

<<疯狂Java讲义>>

图书基本信息

书名：<<疯狂Java讲义>>

13位ISBN编号：9787121066467

10位ISBN编号：7121066467

出版时间：2008-10

出版时间：电子工业

作者：李刚

页数：889

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<疯狂Java讲义>>

前言

Java语言赢得了前所未有的成功：从2001年到2007年，Java语言一直是世界上应用最广泛的编程语言，因此现在有越来越多的工作者、学习者正努力进入Java领域或将要努力进入Java领域。

为了帮助广大工作者、学习者能真正掌握Java编程，感受到Java语言的魅力，领会到Java编程的快乐，笔者根据近3年来的Java授课经验，精心编写了本书。

当笔者开始写作本书之前，已经接触了非常多刚刚毕业的大学生，他们告诉笔者：之所以选择编程，仅仅是想找一份工作。

笔者问他们，不喜欢编程吗？

他们回答：编程有什么乐趣？

还不就是写 $1 + 2 + 3 + \dots + 100 = ?$这样的程序。

于是笔者知道了：他们误解了程序。

于是笔者告诉他们：如果我来上C语言，至少讲到可以编写出一个简单的游戏外挂，如果再深入一点，可以结合操作系统理论，编写一个操作系统的雏形——这才叫写程序。

程序的作用是：解决问题——如果你的程序不能解决你自己的问题，如何期望你的程序去解决别人的问题呢？

那你的程序的价值何在？

所以笔者认为：最好的学习方法就是“案例驱动”——知道为什么要用这个知识点，才去学这个知识点，而不是盲目学习！

因此本书强调编程实战，强调以项目激发编程兴趣。

<<疯狂Java讲义>>

内容概要

从2000年至今, Java语言一直是应用最广的开发语言, 并拥有最广泛的开发人群。如今, Java已经不再简单地是一门语言, 它更像一个完整的体系, 一个系统的开发平台。更甚至, 它被延伸成一种开源精神。

本书深入介绍了Java编程的相关方面, 全书内容覆盖了Java的基本语法结构、Java的面向对象特征、Java集合框架体系、Java泛型、异常处理、Java GUI编程、JDBC数据库编程、Java注释、Java的IO流体系、Java多线程编程、Java网络通信编程和Java反射机制。

共覆盖了java.awt、java.lang、java.io和java.nio、java.sql、java.text、java.util、javax.swing包下绝大部分类和接口。

本书并不单纯从知识角度来讲解Java, 而是从解决问题的角度来介绍Java语言, 所以本书中介绍了大量实用案例开发: 五子棋游戏、梭哈游戏、仿QQ的游戏大厅、MySQL企业管理器、仿EditPlus的文本编辑器、多线程、断点下载工具、Spring框架的IoC容器.....这些案例既能让读者巩固每章的知识, 又可以让读者学以致用、激发编程自豪感, 进而引爆内心的编程激情。

本书光盘里包含书中所有示例的代码, 如果读者希望获得课后练习的程序代码。

本书为所有打算深入掌握Java编程的读者而编写, 适合各种层次的Java学习者和工作者阅读, 也适合作为大学教育、培训机构的Java教材。

但如果只是想简单涉猎Java, 则本书过于庞大, 不适合阅读。

<<疯狂Java讲义>>

作者简介

李刚从事8年的Java EE应用开发。

曾任LITEON公司的J2EE技术主管，负责该公司的企业信息平台的架构设计。

曾任广州电信、广东龙泉科技等公司的技术培训导师。

2007年3月26日的《电脑报》专访人物。

现任新东方IT培训广州中心软件教学总监，曾兼任广东技术

<<疯狂Java讲义>>

书籍目录

- 第1章 Java概述 1.1 Java语言的发展简史 1.2 Java的竞争对手及各自优势 1.2.1 C#简介和优势
1.2.2 Ruby简介和优势 1.2.3 Python的简介和优势 1.3 Java程序运行机制 1.3.1 高级语言的运
行机制 1.3.2 Java程序的运行机制和JVM 1.4 开发Java的准备 1.4.1 安装JDK 学生提问：不
说JVM是运行Java程序的虚拟机吗？
那JRE和JVM的关系是怎样的呢？
学生提问：为什么不安装公共JRE系统呢？
1.4.2 设置PATH环境变量 学生提问：为什么选择设置用户变量，用户变量和系统变量有什么区
别呢？
1.5 第一个Java程序 1.5.1 编辑Java源代码 1.5.2 编译Java程序 学生提问：当我们使用编译
程序时，不仅需要指定存放目标文件的位置，也需要指定目标文件的文件名，这里使用javac编译Java
程序时怎么不需要指定目标文件的文件名呢？
1.5.3 运行Java程序 1.5.4 根据CLASSPATH环境变量定位类 1.6 Java程序的基本规则 1.6.1
Java程序的组织形式 1.6.2 Java源文件的命名规则 1.6.3 初学者容易犯的错误 1.7 垃圾回收机
制 1.8 何时开始使用IDE工具 学生提问：老师，我想学习Java编程，到底是学习Eclipse好呢，还是
学习JBuilder好呢？
1.9 本章小结 本章练习 第2章 理解面向对象 2.1 面向对象 2.1.1 结构化程序设计简介
程序的三种基本结构 2.1.3 面向对象程序设计简介 2.1.4 面向对象的基本特征 2.2 UML（统一
建模语言）介绍 2.2.1 用例图 2.2.2 类图 2.2.3 组件图 2.2.4 部署图 2.2.5 顺序图
2.2.6 活动图 2.2.7 状态机图 2.3 Java的面向对象特征 2.3.1 一切都是对象 2.3.2 类和对象
2.4 本章小结 第3章 数据类型和运算符 3.1 注释 3.1.1 单行注释和多行注释 3.1.2 文档注释
学生提问：API文档是什么？
学生提问：为什么要掌握查看API文档的方法？
3.2 标识符和关键字 3.2.1 分隔符 3.2.2 标识符规则 3.2.3 Java关键字 3.3 数据类型分
3.4 基本数据类型 3.4.1 整型 3.4.2 字符型 3.4.3 浮点型 3.4.4 布尔型 3.5 基本类型的
类型转换 3.5.1 自动类型转换 3.5.2 强制类型转化 3.5.3 表达式类型的自动提升 3.6 直接
3.6.1 直接量的类型 3.6.2 直接量的赋值 3.7 运算符 3.7.1 算术运算符 3.7.2 赋值运算
符 3.7.3 位运算符 3.7.4 扩展后的赋值运算符 3.7.5 比较运算符 学生提问：Java为什么要
对这些数据进行缓存呢？
3.7.6 逻辑运算符 3.7.7 三目运算符 3.7.8 运算符的结合性和优先级 3.8 本章小结 本
习 第4章 流程控制和数组 4.1 顺序结构 4.2 分支结构 4.2.1 if条件语句 4.2.2 switch分支语
4.3 循环结构 4.3.1 while循环语句 4.3.2 do while循环语句 4.3.3 for循环 4.3.4 嵌套循
4.4 控制循环结构 4.4.1 使用break结束循环 4.4.2 使用continue结束本次循环 4.4.3 使
用return结束方法 4.5 数组类型 4.5.1 理解数组：数组也是一种类型 学生提问：int[]是一种类型
吗？
怎么使用这种类型呢？
4.5.2 定义数组 4.5.3 数组的初始化 学生提问：能不能只分配内存空间，不赋初始值呢？
4.5.4 使用数组 学生提问：为什么要我记住这些异常信息？
4.5.5 JDK1.5提供了foreach循环 4.6 深入数组 4.6.1 内存中的数组 学生提问：为什么有栈
存和堆内存之分？
4.6.2 基本类型数组的初始化 4.6.3 引用类型数组的初始化 4.6.4 没有多维数组 学生提
：我是否可以让图4.13中灰色覆盖的数组元素再次指向另一个数组？
这样不可以扩展成三维数组吗？
甚至扩展到更多维的数组？
4.6.5 操作数组的工具类 4.6.6 数组的应用举例 4.7 本章小结 本章练习 第5章 面向对象
) 5.1 类和对象 5.1.1 定义类 学生提问：构造器不是没有返回值吗？

<<疯狂Java讲义>>

为什么不能用void修饰呢？

5.1.2 对象的产生和使用

5.1.3 对象、引用和指针

5.1.4 对象的this引用

5.2 方法详解

5.2.1 方法的所属性

5.2.2 方法的参数传递机制

5.2.3 形参长度可变的方法

5.2.4 递归方法

5.2.5 方法重载

学生提问：为什么方法的返回值类型不能用于区分重载的方法？

5.3 成员变量和局部变量

5.3.1 成员变量和局部变量

5.3.2 成员变量的初始化和内存中的运行

机制 5.3.3 局部变量的初始化和内存中的运行机制

5.3.4 变量的使用规则

5.4 隐藏和封装

5.4.1 理解封装

5.4.2 使用访问控制符

5.4.3 package和import

5.4.4 Java的常用包

5.5 深

构造器

5.5.1 使用构造器执行初始化

学生提问：构造器是创建Java对象的途径，是不是说构造器

完全负责创建Java对象？

5.5.2 构造器的重载

学生提问：为什么要用this来调用另一个重载的构造器？

我把另一个构造器里的代码复制、粘贴到这个构造器里不就可以了吗？

5.6 类的继承

5.6.1 继承的特点

5.6.2 重写父类的方法

5.6.3 父类实例的super引用

学

问：我们只是创建了一个Ostrich对象时，哪来的Bird对象？

5.6.4 调用父类构造器

学生提问：为什么我创建Java对象时从未感觉到java.lang.Object的构造器被

调用过？

5.7 多态

5.7.1 多态性

5.7.2 引用变量的强制类型转换

5.7.3 instanceof运算符

5.8 继

组合

5.8.1 使用继承的注意点

5.8.2 利用组合实现复用

学生提问：使用组合关系来实现复用

时，需要创建两个Animal对象，是不是意味着使用组合关系时系统开销更大？

5.9 初始化块

5.9.1 使用初始化块

5.9.2 初始化块和构造器

5.9.3 静态初始化块

5.10 本

章小结

本章练习

第6章 面向对象（下）

6.1 基本数据类型的包装类

6.2 处理对象

6.2.1 打

象和toString方法

6.2.2 ==和equals比较运算符

6.3 类成员

6.3.1 理解类成员

6.3.2 单例

（Singleton）类

6.4 final修饰符

6.4.1 final变量

6.4.2 final方法

6.4.3 final类

6.4.4 不

类

6.4.5 缓存实例的不可变类

6.5 抽象类

6.5.1 抽象方法和抽象类

6.5.2 抽象类的作用

6.6 更彻底的抽象：接口

6.6.1 接口的概念

6.6.2 接口的定义

6.6.3 接口的继承

6.6.4

使用接口

6.6.5 接口和抽象类

6.6.6 面向接口编程

6.7 内部类

6.7.1 非静态内部类

学

问：非静态内部类对象和外部类对象的关系是怎样的？

6.7.2 静态内部类

学生提问：为什么静态内部类实例方法也不能访问外部类的实例属性呢？

学生提问：接口里是否能定义内部接口？

6.7.3 使用内部类

学生提问：既然内部类是外部类的成员，是否可以为外部类定义子类，在子类

中再定义一个内部类来重写其父类中的内部类？

6.7.4 局部内部类

6.7.5 匿名内部类

6.7.6 闭包（Closure）和回调

6.8 枚举类

6.8.

动实现枚举类

6.8.2 枚举类入门

6.8.3 枚举类的属性、方法和构造器

6.8.4 实现接口的枚举

类

6.8.5 包含抽象方法的枚举类

6.9 对象与垃圾回收

6.9.1 对象在内存中的状态

6.9.2 强

垃圾回收

6.9.3 finalize方法

6.9.4 对象的软、弱和虚引用

6.10 修饰符的适用范围

6.11 使

用JAR文件

6.11.1 jar命令详解

6.11.2 创建可执行的JAR包

6.11.3 关于JAR包的技巧

6.12

章小结

本章练习

第7章 Java集合

7.1 Java集合概述

7.2 Collection和Iterator接口

7.2.1 使用Itera

接口遍历集合元素

7.2.2 使用foreach循环遍历集合元素

7.3 Set接口

7.3.1 HashSet类

学生提

：hashCode方法对于HashSet的作用是什么？

7.3.2 TreeSet类

7.3.3 EnumSet类

7.4 List接口

7.4.1 List接口和ListIterator接口

7.4.2

ArrayList和Vector实现类

7.4.3 固定长度的List

7.5 Queue接口

7.5.1 LinkedList实现类

7.5.

PriorityQueue实现类

7.6 Map

7.6.1 HashMap和Hashtable实现类

7.6.2 SortedMap接口和TreeMap

实现类

7.6.3 WeakHashMap实现类

7.6.4 IdentityHashMap实现类

7.6.5 EnumMap实现类

HashSet和HashMap的性能选项

7.8 操作集合的工具类：Collections

7.8.1 排序操作

7.8.2 查找

，替换操作

7.8.3 同步控制

7.8.4 设置不可变集合

7.9 烦琐的接口：Enumeration

7.10 本章

结

本章练习

第8章 泛型

8.1 泛型入门

8.1.1 编译时不检查类型的异常

8.1.2 手动实现编

检查类型

8.1.3 使用泛型

8.2 深入泛型

8.2.1 定义泛型接口、类

8.2.2 从泛型类派生子类

8.2.3 并不存在泛型类

8.3 类型通配符

8.3.1 使用类型通配符

8.3.2 设定类型通配符的上限

<<疯狂Java讲义>>

8.3.3 设定类型形参的上限 8.4 泛型方法 8.4.1 定义泛型方法 8.4.2 泛型方法和类型通配符的区别 8.4.3 设定通配符的下限 8.4.4 泛型方法与重载 8.5 擦除和转换 8.6 泛型与数组

8.7 本章小结 第9章 与运行环境交互 9.1 与用户互动 9.1.1 运行Java程序的参数 9.1.2 使用Scanner获取键盘输入 9.1.3 使用BufferedReader获取键盘输入 9.2 系统相关 9.2.1 System类 9.2.2 Runtime类 9.3 常用类 9.3.1 Object类 9.3.2 String、StringBuffer和StringBuilder类 9.3.3 Math类 9.3.4 Random类 9.3.5 BigDecimal类 9.4 处理日期的类 9.4.1 Date类 9.4.2 Calendar类 9.4.3 TimeZone类 9.5 正则表达式 9.5.1 创建正则表达式 9.5.2 使用正则表达式 9.6 程序国际化 9.6.1 Java国际化的思路 9.6.2 Java支持的语言和国家 9.6.3 完成程序国际化 9.6.4 使用MessageFormat处理包含占位符的字符串 9.6.5 使用类文件代替资源文件 9.6.6 使用NumberFormat格式化数字 9.6.7 使用DateFormat格式化日期 9.7 本章小结 本章练习 第10章 异常处理 10.1 异常概述 10.2 异常处理机制 10.2.1 使用try...catch捕获异常 10.2.2 异常类的继承体系 10.2.3 访问异常信息 10.2.4 使用finally回收资源 10.2.5 异常处理的嵌套 10.3 Checked异常和Runtime异常体系 10.3.1 使用throws声明抛出异常 10.4 使用throw抛出异常 10.4.1 抛出异常 10.4.2 自定义异常类 10.4.3 catch和throw同时使用 10.4.4 异常链 10.5 Java的异常跟踪栈 10.6 异常处理规则 10.6.1 不要过度使用异常 10.6.2 不要使用过于庞大的try块 10.6.3 避免使用Catch All语句 10.6.4 不要忽略捕获到的异常 10.7 本章小结 本章练习

第11章 AWT编程 11.1 GUI (图形用户界面) 和AWT 11.2 AWT容器 11.3 布局管理器 11.3.1 FlowLayout布局管理器 11.3.2 BorderLayout布局管理器 学生提问: BorderLayout最多只能放置5个组件吗? 那它还有什么作用? 11.3.3 GridLayout布局管理器 11.3.4 GridBagLayout布局管理器 11.3.5 CardLayout布局管理器 11.3.6 绝对定位 11.3.7 BoxLayout布局管理器 学生提问: 图11.15和图11.16显示的所有按钮都紧挨在一起, 如果希望像FlowLayout、GridLayout等布局管理器指定组件的间距该怎么办? 11.4 AWT 常用组件 11.4.1 基本组件 11.4.2 对话框 11.5 事件处理 11.5.1 Java事件模型 11.5.2 事件和事件监听器 11.5.3 事件适配器 11.5.4 事件监听器的实现形式 11.6 AWT的菜单 11.6.1 菜单条、菜单和菜单项 11.6.2 右键菜单 学生提问: 为什么即使我没有给多行文本域编写右键菜单, 但当我在多行文本域上单击右键时一样会弹出右键菜单? 11.7 在AWT中绘图 11.7.1 画图的实现原理 11.7.2 使用Graphics类 11.8 处理位图 11.8.1 Image抽象类和BufferedImage实现类 11.8.2 使用ImageIO输入/输出位图 11.9 剪贴板 11.9.1 数据传递的类和接口 11.9.2 传递文本 11.9.3 使用系统剪贴板传递图像 11.9.4 使用本地剪贴板来传递对象引用 11.9.5 通过系统剪贴板传递Java对象 11.10 本章小结 本章练习 第12章 Swing编程 12.1 Swing概述 12.2 Swing基本组件的用法 12.2.1 Swing组件层次 12.2.2 AWT组件的Swing实现 学生提问: 为什么单击Swing多行文本域时不是弹出像AWT多行文本域中的右键菜单? 12.2.3 使用JToolBar创建工具条 12.2.4 使用JColorChooser和JFileChooser 12.2.5 使用JOptionPane 12.3 Swing中的特殊容器 12.3.1 使用JSplitPane 12.3.2 使用JTabbedPane 12.3.3 使用JLayeredPane、JDesktopPane和JInternalFrame 12.4 Swing拖放功能 12.4.1 拖放目标 12.4.2 拖放源 12.4.3 简化的拖放操作 12.5 Applet和JApplet 12.5.1 Applet简介及其安全性 12.5.2 开发Applet类 12.5.3 使用HTML页面装载Applet 12.5.4 appletviewer简介 12.5.5 Applet的生命周期和动画机制 学生提问: 程序中重写paint方法时绘制了一个字符串, 但图12.28上则绘制出了如此之多的字符串, 这是为什么呢? 12.5.6 使用Applet创建音乐播放器 12.6 使用JProgressBar、ProgressMonitor和BoundedRangeModel创建进度条 12.6.1 创建进度条 12.6.2 创建进度对话框 12.7 使用JSlider和BoundedRangeModel创建滑动条 12.8 使用JSpinner和SpinnerModel创建微调控制器 12.9 使用JList、JComboBox创建列表框 12.9.1 简单列表框 12.9.2 不强制存储列表项的ListModel和ComboBoxModel 12.9.3 强制存储列表项的DefaultListModel和DefaultComboBoxModel 学生提问: 为什么JComboBox提供了添加、删除列表项目的方法? 而JList没有提供添加、删除列表项目的方法呢?

<<疯狂Java讲义>>

12.9.4 使用ListCellRenderer改变列表项外观 12.10 使用JTree和TreeModel创建树 12.10.1 创建树 12.10.2 拖动、编辑树节点 12.10.3 监听节点事件 12.10.4 使用TreeCellRendering改变节点外观 12.11 使用JTable和TableModel创建表格 12.11.1 创建JTable 学生提问：我们指定的表格数据、表格列标题都是Object类型的数组，JTable如何显示这些Object对象？ 12.11.2 TableModel和监听器 12.11.3 TableColumnModel和监听器 12.11.4 实现排序 12.11.5 绘制单元格内容 12.11.6 编辑单元格内容 12.12 使用JFormattedTextField和JTextPane创建格式文本 12.12.1 监听Document的改变 12.12.2 使用JPasswordField 12.12.3 使用JFormattedTextField 12.12.4 使用JEditorPane 12.12.5 使用JTextPane 12.13 本章小结 本章练习 第13章 JDBC编程 13.1 JDBC基础 13.1.1 JDBC简介 13.1.2 JDBC驱动程序类型 13.2 SQL语句基础介绍 13.2.1 安装数据库 13.2.2 关系数据库基本概念和MySQL基本命令 13.2.3 SQL语句基础 13.2.4 DDL语句 13.2.5 DML语句语法 13.2.6 select语句和SQL函数 13.2.7 分组和组数 13.2.8 多表连接查询和子查询 13.3 JDBC的典型用法 13.3.1 JDBC常用接口和类简介 13.3.2 JDBC编程步骤 学生提问：前面给出的仅是MySQL和Oracle两种数据库的驱动，我看不出驱动类字符串有什么规律啊。 如果我希望使用其他数据库，我怎么用其他数据库的驱动类呢？ 13.4 执行SQL语句的方式 13.4.1 使用executeUpdate执行DDL和DML语句 13.4.2 使用execute方法执行SQL语句 13.4.3 使用PreparedStatement执行SQL语句 13.4.4 使用CallableStatement调用存储过程 13.5 管理结果集 13.5.1 可滚动、可更新的结果集 13.5.2 处理Blob类型数据 13.5.3 用ResultSetMetaData分析结果集 13.6 事务处理 13.6.1 事务的概念和MySQL事务支持 13.6.2 JDBC的事务支持 13.6.3 批量更新 13.7 分析数据库信息 13.7.1 使用DatabaseMetaData分析数据库信息 13.7.2 使用系统表分析数据库信息 13.7.3 选择合适的分析方式 13.8 使用连接池管理连接 13.9 本章小结 本章练习 第14章 Annotation（注释） 14.1 基本Annotation 14.1.1 限定父类方法：@Override 14.1.2 标示已过时：@Deprecated 676 14.1.3 抑制编译器警告：@SuppressWarnings 14.2 自定义Annotation 14.2.1 定义Annotation 14.2.2 提取Annotation的信息 14.2.3 使用Annotation的例子 14.3 JDK的元Annotation 14.3.1 使用@Retention 14.3.2 使用@Target 14.3.3 使用@Documented 14.3.4 使用@Inherited 14.4 使用APT处理Annotation 14.4.1 本章小结 第15章 输入/输出 15.1 File类 15.1.1 访问文件和目录 15.1.2 文件过滤器 15.2 理解Java的IO流 15.2.1 流的分类 15.2.2 流的概念模型 15.3 字节流和字符流 15.3.1 InputStream和Reader 15.3.2 OutputStream和Writer 15.4 输入/输出流体系 15.4.1 处理流的用法 15.4.2 输入/输出流体系 15.4.3 转换流 学生提问：怎么没有把字符流转换成字节流的转换流呢？ 15.4.4 推回输入流 15.5 重定向标准输入/输出 15.6 Java虚拟机读写其他进程的数据 15.7 RandomAccessFile 15.8 对象序列化 15.8.1 序列化的含义和意义 15.8.2 使用对象流实现序列化 15.8.3 对象引用的序列化 15.8.4 自定义序列化 15.8.5 另一种自定义序列化机制 15.8.6 版本 15.9 Java新IO 15.9.1 Java新IO概述 15.9.2 使用Buffer 15.9.3 使用Channel 15.9.4 代码集和Charset 15.9.5 文件锁 15.10 本章小结 本章练习 第16章 多线程 16.1 线程概述 线程和进程 16.1.2 多线程的优势 16.2 线程的创建和启动 16.2.1 继承Thread类创建线程类 16.2.2 实现Runnable接口创建线程类 16.2.3 两种方式所创建线程的对比 16.3 线程的生命周期 16.3.1 新建和就绪状态 16.3.2 运行和阻塞状态 16.3.3 线程死亡 16.4 控制线程 16.4.1 join线程 16.4.2 后台线程 16.4.3 线程睡眠：sleep 16.4.4 线程让步：yield 16.4.5 改变线程优先级 16.5 线程的同步 16.5.1 线程安全问题 16.5.2 同步代码块 16.5.3 同步方法 16.5.4 释放同步监视器的锁定 16.5.5 同步锁（Lock） 16.5.6 死锁 16.6 线程通信 16.6.1 线程的协调运行 16.6.2 使用条件变量控制协调 16.6.3 使用管道流 16.7 线程组和未处理的异常 16.8 Callable和Future 16.9 线程池 16.10 线程相关类 16.10.1 ThreadLocal类 16.10.2 线程不安全的集合 16.10.3 线程安全的集合类 16.11 本章小结 第17章 网络编程 17.1 网络编程的基础知识 17.1.1 网络基础知识 17.1.2 IP地址和端口号 17.2 Java的基本网络支持 17.2.1 使用InetAddress 17.2.2 使用URLDecoder和URLEncoder 17.2.3 使用URL和URLConnection 17.3

<<疯狂Java讲义>>

基于TCP协议的网络编程	17.3.1 TCP协议基础	17.3.2 使用ServletSocket创建TCP服务器端
17.3.3 使用Socket进行通信	17.3.4 加入多线程	17.3.5 记录用户信息
阻塞Socket通信	17.4 基于UDP协议的网络编程	17.3.6 使用NIO实现非阻塞Socket通信
17.4 基于UDP协议的网络编程	17.4.1 UDP协议基础	17.4.2 使用DatagramSocket发送、接收数据
17.4.1 UDP协议基础	17.4.3 使用MulticastSocket实现多点广播	17.5 使用代理服务器
17.4.2 使用DatagramSocket发送、接收数据	17.5.1 直接使用Proxy创建连接	17.6 本章小结
17.4.3 使用MulticastSocket实现多点广播	17.5.2 使用ProxySelector选择代理服务器	本章练习
17.5.1 直接使用Proxy创建连接	18.1 类的加载、连接和初始化	第18章 类加载和反射
17.5.2 使用ProxySelector选择代理服务器	18.1.1 JVM和类	18.1.2 类的加载
18.1 类的加载、连接和初始化	18.1.3 类的连接	18.1.4 类的初始化
18.1.1 JVM和类	18.1.4 类的初始化	18.1.5 类初始化的时机
18.1.2 类的加载	18.2 类加载器	18.2.1 类加载器简介
18.1.3 类的连接	18.2.1 类加载器简介	18.2.2 URLClassLoader类
18.1.4 类的初始化	18.2.2 URLClassLoader类	18.2.3 创建并使用自定义的类加载器
18.1.5 类初始化的时机	18.2.3 创建并使用自定义的类加载器	18.2.4 通过反射查看类信息
18.2 类加载器	18.2.4 通过反射查看类信息	18.3.1 获得Class对象
18.2.1 类加载器简介	18.3.1 获得Class对象	18.3.2 从Class中获取信息
18.2.2 URLClassLoader类	18.3.2 从Class中获取信息	18.4 使用反射生成并操作对象
18.2.3 创建并使用自定义的类加载器	18.4 使用反射生成并操作对象	18.4.1 创建对象
18.2.4 通过反射查看类信息	18.4.1 创建对象	18.4.2 调用方法
18.3.1 获得Class对象	18.4.2 调用方法	18.4.3 访问属性值
18.3.2 从Class中获取信息	18.4.3 访问属性值	18.4.4 操作数组
18.4 使用反射生成并操作对象	18.4.4 操作数组	18.5 使用反射生成JDK动态代理和AOP
18.4.1 创建对象	18.5 使用反射生成JDK动态代理和AOP	18.5.1 使用Proxy和InvocationHandler创建动态代理
18.4.2 调用方法	18.5.1 使用Proxy和InvocationHandler创建动态代理	18.5.2 动态代理和AOP
18.4.3 访问属性值	18.5.2 动态代理和AOP	18.6 反射和泛型
18.4.4 操作数组	18.6 反射和泛型	18.6.1 泛型和Class类
18.5 使用反射生成JDK动态代理和AOP	18.6.1 泛型和Class类	18.6.2 使用反射来获取泛型信息
18.5.1 使用Proxy和InvocationHandler创建动态代理	18.6.2 使用反射来获取泛型信息	18.7 本章小结
18.5.2 动态代理和AOP	18.7 本章小结	本章练习
18.6 反射和泛型	本章练习	上善若水——兼谈我的创作理念
18.6.1 泛型和Class类	上善若水——兼谈我的创作理念	
18.6.2 使用反射来获取泛型信息		

章节摘录

Java语言历时12年，已发展成为人类计算机史上影响深远的编程语言，从某种程度上来看，它甚至超出了编程语言的范畴，成为一种开发平台，一种开发规范。

更甚至于：Java已成为一种信仰，Java语言所崇尚的开源、自由等精神，吸引了全世界无数优秀的程序员。

事实是，从人类有史以来，从来没有一门编程语言能吸引这么多的程序员，也没有一门编程语言能衍生出如此之多的开源框架。

Java语言是一门非常纯粹的面向对象编程语言，它吸收了C++语言的各种优点，又摒弃了C++里难以理解的多继承、指针等概念，因此Java语言具有功能强大和简单易用两个特征。

Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程开发。

不仅如此，Java语言相关的Java EE规范里包含了时下最流行的各种软件工程理念，各种先进的设计思想总能在Java EE规范、平台以及相关框架里找到相应实现。

从某种程度上来看，学精了Java语言的相关方面，相当于系统地学习了软件开发相关知识，而不是仅仅学完了一门编程语言。

时至今日，大部分银行、电信、证券、电子商务、电子政务等系统或者已经采用Java EE平台构建，或者正在逐渐过渡到采用Java EE平台来构建，Java EE规范是目前最成熟的，也是应用最广的企业级应用开发规范。

编辑推荐

疯狂源自梦想，技术成就辉煌 本书来自作者3年的Java培训经历，凝结了作者近3000个小时的授课经验，总结了几百个Java学员学习过程中的典型错误。

1.案例驱动，引爆编程激情 本书不再是知识点的铺陈，而是致力于将知识点融入实际项目的开发，所以书中涉及了大量Java案例：仿QQ的游戏大厅、MySQL企业管理器、仿EditPlus的文本编辑器、多线程、断点下载工具……希望读者通过编写这些程序找到编程的乐趣。

2.再现李刚老师课堂氛围 本书的内容是笔者近3年授课经历的总结，知识体系取自李刚疯狂Java实战课程体系。

本书力求再现笔者的课堂氛围：以浅显比喻代替乏味的讲解，以疯狂实战代替空洞的理论。

3.注释详细，轻松上手 为了降低读者阅读的难度，书中代码的注释非常详细，几乎每两行代码就有一行注释。

不仅如此，本书甚至还把一些简单理论作为注释穿插到代码中，力求让读者能轻松上手。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>