，对对象的操作只能通过消息的传递来实现，所以只要消息模式即对应的方法界面不变，方法体的任何修改都不会导致发送消息的程序修改，这显然对程序的维护带来了方便。另外，类的封装和信息隐蔽机制使得外界对其中的数据和程序代码的非法操作作为不可能，这也就大大地减少了程序的错误率。

4. 能够更好地支持大型程序设计

在开发一个大型系统时，应对任务进行清晰的、严格的划分，使每个程序员了解自己要做的工作以及与他人的接口，使每个程序员可以独立设计、调试自己负责的模块，以使各个模块能够顺利地应用到整个系统中去。

类是一种抽象的数据类型，所以类可以作为一个程序模块。要比通常的子程序的独立性强得多，面向对象技术在数据抽象和抽象数据类型之上又引入了动态连接和继承性等机制，进一步发展了基于数据抽象的模块设计，使其更好地支持大型程序设计。

5. 增强了计算机信息处理的范围

面向对象程序设计方法模拟人习惯的解题方法，代表了计算机程序设计的新颖的思维方法。这种方法把描述事物静态属性的数据结构和表示事物动态行为的操作放在一起构成一个整体，完整地、自然地表示客观世界中的实体。

用类来直接描述现实世界中的类型，可使计算机系统的描述和处理对象从数据扩展到现实世界和思维世界的各种事物，这实际上大大扩展了计算机系统处理信息量和信息类型。

6. 能很好地适应新的硬件环境

面向对象程序设计中的对象、消息传递思想和机制，与分布式、并行处理、多机系统及网络等硬件环境也恰好相吻合。面向对象程序设计能够开发出适应这些新环境的软件系统。面向对象的思想也影响到计算机硬件的体系结构，现在已在研究直接支持对象概念的某些对象计算机。这样的计算机将会更适合于面向对象程序设计，更充分地发挥面向对象技术的威力。

由于面向对象程序设计的上述优点，可以看到，面向对象程序设计是目前解决软件开发面临的难题最有希望、最有前途的方法之一。

1.2　Java 概述

1.2.1　Java 的起源与发展

1991 年，Sun 公司的 Jame Gosling、Bill Joe 等人，为了解决家用消费类电子产品智能化过程中的控制和通信问题，设计出了一个新的软件。这个软件是 Jame 等人在充分研究了传统计算机语言，尤其是 C/C++语言的特点后，设计出的一种适合开发跨平台嵌入式软件的语言，当时命名为 Oak，这就是 Java 的前身。但由于智能家用电器并未像预期的那样快速发展，同时 Sun 公司参与投标的一个交互式电视系统的商业项目也未获成功，Oak 面临夭折。但此时，国际互联网的应用却高速发展，Sun 公司看到了 Oak 的跨平台特性在计算机网络应用方面将大有可为，于是对 Oak 系统进行了结构和功能上的改造并加以扩充，将 Oak 重新命名为 Java。

1995 年 5 月，Sun 公司正式发布了 Java 不仅是一种编程语言，而且包含了运行环境，这就克服了传统计算机语言的操作系统依赖性强的弱点。同时，Java 提供了强大的图形、图像、