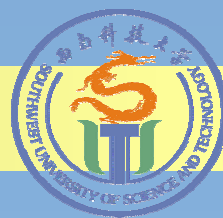


西南科技大学 (SouthWest University of Science and Technology)



## Experimental Instructor of Java Programing

# Java 程序设计 实验指导书

软件教研室 编写

计算机科学与技术学院

二 00 七年

# 前言

## FOREWORD FOREWORD FOREWORD

---

本书是课程《Java 程序设计》的配套书。根据课程教学大纲和 Java 程序设计的特点,从面向对象程序设计的基本应用入手,结合实际应用程序的设计过程,由浅入深地实践 Java 应用程序的开发方法和技巧。通过实验,加深学生对《Java 程序设计》理论课程的理解,进一步理解并具备面向对象观念,掌握面向对象程序设计的方法,提高学生对 Java 语言各部分内容的综合使用能力,逐步掌握 Java 语言程序设计的规律与技巧,培养软件设计能力。

书中共设计了 9 个验证型实验、5 个设计型实验和 2 个综合型实验,开课教师可以根据大纲要求进行选取。在以往的实验指导书指导基础上,细化了实验要求,给出了参考的考核和评分标准。实验内容和设计题目的设计针对学生的实际情况,做到难易适中,验证型实验、设计型实验和综合型实验分别在实验要求上分成不同的层次,力争让学生经过一定的努力,都能够完成相应题目,收获成功的喜悦,从而激发他们学习的兴趣和积极性。此外,书中附录部分介绍了集成开发环境,专门设计了针对本课程的实验报告和设计报告,并对报告各个部分的写法和要求作了详细说明。

本指导书中程序运行环境使用 JDK1.5,集成开发环境使用 NetBeans5.5。本书可供计算机科学与技术、软件工程、信息安全以及计算机学科其他相关专业选用。

《Java 程序设计》实验指导书建设小组

2007 年 5 月

# 目 录

<b>第 1 章 概述</b>	<b>1</b>
1.1 课程简介	1
1.2 实验类型	2
1.3 实验环境选择	3
<b>第 2 章 实验要求</b>	<b>4</b>
2.1 实验过程要求	4
2.2 考核及评分标准	4
<b>第 3 章 实验内容与指导</b>	<b>6</b>
3.1 实验一 JDK 开发工具	6
3.1.1 实验类型	6
3.1.2 实验目的	6
3.1.3 知识点介绍	6
3.1.4 实验内容	6
3.2 实验二 Java 语言基础	12
3.2.1 实验类型	12
3.2.2 实验目的	12
3.2.3 知识点介绍	12
3.2.4 实验内容	14
3.3 实验三 类的封装和继承	18
3.3.1 实验类型	18
3.3.2 实验目的	18
3.3.3 知识点介绍	18
3.3.4 实验内容	18
3.4 实验四 多态与接口	24
3.4.1 实验类型	24
3.4.2 实验目的	24
3.4.3 知识点介绍	24
3.4.4 实验内容	24
3.5 实验五 常用基础类库与工具类库	26
3.5.1 实验类型	26
3.5.2 实验目的	26
3.5.3 知识点介绍	26
3.5.4 实验内容	26
3.6 实验六 异常处理	30
3.6.1 实验类型	30
3.6.2 实验目的	30
3.6.3 知识点介绍	30
3.6.4 实验内容	30
3.7 实验七 多线程机制	32
3.7.1 实验类型	32
3.7.2 实验目的	32

3.7.3 知识点介绍 .....	32
3.7.4 实验内容 .....	33
3.8 实验八 流式输入输出 .....	39
3.8.1 实验类型 .....	39
3.8.2 实验目的 .....	39
3.8.3 知识点介绍 .....	39
3.8.4 实验内容 .....	40
3.9 实验九 Applet 简单应用 .....	44
3.9.1 实验类型 .....	44
3.9.2 实验目的 .....	44
3.9.3 知识点介绍 .....	44
3.9.4 实验内容 .....	45
3.10 设计型实验 .....	48
3.10.1 题目一 算术计算器 .....	48
3.10.2 题目二 日历查看器 .....	48
3.10.3 题目三 简单画图程序 .....	49
3.10.4 题目四 简单文本编辑器 .....	49
3.10.5 题目五 商店购物结算器 .....	50
3.11 综合型实验 .....	51
3.11.1 题目一 学生成绩管理系统的设计 .....	51
3.11.2 题目二 Socket 编程实现网络聊天室 .....	52
附录 A Java 编码规范 .....	54
附录 B NetBeans 集成开发环境 .....	59
附录 C 验证型实验报告格式 .....	65
附录 D 设计型、综合型实验报告格式 .....	67

# 第 1 章 概述

## 1.1 课程简介

Java 语言是目前除 C++ 以外的另一种主流编程语言，体现了面向对象程序设计的诸多特色，具有较强的安全性、健壮性、可移植性以及网络功能。《Java 语言程序设计》是为计算机专业二年级以上本科生开设的专业选修课程。

本实验根据课程教学大纲和 Java 程序设计的特点，从面向对象程序设计的基本应用入手，结合实际应用程序的设计过程，由浅入深地实践 Java 应用程序的开发方法和技巧。实验主要包括 Java 语言的基本语法和基本结构、Java 应用程序和小应用程序（Applet）的用户界面设计、Java 面向对象的程序设计、多线程设计、图形及事件处理以及 Java 在网络编程和数据库方面的应用。通过实验，加深学生对《Java 语言程序设计》理论课程的理解，进一步理解并具备面向对象观念，掌握面向对象程序设计的方法，提高学生对 Java 语言各部分内容的综合使用能力，逐步掌握 Java 语言程序设计的规律与技巧，培养软件设计能力。

本课程的实验体系如表：1-1 所示。

表 1-1 实验体系表

《Java 语言程序设计》实验体系					
实验类型	序号	学时	实验名称	选题要求	备注
验证型实验	1	1	实验一 JDK 开发工具	必做	以配置为主
	2	2	实验二 Java 语言基础	选做	
	3	2	实验三 类的封装和继承	必做	
	4	2	实验四 多态与接口	必做	
	5	2	实验五 常用基础类库与工具类库	选做	
	6	2	实验六 异常处理	选做	
	7	3	实验七 多线程机制	必做	
	8	2	实验八 流式输入输出	必做	
	9	2	实验九 Applet 简单应用	必做	
设计型实验	1	3	题目一 算术计算器	必做其一	
	2	3	题目二 日历查看器		
	3	3	题目三 简单画图程序	选做其一	
	4	3	题目四 简单文本编辑器		
	5	3	题目五 商店购物结算器	选做	
综合型实验	6	4	题目一 学生成绩管理系统	必做	
	7	5	题目二 Socket 编程实现网络聊天室	必做	
1.实验的目的、要求、内容、方法详见第三章试验内容部分。					
2.设计型实验独立完成。					

实验类型情况统计如表 1-2 和表 1-3 所示。

表 1-2 实验体系表

实验类型统计表									
	实验设计（36 学时）			验证型（18 学时）		设计型（18 学时）		设计型（18 学时）	
	学时数	比例	实验个数	学时数	实验个数	学时数	实验个数	学时数	实验个数
必做	24	63%	10	12	6	3	2	9	2
选做	12	37%	6	6	3	6	3	0	0
小计	36	100%	22	18	9	9	5	9	2

表 1-2 实验学时数对比表

	实验设计	验证型	设计型	综合型
学时数	36	18	9	9
比例	100%	50%	25%	25%

## 1.2 实验类型

实验的分类方法很多，按性质分，实验的不同类型包括：验证型实验、设计型实验、综合型实验。本课程涉及到的实验类型主要有：验证型实验、设计型实验和综合型实验。

### 验证型实验

验证型实验作为一种重要的实验形式，作用是任何其他类型的实验所无法替代的，主要培养学生加深对理论的理解。实际上，与课程相关的大部分实验都是验证型实验。实验设计者给出较为详细的实验步骤，旨在减少实验者摸索的过程，争取在较短的时间内掌握基本的编程方法和思想。

验证型实验的方法：

1. 明确实验题目、实验目的和实验要求；
2. 熟悉实验背景知识；
3. 按照实验内容进行实验；
4. 分析实验结果书写实验报告。

### 设计型实验

设计型实验培养学生的设计能力。这类实验是课程中较典型的实验，是针对某一类知识点的基本训练的基础之上，提出的有一定实用价值的实验题目。题目的描述，以提出任务、要求和阐述应用背景为宜，而如何解决问题，解决问题的原理、方法等由同学们自行提出并实践。目的是使学生运用所学的理论知识和实验技能，在编程方法的考虑、使用工具的选择、测试方法的确定等方面受到比较有针对性的训练。

设计型实验的方法：

1. 了解题目要求，明确任务；
2. 查阅有关资料，画出必要的符合规范的图表，寻求各种解决问题的方法，从原理、方法和

使用工具等多方面提出完成课题任务的依据及实验步骤；

3. 做实验；

4. 测试结果评价，总结分析并书写设计报告。

### 综合型实验

综合型实验培养学生对课程知识点的综合运用能力。这类实验是课程中较大的实验，有一定应用价值，是一个小的软件系统，有一定难度。题目的描述，以提出任务、要求和阐述应用背景为宜，必要时给予一定的提示，而如何解决问题，解决问题的原理、方法等由同学们自行提出并实践。目的是使学生综合运用所学的理论知识和实验技能，在编程方法的考虑、使用工具的选择、测试方法的确定等方面受到比较全面的训练。

综合型实验的方法：

1. 根据提示和自己的理解，了解题目要求，明确任务；

2. 查阅有关资料，画出必要的符合规范的图表，寻求各种解决问题的方法，从原理、方法和工具等多方面提出完成课题任务的依据及实验步骤；

3. 在动手编程之前，应多做目标系统在功能上和性能上的分析，设计出目标系统的结构；

4. 在分析的基础上，把需要的类识别出来，并做好类的设计

5. 按照编码规范进行程序设计；

6. 测试结果评价，总结分析并书写设计报告。

## 1.3 实验环境选择

本课程实验的环境也就是 Java 的编写、编译、运行的环境，如下所示：

1. JDK 版本：JDK1.5。

2. 开发工具以常见开发工具为主：

✧ 记事本/写字板

✧ NetBeans

✧ Eclipse

✧ JBuilder

✧ JCreator

✧ UltraEdit

建议：在刚开始的时候使用记事本进行程序编写，使用命令行编译和运行程序，让学生充分熟悉整个过程；然后学生可以逐步选择一种适合自己开发集成开发工具。

对 NetBeans 集成开发环境的使用参见附录 B。

## 第 2 章 实验要求

### 2.1 实验过程要求

实验中，实验者必须服从指导教师和实验室工作人员的安排，遵守纪律与实验制度，爱护设备及卫生。在指定的实验时间内，必须到实验室内实验，同时学生应该充分利用课余时间进行队实验内容的分析和程序设计。实验所涉及的 Java 程序代码，都要求有较高的可读性和可重用性，符合面向对象的编程思想和规范。

实验前要充分做好准备工作，建议如下：

1. 预习、思考实验内容；
2. 复习和掌握与本实验有关的知识内容；
3. 准备好上机所需的程序代码，实验课内时间主要是解决问题、调试程序、运行程序、测试程序和分析结果的时间，不能不编写程序或抄袭别人程序去做实验；
4. 对程序中自己有不懂或疑问的地方，应做出记录，以便在实验课上集中解决；
5. 准备好调试和运行所需的数据。

实验时一人一组，独立上机。对于上机过程中出现的问题，尽量先独立思考和解决，尤其是语法错误、编译器提示信息，应善于独立分析判断，这是提高程序调试能力最主要的途径；对于难以解决的问题可以和同学交流或问老师；对于一个实验题目，可以先考虑尽可能多的方法，然后再这些方法里面选择一种较为有效的方法来实现。

作为实验结果之一的实验代码书写要求规范清晰，如缩进、空行、程序块对齐等；同时应该有必要的注释，具体要求参见附录 A。

实验后，应及时整理出实验报告，提交电子及书面文档的实验报告（实验报告的具体格式见附录 A 和附录 B）和实验程序。

综合型实验应有答辩环节，分为自述和教师提问两部分，自述时间不得超过 5 分钟，内容包括：演示、描述本课题设计思想、关键代码分析等；教师提问可以由指导教师灵活掌握，要求能考查学生的真实能力。指导教师可以根据实际情况，选择需要答辩的实验，但至少应该有一个综合型实验的答辩安排。

### 2.2 考核及评分标准

本实验考核根据教务处和学院的有关规定和实验的具体情况进行。实验课程中所有必做实验学时数之和为 24 学时。实验的评分采用结构化评分方式，建议分值为：验证型实验（42 分）+ 设计型实验（12 分）+ 综合型实验（36）+ 其他（10 分）。其他部分，主要由指导教师灵活处理，可以是考勤等，验证型实验和设计型实验主要考核指标如下：

验证型实验：

1. 实验者是否真实、认真的完成了本次实验；
2. 实验代码是否规范、可读性怎样；
3. 提交的代码是否可重现，运行结果是否能达到题目要求；
4. 实验报告是否书写完毕，格式是否规范，是否有抄袭；
5. 建议每次必做实验满分记分 7 分。

设计型、综合型实验：

1. 代码是否调试通过、运行结果是否能达到题目要求，是否具备良好可读性；



2. 设计报告是否层次清楚、整洁规范、有无抄袭、雷同，对 6 学时设计型实验的设计报告要求应较高；
3. 答辩考查学生是否思路清晰、逻辑严谨，考虑问题是否周到，设计方案是否正确，答辩结束后，指导教师根据答辩情况并仔细审查设计报告和相关文档，参照题目的难度，给出本设计的最终成绩；
4. 设计型实验只有一次，建议记分 12 分，综合型实验每次计分 18 分。

有关取消考核资格以及成绩记 0 分的建议：

1. 如果学生在指定的实验地点和时间被检查到三次不到课，则取消考核资格；
2. 如果学生需要请假，必须提前出具正式假条（需要班主任或者辅导员签字），不接受事后假条（如有特殊情况，需要有辅导员或班主任老师的情况说明）；
3. 如果发现学生抄袭、伪造实验数据，或实验报告和设计报告抄袭、雷同，则涉及的所有学生的该课程实验成绩记为 0 分，如有行为特别恶劣者，报送教务处处理。

同一届学生的实验成绩评价结构及考核方式的细则由承担该门实验课程的所有教师在实验开始之前协商统一，并遵照执行，其中实验分值也可根据具体情况作调整。

## 第 3 章 实验内容与指导

### 3.1 实验一 JDK 开发工具

#### 3.1.1 实验类型

验证型实验

#### 3.1.2 实验目的

1. 熟悉安装和配置 JDK 开发环境
2. 熟悉安装和配置 IDE 开发工具
3. 掌握 Java 程序编辑、编译和运行的过程
4. 掌握 Java 程序的构成特点
5. 总结在调试过程中的错误

#### 3.1.3 知识点介绍

Java 语言的出现是源于对独立于平台语言的需要，即这种语言编写的程序不会因为芯片的变化而无法运行或出现运行错误。目前，随着网络迅速发展，Java 语言的优势更加明显，Java 已经成为网络时代最重要的语言之一。

Sun 公司为了实现“编写一次，到处运行”(write once, run anywhere)的目标，提供了相应的 Java 运行平台。目前 Java 运行平台主要分为 3 个版本：

1. J2SE: Java 标准版或 Java 标准平台。J2SE 提供了标准的 SDK 开发平台（以前称做 JDK 开发平台）。利用该平台可以开发 Java 桌面应用程序和低端的服务器应用程序，也可以开发 Java Applet 程序。
2. J2EE: Java 企业版或 Java 企业平台。使用 J2EE 可以构建企业级的服务应用。J2EE 平台包含了 J2SE 平台，并增加了附加类库，以便支持目录管理、交易管理和企业级消息处理等功能。
3. J2ME: Java 微型版或 Java 小型平台。J2ME 是一种很小的 Java 运行环境，用于嵌入式的消费产品中，如移动电话、掌上电脑或其他无线设备等。

无论上述哪种 Java 运行平台，都包括了相应的 Java 虚拟机(Java virtual machine)。虚拟机负责将字节码文件（包括程序使用的类库中的字节码）加载到内存，然后采用解释方式来执行字节码文件，即根据相应硬件的机器指令翻译一句执行一句。J2SE 平台是学习掌握 Java 语言的最佳平台，而掌握 J2SE 又是进一步学习 J2EE 和 J2ME 所需要的准备。

#### 3.1.4 实验内容

##### 题目 1 JDK 的安装与环境变量的配置

安装 JDK，观察安装后的目录体系，并在 WindowsXP 环境下分别配置 JAVA\_HOME、CLASSPATH、PATH 三个环境变量。具体步骤如下所示。

## 1. JDK 下载安装过程。

(1) 进入 JDK1.5 下载网址：[http://java.sun.com/javase/downloads/index\\_jdk5.jsp](http://java.sun.com/javase/downloads/index_jdk5.jsp)。

(2) 选择要下载的项目(此处需要选择 Java Development Kit 5.0 Update 12), 然后点击“Download”按钮, 如图 3-1 所示。

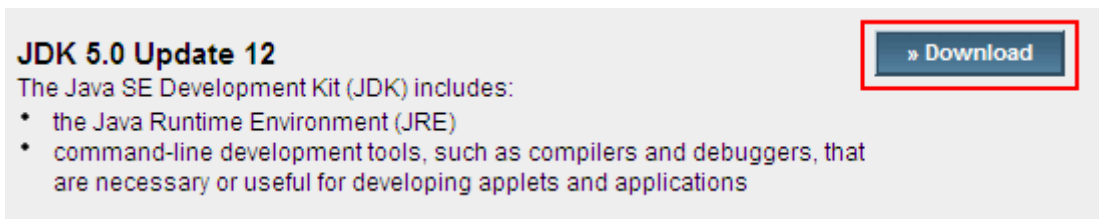


图 3-1 下载前 J2SE SDK 选择示意图

(3) 选择“Accept”, 如图 3-2 所示。这时页面会自动处理, 给出下载的连接。



图 3-2 下载前接受协议示意图

(4) 下载页面会列出各个平台下的 JDK 版本, 其中 Windows 版(分为 32 位和 64 位两种 Windows 系统)有两种安装方式, 一种是完全下载后再安装, 一种是在线安装, 我们选择下载后再安装, 如图 3-3 所示。

Windows Platform - Java Development Kit 5.0 Update 12		
↓ Windows Offline Installation, Multi-language	jdk-1_5_0_12-windows-i586-p.exe	51.33 MB
↓ Windows Online Installation (typical download size is ~33.7MB), Multi-language	jdk-1_5_0_12-windows-i586-p-iftw.exe	245.77 KB
Linux Platform - Java Development Kit 5.0 Update 12		
↓ Linux RPM in self-extracting file	jdk-1_5_0_12-linux-i586-rpm.bin	45.58 MB
↓ Linux self-extracting file	jdk-1_5_0_12-linux-i586.bin	47.32 MB
Solaris SPARC Platform - Java Development Kit 5.0 Update 12		
↓ Solaris SPARC 32-bit self-extracting file	jdk-1_5_0_12-solaris-sparc.sh	51.46 MB
↓ Solaris SPARC 32-bit packages - tar.Z	jdk-1_5_0_12-solaris-sparc.tar.Z	88.36 MB
↓ Solaris SPARC 64-bit self-extracting file (use 32-bit version for applet and Java Web Start support)	jdk-1_5_0_12-solaris-sparcv9.sh	10.02 MB
↓ Solaris SPARC 64-bit packages - tar.Z (use 32-bit version for applet and Java	jdk-1_5_0_12-solaris-sparcv9.tar.Z	12.55 MB

图 3-3 JDK 版本选择及下载示意图

(5) 下载完成后, 双击图标进行安装, 安装过程中可以自定义安装目录等信息, 例如选择安装目录为 C:\jdk15。

## 2. 配置 JDK 环境变量

(1) 右击“我的电脑”, 点击“属性”, 如图 3-4 所示。

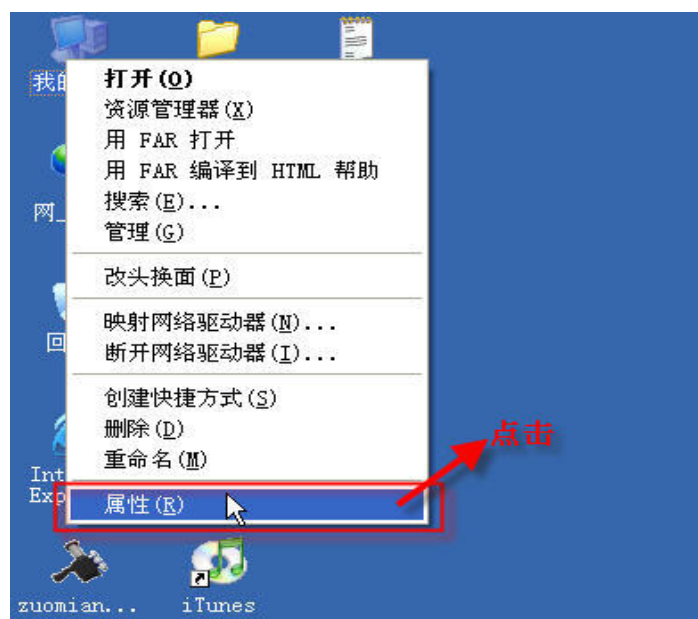


图 3-4 选择属性示意图

(2) 选择“高级”选项卡，点击“环境变量”，如图 3-5 所示。



图 3-5 环境变量选取示意图

(3) 在“系统变量”中，设置 3 个变量：JAVA\_HOME, Path, ClassPath（大小写均可）。若已存在则点击“编辑”，不存在则点击“新建”，如图 3-6 所示。



图 3-6 需要的属性设置示意图

#### (4) 环境变量的设置

JAVA\_HOME 指明 JDK 安装路径，就是刚才安装时所选择的路径 C，此路径下包括 lib，bin，jre 等文件夹，如图 3-7 所示。

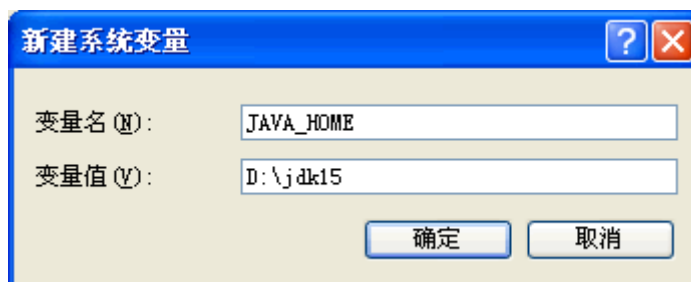


图 3-7 环境变量 JAVA\_HOME 设置示意图

PATH 使得系统可以在任何路径下识别 Java 命令，设为：

%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin (只设为%JAVA\_HOME%\bin 也行)，如图 3-8 所示。

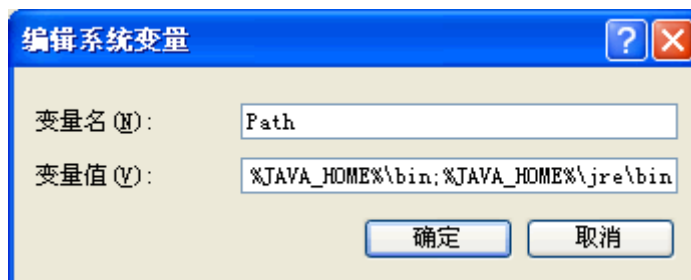


图 3-8 环境变量 PATH 设置示意图

CLASSPATH 为 java 加载类(class or lib)路径，设为：

.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar (一定要加“.”号表示当前路径)，如图 3-9 所示。

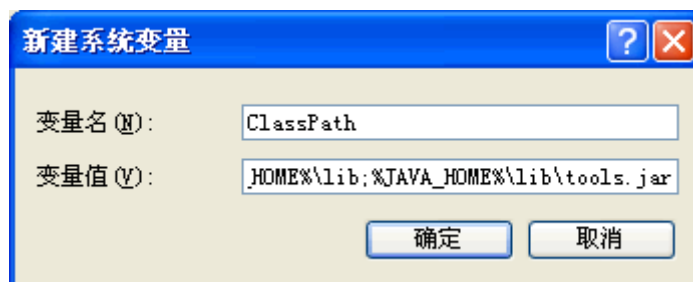


图 3-9 环境变量 CLASS PATH 设置示意图

### 3. JDK 安装与配置的有效性检验

(1) 打开 Dos 窗口：“开始”>“运行”，键入“cmd”，然后点击确定，如图 3-10 所示。



图 3-10 打开 Dos 窗口示意图

(2) 键入命令“java -version”，出现下图画面，说明环境变量配置成功，如图 3-11 所示。

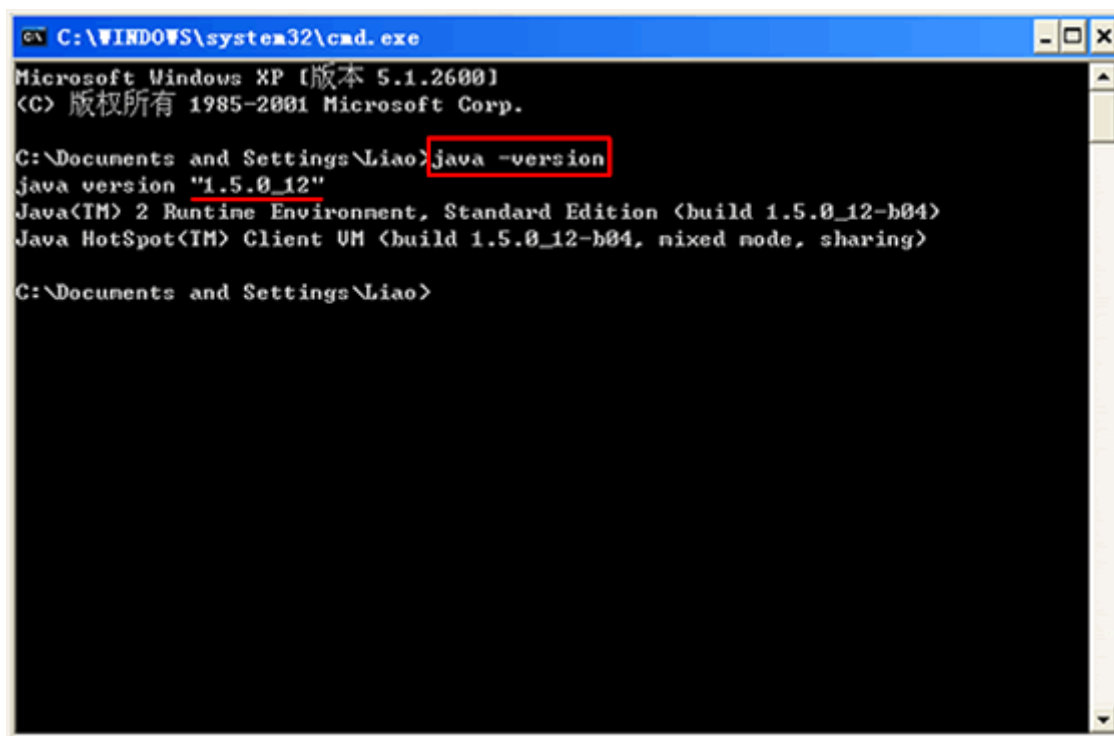


图 3-11 “java -version”命令运行结果窗口

## 题目 2 集成开发环境的安装

获取集成开发环境的安装程序，双击安装程序图标，按照向导的提示，根据实际情况正确选择安装选项。

## 题目 3 编写一个简单的 Java 应用程序

程序在命令窗口输出两行文字：“你好，很高兴学习 Java”和“We are students”。

**[思考与扩展]**

1. 编译器是怎样提示丢失大括号的错误。
2. 编译器是怎样提示语句丢失分号的错误。
3. 编译器是怎样提示将 `System` 写成 `system` 的错误。
4. 编译器是怎样提示将 `String` 写成 `string` 的错误。

## 3.2 实验二 Java 语言基础

### 3.2.1 实验类型

验证型实验

### 3.2.2 实验目的

1. 掌握变量名等标识符的命名方法
2. 掌握数据类型间的互相转换，同时了解 unicode 字符表
3. 掌握 Java 程序几种控制结构：if-else 分支、while 循环

### 3.2.3 知识点介绍

#### 1. 标识符的命名方法

用来标识类名、变量名、方法名、类型名、数组名、文件名的有效字符序列称为标识符。标识符是 Java 语言的基本组成部分，定义的规则如下：

- ✧ 由字母、数字、下划线和美元符组成；
- ✧ 必须以字母、下划线或美元符号打头；
- ✧ 严格区分大小写，没有长度限制；
- ✧ 应该具有特定的意义。

#### 2. 数据类型

Java 的基本数据类型包括 byte、short、int、long、float、double 和 char。要特别掌握基本类型的数据转换规则，基本数据类型按精度级别由低到高的顺序为

byte short int long float double

当把级别低的变量的值赋给级别高的变量时，系统自动完成数据类型的转换。当把级别高的变量的值赋给级别低的变量时，必须使用显示类型转换运算。

要观察一个字符在 unicode 表中的顺序位置，必须使用 int 类型显示转换，如(int)'a'。不可以使用 short 型转换，因为 char 的最高位不是符号位。同样，要得到一个 0~65536 之间的数所代表的 unicode 表中相应位置上的字符，也必须使用 char 型显示转换。char 类型数据和 byte、short、int 或 long 运算的结果总是 int 类型数据。

#### 3. 分支语句

if-else 语句是 Java 中的一条语句，由一个 if、else 和两个复合语句按一定格式构成，if-else 语句的格式如下：

```
if (表达式) {  
    若干语句  
}  
else {  
    若干语句  
}
```

一条 if-else 语句的作用是根据不同的条件产生不同的操作，执行法则是，if 后面 ( ) 内的表达式的值必须是 boolean 型的。如果表达式的值为 true，则执行紧跟着的复合语句；如果表达式的值为 false，则执行 else 后面的复合语句。



if-else if-else 语句称为对条件分支语句，如果需要根据多条件来选择某一操作，这时就可以使用 if-else if-else 语句。if-else if-else 语句是 Java 中的一条语句，由一个 if、若干个 else if、一个 else 与若干个复合语句按一定规则构成。语句的格式如下所示。

```
if (表达式 1) {
    若干语句
}
else if (表达式 2) {
    若干语句
}
else if (表达式 n) {
    若干语句
}
else {
    若干语句
}
```

有时为了编程的需要，复合语句中可以没有任何语句。一条 if-else if-else 语句的作用是根据不同的条件产生不同的操作，执行法则如下：

if 以及 else if 后面 ( ) 内的表达式的值必须是 boolean 型的；该语句执行时，首先计算 if 后括号中的表达式的值，如果该表达式的值为 true，则执行紧跟着的复合语句，然后就结束整个语句的执行；如果 if 后括号中的表达式的值为 false，就依次再计算后面的 else if 的表达式值，直到出现某个表达式的值为 true 为止，然后执行该 else if 后面的复合语句，结束整个语句的执行；如果所有的表达式的值都是 false，就执行 else 后面的复合语句，结束整个语句的执行。

#### 4. 循环语句

循环是控制结构语句中的最重要的语句之一，循环语句是根据条件反复执行同一代码块。循环语句有下列三种。

##### (1) while 循环

while 语句的一般格式：

```
while (表达式) {
    若干语句
}
```

while 语句由关键字 while、括号中的一个求值为 boolean 型数据的表达式和一个复合语句组成，其中的复合语句称为循环体，循环体只有一条语句时，大括号 “{ }” 可以省略，但最好不要省略，以便增加程序的可读性。表达式称为循环条件。

##### (2) do-while 循环

一般格式：

```
do {
    若干语句
} while (表达式);
```

do-while 循环和 while 循环的区别是，do-while 的循环体至少被执行一次。

##### (3) for 循环

for 语句的一般格式：

```
for (表达式 1; 表达式 2; 表达式 3) {
    若干语句
}
```

for 语句由关键字 for、括号中用分号分割的 3 个表达式，以及一个复合语句组成，其中的“表达式 2”必须是一个求值为 boolean 型数据的表达式，而复合语句称为循环体。循环体只有一条语句时，大括号“{}”可以省略，但最好不要省略，以便增加程序的可读性。“表达式 1”负责完成变量的初始化；“表达式 2”是值为 boolean 型的表达式，称为循环条件；“表达式 3”用来修整变量，改变循环条件。

### 3.2.4 实验内容

#### 题目 1 输出希腊字母表

编写一个 Java 应用程序，该程序在命令窗口输出希腊字母表。程序运行效果如图 3-12 所示。

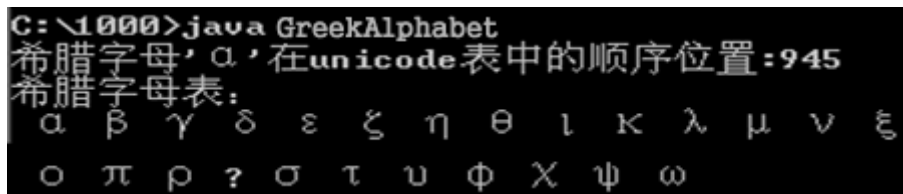


图 3-12 程序运行结果

#### [实验提示]

为了书库希腊字母表，首先应该获取希腊字母表中的第一个字母和最后一个字母在 unicode 表中的位置，然后使用循环输出其余的希腊字母。

#### [程序模版]

```
//GreekAlphabet.java
public class GreekAlphabet{
    public static void main (String args[ ]) {
        int startPosition=0,endPosition=0;
        char cStart='α',cEnd='ω';
        【代码 1】    // cStart 做 int 型转换据运算，并将结果赋值给 startPosition。
        【代码 2】    // cEnd 做 int 型转换运算，并将结果赋值给 endPosition。
        System.out.println("希腊字母'\α\'在 unicode 表中的顺序位置:"+(int)c);
        System.out.println("希腊字母表: ");
        for(int i=startPosition;i<=endPosition;i++) {
            char c='\0';
            【代码 3】    // i 做 int 型转换运算，并将结果赋值给 c。
            System.out.print(" "+c);
        }
    }
}
```

#### 题目 2 回文数

编写一个 Java 应用程序。用户从键盘输入一个 1~9999 之间的数，程序将判断这个数是几位数，并判断这个数是否是回文数。回文数是指将该数字逆序排列后得到的数和原数相同，例如 12121、3223 都是回文数。程序运行效果如图 3-13 所示。

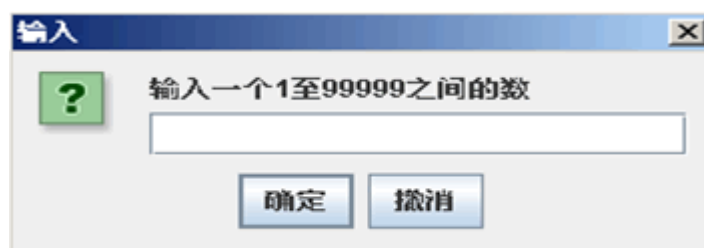


图 3-13 程序运行效果

## [程序模版]

```

import javax.swing.JOptionPane;
public class Number{
    public static void main(String args[]) {
        int number=0,d5,d4,d3,d2,d1;
        String str=JOptionPane.showInputDialog("输入一个 1 至 99999 之间的数");
        number=Integer.parseInt(str);
        if(【代码 1】){ //判断 number 在 1 至 99999 之间的条件。
            【代码 2】    // 计算 number 的最高位（万位）d5。
            【代码 3】    // 计算 number 的千位 d4。
            【代码 4】    // 计算 number 的百位 d3。
            d2=number%100/10;
            d1=number%10;
            if(【代码 5】){ // 判断 number 是 5 位数的条件。
                System.out.println(number+"是 5 位数");
                if(【代码 6】){ // 判断 number 是回文数的条件。
                    System.out.println(number+"是回文数");
                }
                else {
                    System.out.println(number+"不是回文数");
                }
            }
            else if(【代码 7】){ // 判断 number 是 4 位数的条件。
                System.out.println(number+"是 4 位数");
                if(【代码 8】){ // 判断 number 是回文数的条件码。
                    System.out.println(number+"是回文数");
                }
                else {
                    System.out.println(number+"不是回文数");
                }
            }
            else if(【代码 9】){ // 判断 number 是 3 位数的条件。
                System.out.println(number+"是 3 位数");
                if(【代码 10】){ // 判断 number 是回文数的条件。
                    System.out.println(number+"是回文数");
                }
                else {

```

```

        System.out.println(number+"不是回文数");
    }
}
else if(d2!=0) {
    System.out.println(number+"是 2 位数");
    if(d1==d2) {
        System.out.println(number+"是回文数");
    }
    else {
        System.out.println(number+"不是回文数");
    }
}
else if(d1!=0) {
    System.out.println(number+"是 1 位数");
    System.out.println(number+"是回文数");
}
}
else {
    System.out.println(number + "不在 1 至 99999 之间");
}
}
}

```

### 题目 3 猜数字游戏

编写一个 Java 应用程序，实现下列功能：

- ✧ 程序随机分配给客户一个 1~100 之间的整数；
- ✧ 用户从输入对话框输入自己的猜测；
- ✧ 程序返回提示信息，提示信息分别是“猜大了”、“猜小了”和“猜对了”；
- ✧ 用户可根据提示信息再次输入猜测，直到提示信息是“猜对了”。

#### [程序模版]

```

import javax.swing.JOptionPane;
public class GuessNumber{
    public static void main (String args[] ) {
        System.out.println("给你一个 1 至 100 之间的整数,请猜测这个数");
        int realNumber=(int)(Math.random()*100)+1;
        int yourGuess=0;
        String str=JOptionPane.showInputDialog("输入您的猜测:");
        yourGuess=Integer.parseInt(str);
        while(【代码 1】) { // 循环条件。
            if(【代码 2】) { // 条件代码。
                str=JOptionPane.showInputDialog("猜大了,在再输入你的猜测:");
                yourGuess=Integer.parseInt(str);
            }
            else if(【代码 3】) { // 条件代码。
                str=JOptionPane.showInputDialog("猜小了,在再输入你的猜测:");
            }
        }
    }
}

```

```
        yourGuess=Integer.parseInt(str);
    }
}
System.out.println("猜对了!");
}
```

**[思考与扩展]**

1. 定义变量的作用是什么？
2. Java 运算符的优先级与结合性是怎样的？
3. 使用 if 语句实现单分支、两分支和使用 switch 语句实现多分支的程序程序结构和流程分别是什么？
4. 使用 if 语句和 switch 语句都可以实现多分支，它们之间的区别是什么？
5. 使用 while、do-while 和 for 语句实现循环的程序程序结构和流程分别是什么？
6. 使用 while、do-while 和 for 语句实现循环的区别是什么，相互之间可以替换吗，怎样替换？

## 3.3 实验三 类的封装和继承

### 3.3.1 实验类型

验证型实验

### 3.3.2 实验目的

1. 掌握使用类来封装对象的属性和功能
2. 掌握子类的继承性
3. 掌握子类对象的创建过程
4. 掌握成员变量的继承和隐藏
5. 掌握方法的继承

### 3.3.3 知识点介绍

#### 1. 类

面向对象编程核心思想之一就是数据和对数据的操作封装在一起。通过抽象，即从具体的实例中抽取共同的性质形成某种一般的概念，比如类的概念。人们经常谈到的机动车类就是从具体的实例中抽取共同的属性和功能形成的一个概念，而一个具体的轿车是机动车类的一个实例，即对象。对象将数据和对数据的操作合理有效地封装在一起，每辆轿车调用“加大油门”改变的都是自己的运行速度。Java 语言与其他面向对象语言一样，引入了类的概念，类是用来创建对象的模板，它包含被创建的对象属性和功能。Java 程序设计的基本单位是类(class)，Java 的源文件就是由若干个书写形式互相独立的类构成的。

因此，要学习 Java 编程就必须学会怎样去写类，即怎样用 Java 的语法去描述一类事物共有的属性和功能。类有两种基本的成员：变量和方法。变量用来刻画对象的属性，方法用来体现对象的功能，即方法使用某种算法操作变量来实现一个具体的功能。

#### 2. 继承

继承是一种由已有的类创建新类的机制。利用继承，可以先创建一个共有属性的一般类，根据该一般类再创建具有特殊属性的新类，新类继承一般类的状态和行为，并根据需要增加它自己的新的状态和行为。由继承而得到的类称为子类，被继承的类称为父类（超类）。父类可以是 Java 类库中的类，也可以是自己编写的类。利用继承可以有效地实现代码的重复使用，子类只需要添加新的功能代码即可。

Java 不支持多重继承，即子类只能有一个父类。所谓子类继承父类的成员变量作为自己的一个成员变量，就好像它们是在子类中直接声明一样，可以被子类中自己声明的任何实例方法操作，也就是说，一个子类继承的成员应当是这个类的完全意义的成员，如果子类中声明的实例方法不能操作父类的某个成员变量，该成员变量就没有被子类继承；所谓子类继承父类的方法作为子类中的一个方法，就像它们是在子类中直接声明一样，可以被子类中自己声明的任何实例方法调用。

### 3.3.4 实验内容

#### 题目 1 三角形、梯形和圆形的类封装

编写一个 Java 应用程序，该程序中有 3 个类：Trangle, Lader 和 Circle，分别用来刻画“三角形”、

“梯形”和“圆形”。具体要求如下：

1. Trangle 类具有类型为 double 的三个边以及周长、面积属性，Trangle 类具有返回周长、面积以及修改三个边的功能，另外，Trangle 还具有一个 boolean 类型的属性，该属性用来判断三个数是否构成一个三角形；
2. Lader 类具有类型为 double 的上底、下底、高、面积属性，就有返回面积的功能；
3. Circle 类具有类型为 double 的半径、周长和面积属性，具有返回周长、面积的功能。

#### [程序模版]

```
class Trangle {
    double sideA,sideB,sideC,area,length;
    boolean boo;
    public Trangle(double a,double b,double c) {
        【代码 1】 // 参数 a,b,c 分别赋值给 sideA,sideB,sideC。
        if(【代码 2】) { // a,b,c 构成三角形的条件表达式。
            【代码 3】 // 给 boo 赋值。
        }
        else {
            【代码 4】 // 给 boo 赋值。
        }
    }
    double getLength() {
        【代码 5】 // 方法体，要求计算出 length 的值并返回。
    }
    public double getArea() {
        if(boo) {
            double p=(sideA+sideB+sideC)/2.0;
            area=Math.sqrt(p*(p-sideA)*(p-sideB)*(p-sideC));
            return area;
        }
        else {
            System.out.println("不是一个三角形,不能计算面积");
            return 0;
        }
    }
    public void setABC(double a,double b,double c) {
        【代码 6】 // 参数 a,b,c 分别赋值给 sideA,sideB,sideC。
        if(【代码 7】) // a,b,c 构成三角形的条件表达式。 {
            【代码 8】 // 给 boo 赋值。
        }
        else {
            【代码 9】 // 给 boo 赋值。
        }
    }
}

class Lader {
```

```
double above,bottom,height,area;
Lader(double a,double b,double h) {
    【代码 10】 // 方法体。
}
double getArea() {
    【代码 11】 // 方法体, 要求计算出 area 返回。
}
}
class Circle {
    double radius,area;
    Circle(double r) {
        【代码 12】 // 方法体。
    }
    double getArea() {
        【代码 13】 // 方法体, 要求计算出 area 返回。
    }
    double getLength() {
        【代码 14】 // 方法体,要求计算出 length 返回。
    }
    void setRadius(double newRadius) {
        radius=newRadius;
    }
    double getRadius() {
        return radius;
    }
}
}
public class AreaAndLength {
    public static void main(String args[]) {
        double length,area;
        Circle circle=null;
        Trangle trangle;
        Lader lader;
        【代码 15】 // 创建对象 circle。
        【代码 16】 // 创建对象 trangle。
        【代码 17】 // 创建对象 lader。
        【代码 18】 // circle 调用方法返回周长并赋值给 length。
        System.out.println("圆的周长:"+length);
        【代码 19】 // circle 调用方法返回面积并赋值给 area。
        System.out.println("圆的面积:"+area);
        【代码 20】 // trangle 调用方法返回周长并赋值给 length。
        System.out.println("三角形的周长:"+length);
        【代码 21】 // trangle 调用方法返回面积并赋值给 area。
        System.out.println("三角形的面积:"+area);
        【代码 22】 // lader 调用方法返回面积并赋值给 area。
```



```
System.out.println("梯形的面积:"+area);
```

【代码 23】 // triangle 调用方法返回修改三个边的代码，要求将三个边修改为 12,34,1。

【代码 24】 // triangle 调用方法返回面积并赋值给 area。

```
System.out.println("三角形的面积:"+area);
```

【代码 25】 // triangle 调用方法返回周长并赋值给 length。

```
System.out.println("三角形的周长:"+length);
```

```
}
```

```
}
```

## 题目 2 继承

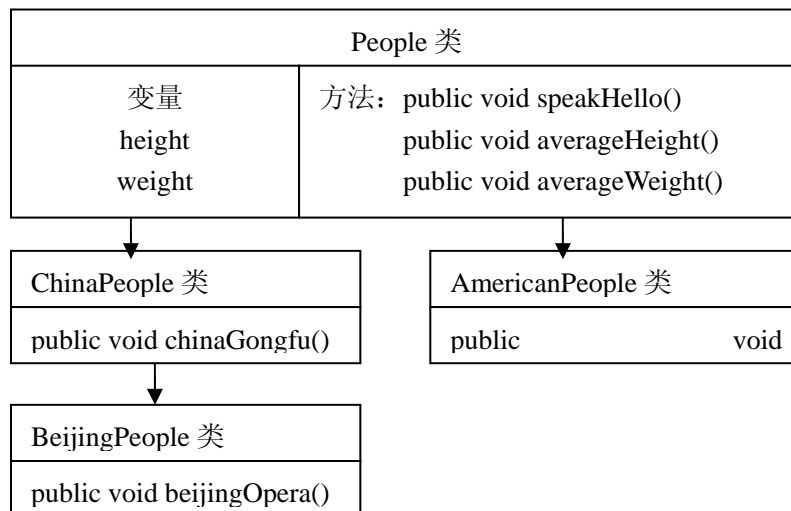
编写一个 Java 程序，除了主类外，该程序中还有 4 个类 People、ChinaPeople、AmericanPeople 和 BeijingPeople 类。要求如下：

(1) People 类含有访问权限为 protected、类型为 double 的成员变量：height 和 weight，以及 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

(2) ChinaPeople 类是 People 的子类，新增了 public void chinaGongfu()方法。要求 chinaPeople 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

(3) AmericanPeople 类是 People 的子类，新增了 public void americanBoxing()方法。要求 AmericanPeople 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

(4) BeijingPeople 类是 ChinaPeople 的子类，新增了 public void beijingOpera()方法。要求 ChinaPeople 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。



### [程序模版]

```
class People{
    protected double weight,height;
    public void speakHello() {
        System.out.println("yayawawa");
    }
    public void averageHeight() {
        height=173;
        System.out.println("average height:"+height);
    }
}
```

```
public void averageWeight() {
    weight=70;
    System.out.println("average weight:"+weight);
}

}

class ChinaPeople extends People{
    【代码 1】 // 重写 public void speakHello()方法，要求输出类似“你好，吃了吗”这样的
                // 汉语信息。
    【代码 2】 // 重写 public void averageHeight()方法，要求输出类似
                // “中国人的平均身高：168.78 厘米”这样的汉语信息。
    【代码 3】 // 重写 public void averageWeight()方法，
                // 要求输出类似“中国人的平均体重：65 公斤”这样的汉语信息。
    public void chinaGongfu() {
        【代码 4】 // 输出中国武术的信息，例如：“坐如钟,站如松,睡如弓”等。
    }
}

class AmericanPeople extends People{
    【代码 5】 // 重写 public void speakHello()方法，要求输出类似
                // “How do you do”这样的英语信息。
    【代码 6】 // 重写 public void averageHeight()方法。
    【代码 7】 // 重写 public void averageWeight()方法。
    public void americanBoxing() {
        【代码 8】 // 输出拳击的信息，例如，“直拳”、“钩拳”等。
    }
}

class BeijingPeople extends ChinaPeople {
    【代码 9】 // 重写 public void speakHello()方法，要求输出类似“您好”这样的汉语信息。
    【代码 10】 // 重写 public void averageHeight()方法。
    【代码 11】 // 重写 public void averageWeight()方法。
    public void beijingOpera() {
        【代码 12】 // 输出京剧的信息。
    }
}

public class Example{
    public static void main(String args[]) {
        ChinaPeople chinaPeople=new ChinaPeople();
        AmericanPeople americanPeople=new AmericanPeople();
        BeijingPeople beijingPeople=new BeijingPeople();
        chinaPeople.speakHello();
        americanPeople.speakHello();
        beijingPeople.speakHello();
        chinaPeople.averageHeight();
        americanPeople.averageHeight();
    }
}
```

```

        beijingPeople.averageHeight();
        chinaPeople.averageWeight();
        americanPeople.averageWeight();
        beijingPeople.averageWeight();
        chinaPeople.chinaGongfu();
        americanPeople.americanBoxing();
        beijingPeople.beijingOpera();
        beijingPeople.chinaGongfu();
    }
}

```

### [思考与扩展]

1. 一个方法或一个块内定义的变量是否可以在方法外或块外使用？这种变量称为什么？方法的形式参数是否可以在方法之外使用？
2. 为什么说构造函数是一种特殊的方法？特殊在哪里？构造函数什么时候执行？被谁调用？
3. 子类重新定义与父类方法的方法头完全相同的方法，这种情况称为什么？
4. 同名的不同方法共存的情况称为什么？如何区分这些同名方法？
5. 定义一个类实现银行帐户的概念，包括的变量有“帐号”和“存款余额”，包括的方法有“存款”、“取款”和“查询余额”。定义主类，创建帐户类的对象，并完成相应操作。

### 关键代码提示

```

public int getleftmoney(){
    return leftmoney;
}

public void savemoney(double money){
    leftmoney+=money;
}

public void getmoney(double money){
    if(money<=leftmoney)
        leftmoney-=money;
    else
        System.out.println("只能取: "+leftmoney);
}
...
bankaccount ba=new bankaccount(123456,1000);
ba.savemoney(2000);
System.out.println("存入 2000 元后余额为: "+ba.getleftmoney());
ba.getmoney(1500);
System.out.println("1500 元后余额为: "+ba.getleftmoney());

```

## 3.4 实验四 多态与接口

### 3.4.1 实验类型

验证型实验

### 3.4.2 实验目的

1. 了解抽象类和接口的原理。
2. 能运用抽象类和接口
3. 理解多态的概念。
4. 学习实现多态。

### 3.4.3 知识点介绍

多态是面向对象编程的又一重要特性，多态时指某个类的不同子雷可根据各执的需要重写父类的某个方法。

接口是 Java 中非常重要的概念，必须很好地理解和掌握。接口的思想是在于它可以增加类需要实现的功能，不同的类可以使用相同的结构，同一个类可以实现多个接口。

### 3.4.4 实验内容

#### 题目 1 定义一个接口，用于查询课程

定义一个类 Student，包括如下属性：学号 (ID)、姓名 (name)、性别 (Sex)、出生日期 (birthDate)、专业 (specialty)、课程 (course)，实现以下方法：每个属性的获取和定义，要求至少包含一个构造函数。定义一个接口类，定义方法 query\_course\_catalog() 用来查询课程。编写一个接口，定义相关选课操作，定义 Student 实现该接口。

#### [程序模版]

// 定义一个接口，用于查询课程

```
public interface ____【代码 1】____ {  
    // 根据专业查询课程  
    public String  query_course_catalog (String specialty);  
}
```

// Student 类，并实现接口

```
public class Student implements ____【代码 2】____{  
    int ID; // 学号  
    String name[]; // 姓名  
    char sex; // 性别  
    String birthdate; // 出生日期 yyyy-MM-dd  
    String specialty; // 专业  
    Sting course; // 课程  
    public Student (int ID,String  name,char sex,String birthdate ,String specialty ,String course ) {
```

// 实现构造函数，用于初始化信息

【代码段 3】

}

public showInfor(){ // 输出学生的基本信息

【代码段 4】

}

public String query\_course\_catalog (String speciality){

return “welcome to you!”;

}

public static void main(String[] args){

Student student=new Student(【代码 5】);

String str= student. query\_course\_catalog(【代码 6】);

System.out.println(str);

System.out.println(student);

}

}

## 题目 2 抽象类和抽象方法

1. 定义一个抽象类 Shape,类里有一个抽象方法 display();
2. 定义一个 Circle 类，继承了 Shape 类，并实现抽象方法 display();
3. 定义一个 Rectangle 类，继承了 Shape 类，并实现抽象方法 display();
4. 定义一个 Triangle 类，继承了 Shape 类，并实现抽象方法 display();
5. 定义一个类，实例化上述几个类的对象，并调用上述几个类的 display 方法。

## 题目 3 改用接口来实现题目 2

### [思考与扩展]

1. 同名的不同方法共存的情况称为什么？如何区分这些同名方法？
2. 创建一个类，声明一个无参数的构造函数，打印类已创建的信息；再重载一个具有 String 参数的构造函数，打印参数信息；并创建主类验证之。
3. 定义一个矩形类，再定义接口 EqualDiagonal，其中包含方法 getDiagonal(); 由矩形类派生出一个正方形类，自行扩充成员变量和方法，并实现此接口 EqualDiagonal。

## 3.5 实验五 常用基础类库与工具类库

### 3.5.1 实验类型

验证型实验

### 3.5.2 实验目的

1. 掌握 Data 类和 Calender 类。
2. 掌握 LinkedList 类的常用方法。

### 3.5.3 知识点介绍

Date 类在 java.util 包中。使用 Date 类的五参数构造方法创建的对象可以获取本地当前时间。用 Date 的构造方法 Date 创建的 Date 对象表示相对 1970 年 1 月 1 日 0 点 (GMT) 的时间, 例如参数 time 取值 60\*60\*1000 秒表示 Thu Jan 01 01: 00: 00GMT1970。

可以使用 DateFormat 的子类 SimpleDateFormat 来实现日期的格式化。SimpleDateFormat 有一个常用构造方法: public SimpleDateFormat (String pattern)。该构造方法可以用参数 pattern 指定的格式创建一个对象, 该对象调用 format (Date date) 方法格式化时间对象 date。

需要注意的是, pattern 中应当含有一些有效的字符序列。

Calendar 类在 java.util 包中。使用 Calendar 类的 static 方法 getInstance () 可以初始化一个日历对象, 如: Calendar calendar=Calendar.getInstance ()。

然后, calendar 对象可以调用方法:

public final void set (int year, int month, int date) 或

public final void set (int year, int month, int date, int hour, int minute) 或

public final void set (int year, int month, int date, int hour, int minute, int second)

将日历翻到任何一个时间, 当参数 year 取负数是表示公元前。

使用 LinkedList 类可以创建链表结构的数据对象。链表是由若干个节点组成的一种数据结构, 每个节点含有一个数据和下一个节点的引用 (单链表), 或含有一个数据并含有上一个节点的引用和下一个节点的引用 (双链表), 节点的索引从 0 开始。链表适合动态地改变它存储的数据, 如动态地增加、删除节点等。

### 3.5.4 实验内容

#### 题目 1 比较日期的大小

编写一个 Java 应用程序, 用户从输入对话框输入两个日期, 程序将判断两个日期的大小关系, 以及两个日期之间的间隔天数。程序运行效果如图 3-14 所示。



图 3-14 程序运行效果

## [程序模版]

```

import java.util.*;
import javax.swing.JOptionPane;
class DateExample{
    public static void main(String args[ ]) {
        String str=JOptionPane.showInputDialog("输入第一个日期的年份:");
        int yearOne=Integer.parseInt(str);
        str=JOptionPane.showInputDialog("输入该年的月份:");
        int monthOne=Integer.parseInt(str);
        str=JOptionPane.showInputDialog("输入该月份的日期:");
        int dayOne=Integer.parseInt(str);
        str=JOptionPane.showInputDialog("输入第二个日期的年份:");
        int yearTwo=Integer.parseInt(str);
        str=JOptionPane.showInputDialog("输入该年的月份:");
        int monthTwo=Integer.parseInt(str);
        str=JOptionPane.showInputDialog("输入该月份的日期:");
        int dayTwo=Integer.parseInt(str);
        Calendar calendar=【代码 1】 // 初始化日历对象。
        【代码 2】 // 将 calendar 的时间设置为 yearOne 年 monthOne 月 dayOne 日。
        long timeOne=【代码 3】 // calendar 表示的时间转换成毫秒。
        【代码 4】 // 将 calendar 的时间设置为 yearTwo 年 monthTwo 月 dayTwo 日。
        long timeTwo=【代码 5】 // calendar 表示的时间转换成毫秒。
        Date date1=【代码 6】 // 用 timeOne 做参数构造 date1。
        Date date2=【代码 7】 // 用 timeTwo 做参数构造 date2。
        if(date2.equals(date1)) {
            System.out.println("两个日期的年、月、日完全相同");
        }
        else if(date2.after(date1)) {
            System.out.println("您输入的第二个日期大于第一个日期");
        }
        else if(date2.before(date1)) {
            System.out.println("您输入的第二个日期小于第一个日期");
        }
        long days=【代码 8】 // 计算两个日期相隔天数。
        System.out.println(yearOne+"年"+monthOne+"月"+dayOne+"日和"
            +yearTwo+"年"+monthTwo+"月"+dayTwo+"相隔"+days+"天");
    }
}

```

```

    }
}

```

## 题目 2 随机布雷

首先编写一个 Block 类。Block 对象具有 String 类型和 boolean 类型的成员变量，Block 对象可以使用 setName (String) 方法、getName () 方法、setBoolean (boolean) 方法来设置对象的名字，返回该对象的名字，放回对象的 boolean 类型成员的值，设置对象的 boolean 类型的值。

在主类中，要求用一个 Block 型的二维数组的每个单元式一个 Block 对象。然后将二维数组的各个单元中的对象存放在一个链表中。

要求在 8×8 的方阵中随机布雷 25 个。程序运行效果如图 3-15 所示。

### [程序模版]

```

import java.util.*;
class Block {
    String name;
    boolean boo=false;
    public void setName(String name) {
        【代码 1】 // 将参数 name 传值给成员变量。
    }
    public String getName() {
        【代码 2】 // 返回成员变量 name。
    }
    boolean isMine() {
        【代码 3】 // 返回成员变量 boo。
    }
    public void setBoolean(boolean boo) {
        【代码 4】 // 将参数 boo 传值给成员变量 boo。
    }
}
class MineExample{
    public static void main(String args[]) {
        int mine=25;
        Block block[][]=new Block[8][8];
        for(int i=0;i<8;i++) {
            for(int j=0;j<8;j++) {
                block[i][j]=new Block();
            }
        }
        LinkedList list= 【代码 5】 // 创建 list。
        for(int i=0;i<8;i++) {
            for(int j=0;j<8;j++) {
                【代码 6】 // 将 block[i][j]添加到 list。
            }
        }
        while(mine>=0) {
            int size= 【代码 7】 // 返回 list 中节点个数。

```

```

G:\Example>java MineExample
1 2 0 1 2 0 5 0
0 3 1 2 4 0 0 0
0 2 0 2 0 0 0 0
2 2 1 3 0 0 0 4
0 2 2 0 4 6 0 3
2 0 2 3 0 4 0 3
1 2 2 3 0 4 4 0
0 1 0 2 2 0 3 0

```

图 3-15 程序运行效果



```

        int randomIndex=(int)(Math.random()*size);
        Block b=【代码 8】 // 返回 list 中索引值为 randomIndex 的节点中的对象。
        b.setName("@");
        b.setBoolean(true);
        【代码 9】 // list 删除索引值为 randomIndex 的节点。
        mine--;
    }
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++){
            if(block[i][j].isMine()) {
                }
            else {
                int mineNumber=0;
                for(int k=Math.max(i-1,0);k<=Math.min(i+1,7);k++) {
                    for(int t=Math.max(j-1,0);t<=Math.min(j+1,7);t++) {
                        if(block[k][t].isMine())
                            mineNumber++;
                    }
                }
                block[i][j].setName(""+mineNumber);
            }
        }
    }
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++){
            System.out.print("  "+block[i][j].getName());
        }
        System.out.println("");
    }
}
}

```

#### [思考与扩展]

1. 怎样实例化一个 Calendar 对象？
2. 编写一个应用程序，输出某年某月的日历页，通过 main 方法的参数将年份和月份时间传递到程序中。

## 3.6 实验六 异常处理

### 3.6.1 实验类型

验证型实验 2 学时

### 3.6.2 实验目的

1. 掌握异常的概念。
2. 掌握异常的定义、抛出和捕捉处理异常。

### 3.6.3 知识点介绍

异常是指程序运行时出现的非正常情况。在传统的语言编程时，程序员只能通过函数的返回值来发出错误的信息。这容易导致很多的错误，因为在很多情况下需要知道错误产生的内部细节。

Java 对“异常”的处理是面向对象的。一个 Java 的 `Exception` 是一个描述“异常情况的对象”。当出现“异常”情况时，一个 `Exception` 对象就产生了，并放到产生这个“异常”的成员函数里。

Java 的“异常”处理通过 5 个关键词来实现：`try`、`catch`、`throw`、`throws` 和 `finally`。用 `try` 来执行一段程序，如果出现“异常”，系统抛出（`throws`）一个“异常”，可以通过它的类型来捕捉（`catch`）它，或最后（`finally`）有缺省处理器来处理。

### 3.6.4 实验内容

**题目 1** 运行下面的程序，理解异常的抛出、捕捉与处理。

```
import java.io.*;

public class ExceptionTest{
    public static void main(String args[]) {
        for(int i = 0; i < 4;i++) {
            int k;
            try {
                switch(i) {
                    case 0:           // divided by zero
                        int zero = 0;
                        k = 911 / zero;
                        break;
                    case 1:           // null pointer
                        int b[ ] = null;
                        k = b[0];
                        break;
                    case 2:           // array index out of bound
                        int c[ ] = new int[2];
                        k = c[9];
                }
            }
        }
    }
}
```

### 题目 2 运行下面的程序，理解异常类的常用方法的使用。

### 考与扩展

## 3.7 实验七 多线程机制

### 3.7.1 实验类型

验证型实验

### 3.7.2 实验目的

1. 掌握线程的概念、线程的生命周期。
2. 掌握采用继承 `Thread` 类创建子线程。
3. 掌握使用 `Runnable` 接口使类线程化。
4. 掌握数据间的共享与独有。
5. 了解 `wait` 方法和 `notifyAll` 方法的使用。

### 3.7.3 知识点介绍

可以使用 `Thread` 类或 `Thread` 类的子类创建对象。线程是比进程更小的执行单位。一个进程在其执行过程中，可以产生多个线程，形成多条执行线索。每条线索，即每个线程，也有它自身的产生、存在和消亡的过程，也是一个动态的概念。线程在它的一个完整的生命周期中通常要经历 4 种状态：新建、运行、中断和死亡。Java 虚拟机（JVM）中的线程调度器负责管理线程，在采用时间片的系统中，每个线程都有机会获得 CUP 的使用权。当线程使用 CUP 资源的时间终了时，即使线程没有完成自己的全部操作，Java 调度器也会中断当前线程的执行，把 CUP 的使用权切换给下一个排队等待的线程，当前线程将等待 CUP 资源的下一次轮回，然后从中断处继续执行。Java 虚拟机中的线程调度器把线程的优先级分为 10 个级别，分别用 `Thread` 类中的类常量表示，在实际编程时，不提倡使用线程的优先级来保证算法的正确执行。要编写正确的、跨平台的多线程代码，必须假设线程在任何时间都有可能被剥夺 CPU 资源的使用权。

线程同步是很重要的内容之一，在处理线程同步时，要做的第一件事就是要把修改数据的方法用关键字 `synchronized` 来修饰。一个方法使用关键字 `synchronized` 修饰后，如果一个线程 A 占有 CPU 资源期间，使得该方法被调用执行，那么在该同步方法返回之前，即同步方法调用执行完毕之前，其他占有 CUP 资源的线程一旦调用这个同步方法就会引起堵塞，堵塞的线程要一直等到堵塞的原因消除（同步方法返回），再排队等待 CUP 资源，以便使用这个同步方法。

在处理同步问题时，有时会涉及到一些特殊问题，如当一个线程使用的同步方法中用到某个变量，而此变量又需要其他线程修改后才能符合本线程的需要。为了解决这样的问题，正在使用同步方法的线程可以执行 `wait()` 方法，使用 `wait()` 方法可以中断线程的执行，使本线程等待，暂时让出 CPU 的使用权，并允许其他线程使用这个同步方法。其他线程如果在使用这个同步方法时不需要等待，那么它使用完这个同步方法的同时，应当用 `notifyAll()` 方法通知所有的由于使用这个同步方法而处于等待的线程结束等待。曾中断的线程就会重新排队等待 CUP 资源，以便从刚才的中断处继续执行这个同步方法。

线程联合也是编写多线程问题中经常使用的技术，一个线程 A 在占有 CUP 资源期间，可以让线程 B 和线程 A 联合：`B.join()`，称 A 在运行期间联合了 B。如果线程 A 在占有 CUP 资源期间一旦联合线程 B，那么线程 A 将立刻中断执行，一直等到它联合的线程 B 执行完毕，线程 A 再重新排队等待 CUP 资源，以便恢复执行。如果线程 A 准备联合的线程 B 已经结束或没有就绪排队等待 CUP 资源，那么 `B.join()` 不会产生任何效果。

线程分为守护（Daemon）线程和非守护线程，非守护线程也称为用户线程。一个线程调用 `setDaemon(boolean on)` 方法可以将自己设置成一个守护线程，如

```
thread.setDaemon(true);
```

当程序中的所有用户线程都已结束运行时，即使守护线程的 `run()` 方法中还有需要执行的语句，守护线程也立刻结束运行。

### 3.7.4 实验内容

#### 题目 1 线程的生命周期

编写一个 Java 应用程序，在主线程中创建 2 个线程，要求线程经历新建、运行、中断和死亡 4 个状态。程序运行的效果如图 3-16 所示。

```
G:\Example>javac ThreadExample.java
G:\Example>java ThreadExample
*****
*****兔子进入死亡状态
*****乌龟进入死亡状态
```

图 3-16 程序运行效果

[程序模版]

```
class Tortoise extends Thread{
    int sleepTime=0, liveLength=0;
    Tortoise(int sleepTime, String name, int liveLength) {
        this.sleepTime=sleepTime;
        this.liveLength=liveLength;
        setName(name);
    }
    public void run() {
        while(true){
            liveLength--;
            System.out.print("@");
            try{
                【代码 1】 // 让线程调用 sleep 方法进入中断状态，sleepTime 毫秒后
                           // 线程重新排队等待 CPU 资源
            }
            catch(InterruptedException e) {
            }
            if(liveLength<=0) {
                System.out.print(getName()+"进入死亡状态\n");
                【代码 2】 // 结束 run 方法的语句
            }
        }
    }
}

class Rabbit extends Thread{
    int sleepTime=0, liveLength;
    Rabbit(int sleepTime, String name, int liveLength) {
```

```

        this.sleepTime=sleepTime;
        this.liveLength=liveLength;
        setName(name);
    }
    public void run() {
        while(true) {
            liveLength--;
            System.out.print("*");
            try{
                【代码 3】 // 让线程调用 sleep 方法进入中断状态，sleepTime 毫秒后线程
                        // 重新排队等待 CUP 资源
            }
            catch(InterruptedException e) {
            }
            if(liveLength<=0) {
                System.out.print(getName()+"进入死亡状态\n");
                【代码 4】 // 结束 run 方法的语句
            }
        }
    }
}

public class ThreadExample{
    public static void main(String args[ ]) {
        Rabbit rabbit;
        rabbit= 【代码 5】 // 新建线程 rabbit。
        Tortoise tortoise;
        tortoise= 【代码 6】 // 新建线程 tortoise
        【代码 7】 // 启动线程 tortoise
        【代码 8】 // 启动线程 rabbit
    }
}

```

## 题目 2 线程之间共享数据

编写一个 Java 应用程序，在主线程中用 Thread 类创建 2 个线程，2 个线程共享一个 int 型的数据，并各自有自己独占的数据。程序运行效果如图 3-17 所示。

### [程序模版]

```

class Bank implements Runnable{
    【代码 1】 // 声明一个 int 型变量
    // money，初值为 100。
    Thread zhang,keven;
    Bank() {
        【代码 2】 // 创建 zhang，Bank
        // 对象为 zhang 的目标对象
    }
}

```

```

会计将money的值改为101
出纳将money的值改为100
出纳线程进入死亡状态
会计将money的值改为101
出纳将money的值改为100
出纳线程进入死亡状态
会计将money的值改为101
出纳将money的值改为100
出纳线程进入死亡状态
会计将money的值改为101
出纳将money的值改为99
出纳线程进入死亡状态
会计将money的值改为100
出纳将money的值改为99
出纳线程进入死亡状态
会计将money的值改为100
出纳将money的值改为100
出纳线程进入死亡状态
会计将money的值改为100
出纳线程进入死亡状态

```

图 3-17 程序运行效果

```

    zhang.setName("会计");
    【代码 3】    // 创建 keven，Bank 对象为 keven 的目标对象
    keven.setName("出纳");
}
public void run(){
    【代码 4】    // 声明一个 int 型变量 i，初值为 0。
    while(true) {
        if(【代码 5】) { // 判断线程 zhang 是否正在占有 CUP 资源
            i=i+1;
            money=money+1;
            System.out.println(zhang.getName()+"将 money 的值改为"+money);
            System.out.print(zhang.getName()+"的局部变量 i="+i);
            if(i>=6) {
                System.out.println(zhang.getName()+"线程进入死亡状态");
                【代码 6】    // 使得线程 zhang 进入死亡状态
            }
            try{
                Thread.sleep(1000);
            }
            catch(InterruptedException e) {
            }
        }
        else if(【代码 7】) { // 判断线程 keven 是否正在占有 CUP 资源
            i=i-1;
            money=money-1;
            System.out.println(keven.getName()+"将 money 的值改为"+money);
            System.out.print(keven.getName()+"的局部变量 i="+i);
            if(i<=-6) {
                System.out.println(keven.getName()+"线程进入死亡状态");
                【代码 8】    // 使得线程 keven 进入死亡状态
            }
            try{
                Thread.sleep(1000);
            }
            catch(InterruptedException e) {
            }
        }
    }
}
}

class BankExample{
    public static void main(String args[ ]) {
        Bank bank=new Bank();
        bank.zhang.start();
    }
}

```

```

        bank.keven.start();
    }
}

```

### 题目 3 线程的挂起、恢复与终止

编写一个 Java 应用程序。同过单击“开始”按钮启动线程，该线程负责移动一个红色的标签。同过单击“挂起”按钮暂时中断线程的执行，单击“恢复”按钮恢复线程。通过单击“终止”按钮终止线程。程序运行效果如图 3-18 所示。



图 3-18 程序运行效果

#### [程序模版]

```

import java.awt.*;
import java.awt.event.*;

class Win extends Frame implements Runnable, ActionListener {
    Thread moveOrStop;
    Button start, hang, resume, die;
    Label moveLabel;
    boolean move=false, dead=false;

    Win() {
        【代码 1】 // 创建线程 moveOrStop，当前窗口为 moveOrStop 的目标对象
        start=new Button("线程开始");
        hang=new Button("线程挂起");
        resume=new Button("线程恢复");
        die=new Button("线程终止");
        start.addActionListener(this);
        hang.addActionListener(this);
        resume.addActionListener(this);
        die.addActionListener(this);
        moveLabel=new Label("线程负责运动我");
        moveLabel.setBackground(Color.red);
        setLayout(new FlowLayout());
        add(start);
        add(hang);
        add(resume);
        add(die);
        add(moveLabel);
        setSize(500,500);
        validate();
        setVisible(true);
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {

```



```
        System.exit(0);
    }
}

);

}

public void actionPerformed(ActionEvent e) {
    if(e.getSource()==start) {
        try {
            move=true;
            【代码 2】 // 启动线程 moveOrStop。
        }
        catch(Exception event) {
        }
    }
    else if(e.getSource()==hang) {
        move=false;
    }
    else if(e.getSource()==resume) {
        move=true;
        【代码 3】 // 调用 resumeThread()方法恢复线程
    }
    else if(e.getSource()==die) {
        dead=true;
    }
}

public void run(){
    while(true) {
        while(!move) {
            try{
                【代码 4】 // 调用 hangThread()方法挂起线程
            }
            catch(InterruptedException e1) {
            }
        }
        int x=moveLabel.getBounds().x;
        int y=moveLabel.getBounds().y;
        y=y+2;
        if(y>=200) y=10;
        moveLabel.setLocation(x,y);
        try{
            moveOrStop.sleep(200);
        }
        catch(InterruptedException e2) {
        }
    }
}
```

```
        }
        if(dead==true) {
            【代码 5】    // 结束 run()方法终止线程
        }
    }
}

public synchronized void  hangThread() throws InterruptedException{
    【代码 6】    // 调用 wait()方法
}

public synchronized void  resumeThread(){
    【代码 7】    //调用 notifyAll()方法
}
}

public class StartOrStop{
    public static void main(String args[]){
        Win win=new Win();
    }
}
```

**[思考与扩展]**

1. 简述并区分程序、进程和线程三个概念。
2. 线程有哪几个基本的状态？Java 中线程调度遵循何种原则？
3. 实现多线程可以用哪两种基本方法？将这两种方法进行比较。

## 3.8 实验八 流式输入输出

### 3.8.1 实验类型

验证型实验

### 3.8.2 实验目的

1. 掌握流的概念。
2. 掌握字符输入/输出流以及缓冲流的使用。
3. 掌握使用流读取文件的内容，并将内容写入到内存。
4. 掌握数据流、对象流的使用

### 3.8.3 知识点介绍

当程序需要读取磁盘上的数据或将程序中的数据存储在磁盘时，就可以使用输入/输出流，简称 I/O 流。I/O 流提供一条通道程序，可以使用这条通道读取“源”中的数据，或把数据送到“目的地”。I/O 流中的输入流的指向称为源，程序从指向源的输入流中读取源中的数据；输出流的指向称为目的地，程序通过向输出流中写入数据把信息传递到目的地。

#### 1. 文件字节流

`FileInputStream` 类是 `InputStream` 的子类，该类创建的对象称为文件字节输入流。文件字节输入流按字节读取文件中的数据。`FileInputStream` 流顺序地读取文件，只要不关闭流，每次调用读取方法时就顺序地读取文件中其余的内容，直到文件的末尾或流被关闭。

`FileOutputStream` 类是 `OutputStream` 的子类，该类创建的对象称为文件字节输出流。文件字节输出流按字节将数据写入到文件中。`FileOutputStream` 流顺序地写文件，只要不关闭流，每次调用写入方法就顺序地向文件写入内容，直到流被关闭。

#### 2. 文件字符流

`FileReader` 类是 `Reader` 的子类，该类创建的对象称为文件字符输入流。文件字符输入流按字符读取文件中的数据。`FileReader` 流顺序地读取文件，只要不关闭流，每次调用读取方法时就顺序地读取文件中其余的内容，直到文件的末尾或流被关闭。

`FileWriter` 类是 `Writer` 的子类，该类创建的对象称为文件字符输出流。字符输出流按字符将数据写入到文件中。`FileWriter` 流顺序地写文件，只要不关闭流，每次调用写入方法就顺序地向文件写入内容，直到流被关闭。

#### 3. 缓冲流

`BufferedReader` 类创建的对象称为缓冲输入流，该输入流的指向必须是一个 `Reader` 流，称为 `BufferedReader` 流的底层流。底层流负责将数据读入缓冲区，`BufferedReader` 流的源就是这个缓冲区，缓冲输入流再从缓冲区中读取数据。

`BufferedWriter` 类创建的对象称为缓冲输出，可以将 `BufferedWriter` 流和 `FileWriter` 流连接在一起，然后使用 `BufferedWriter` 流将数据写到目的地，`FileWriter` 流称为 `BufferedWriter` 的底层流。`BufferedWriter` 流将数据写入缓冲区，底层流负责将数据写到最终的目的地。

#### 4. 字符串流

`StringReader` 使用字符串作为流的源，`StringWriter` 将内存作为流的目的地。

## 5. 数据流

`DataInputStream` 类和 `DataOutputStream` 类创建的对象称为数据输入流和数据输出流。这两个流可以以与机器无关的风格读取 Java 原始数据。

## 6. 对象流

`ObjectInputStream` 类和 `ObjectOutputStream` 类创建的对象被称为对象输入流和对象输出流。对象输出流可以将一个对象写入输出流送往目的地，对象输入流可以从源中读取一个对象到程序中。当我们使用对象流写入或读入对象时，要保证对象是序列化的。这是为了保证能把对象写入到文件，并能再把对象正确读回到程序中的缘故。一个类如果实现了 `Serializable` 接口，那么这个类创建的对象就是所谓序列化的对象。`Serializable` 接口中的方法对程序是不可见的，因此实现该接口的类不需要实现额外的方法。

### 3.8.4 实验内容

#### 题目 1 给文件的内容添加行号

编写一个 Java 应用程序，给已存在的.txt 文本文件添加行号。要求该文本文件事先用文本编辑完毕，保存在 C:\1000 目录中，命名为 hello.txt。程序运行效果如图 3-19 所示。

[程序模版]

```
import java.io.*;

public class ReadExample{
    public static void main(String args[ ]) {
        File file=new File("c:/1000","hello.txt");
        File tempFile=new File("temp.text");
        try{
            FileReader inOne=【代码 1】 // 创建指向文件 file 的输入流。
            BufferedReader inTwo= 【代码 2】 // 创建指向 inOne file 的输入流。
            FileWriter tofile=【代码 3】 // 创建指向文件 tempFile 的输出流。
            BufferedWriter out=【代码 4】 // 创建指向 tofile 的输出流。
            String s=null;
            int i=0;
            s=【代码 5】//inTwo 读取一行。
            while(s!=null) {
                i++;
                out.write(i+" "+s);
                out.newLine();
                s=【代码 6】//inTwo 读取一行。
            }
            inOne.close();
            inTwo.close();
            out.flush();
            out.close();
            tofile.close();
            inOne=【代码 7】 // 创建指向文件 tempFile 的输入流
            inTwo= 【代码 8】 // 创建指向 inOne file 的输入流。
            tofile=【代码 9】 // 创建指向文件 file 的输出流。
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

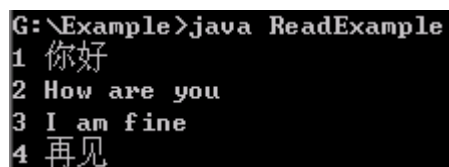


图 3-19 程序运行效果

```

        out=【代码 10】 // 创建指向 tofile 的输出流。
        while((s=【代码 11】)!=null) { // inTwo 读取一行。

            out.write(s);
            out.newLine();
        }
        inOne.close();
        inTwo.close();
        out.flush();
        out.close();
        tofile.close();
        inOne=【代码 12】 // 创建指向文件 file 的输入流。
        inTwo=【代码 13】 // 创建指向 inOne file 的输入流。
        while((s=【代码 14】)!=null) { // inTwo 读取一行。

            System.out.println(s);
        }
        inOne.close();
        inTwo.close();
        tempFile.delete();
    }
    catch(IOException e) {
        System.out.println(e);
    }
}
}

```

## 题目 2 读写基本数据类型数据

编写一个 Java 应用程序，将若干基本数据写入到一个文件，然后再按顺序读出。

### [程序模版]

```

import java.io.*;
public class NumberExample{
    public static void main(String args[]){
        int a1=12,a2=1180;
        long b=808080;
        float x1=3.14F,x2=12.456F;
        double d=1234.9876;
        boolean boo1=true,boo2=false;
        char c='我';
        File f=【代码 1】 // 创建文件。
        try {
            FileOutputStream fos=【代码 1】 // 创建指向 f 文件输出流
            DataOutputStream out_data=【代码 2】 // 创建指向 fos 的数据输出流
            【代码 2】 // out_data 将数据 a1 写入到文件
            【代码 3】 // out_data 将数据 a2 写入到文件
            【代码 4】 // out_data 将数据 b 写入到文件

```

```

        【代码 5】           // out_data 将数据 x1 写入到文件
        【代码 6】           // out_data 将数据 x2 写入到文件
        【代码 7】           // out_data 将数据 d 写入到文件
        【代码 8】           // out_data 将数据 boo1 写入到文件
        【代码 9】           // out_data 将数据 boo2 写入到文件
        【代码 10】          // out_data 将数据 c 写入到文件
    }
    catch (IOException e){
    }
    try{
        FileInputStream fis= 【代码 11】           // 创建指向 f 文件输入流
        DataInputStream in_data= 【代码 12】        // 创建指向 fis 的数据输入流
        System.out.println( 【代码 13】 );          // in_data 读取 int 整数
        System.out.println( 【代码 14】 );          // in_data 读取 int 整数
        System.out.println( 【代码 15】 );          // in_data 读取 long 整数
        System.out.println( 【代码 16】 );          // in_data 读取 float 数
        System.out.println( 【代码 17】 );          // in_data 读取 float 数
        System.out.println( 【代码 18】 );          // in_data 读取 double 数
        System.out.println( 【代码 19】 );          // in_data 读取 boolean 数据
        System.out.println( 【代码 20】 );          // in_data 读取 boolean 数据
        System.out.print( 【代码 21】 );            // in_data 读取 char 数据
    }
    catch(IOException e){
    }
}
}

```

### 题目 3 对象的写入和读取

编写一个 Java 应用程序，将一个 Calendar 对象写入文件，然后顺序读出该对象，并验证读出的对象是否是原始对象的克隆。

#### [程序模版]

```

import java.util.*;
import java.io.*;
public class ObjectExample{
    public static void main(String args[ ]) {
        Calendar calendar1=Calendar.getInstance();    // 创建一个日历对象
        Calendar calendar2=Calendar.getInstance();    // 创建一个日历对象
        calendar1.set(1949,9,1);                      // 将日历时间设置为 1949 年 10 月 1 日,注意 9 表示十月
        calendar2.set(2005,9,1);
        try{
            File f= 【代码 1】           // 创建一个名字为"a.txt"的文件
            FileOutputStream fileOut= 【代码 2】 // 创建指向文件 f 的文件输出流
            ObjectOutputStream objectOut= 【代码 3】 // 创建指向文件 fileOut 的对象输出流
            【代码 4】           // objectOut 写对象 calendar1 到文件
            【代码 5】           // objectOut 写对象 calendar2 到文件
            FileInputStream fileIn= 【代码 6】 // 创建指向文件 f 的文件输入流
            ObjectInputStream objectIn 【代码 7】 // 创建指向文件 fileIn 的对象输入流

```

```

Calendar cloneCalendar1=【代码 8】    // objectOut 读出对象
Calendar cloneCalendar2=【代码 9】    // objectOut 读出对象
cloneCalendar1.setTime(new Date());
cloneCalendar2.set(9999,9,1);
int 年=calendar1.get(Calendar.YEAR),
    月=calendar1.get(Calendar.MONTH)+1,
    日=calendar1.get(Calendar.DAY_OF_MONTH);
System.out.printf("\ncalendar1 的日期:%d-%d-%d",年,月,日);
年=calendar2.get(Calendar.YEAR);
月=calendar2.get(Calendar.MONTH)+1;
日=calendar2.get(Calendar.DAY_OF_MONTH);
System.out.printf("\ncalendar2 的日期:%d-%d-%d",年,月,日);
年=cloneCalendar1.get(Calendar.YEAR);
月=cloneCalendar1.get(Calendar.MONTH)+1;
日=cloneCalendar1.get(Calendar.DAY_OF_MONTH);
System.out.printf("\ncloneCalendar1 的日期:%d-%d-%d",年,月,日);
年=cloneCalendar2.get(Calendar.YEAR);
月=cloneCalendar2.get(Calendar.MONTH)+1;
日=cloneCalendar2.get(Calendar.DAY_OF_MONTH);
System.out.printf("\ncloneCalendar2 的日期:%d-%d-%d",年,月,日);
}
catch(Exception event) {
    System.out.println(event);
}
}
}

```

#### [思考与扩展]

1. 如果准备读取一个文件的内容，应当使用 `FileInputStream` 流还是 `FileOutputStream` 流？
2. `FileInputStream` 流的 `read()` 方法和 `FileReader` 流的 `read()` 方法有何不同？
3. 使用 `ObjectInputStream` 类和 `ObjectOutputStream` 类有哪些注意事项？

## 3.9 实验九 Applet 简单应用

### 3.9.1 实验类型

验证型实验

### 3.9.2 实验目的

1. 熟悉简单 Applet 的编写和运行
2. 掌握多媒体 Applet 的实现

### 3.9.3 知识点介绍

一个 Java Applet 也是由若干个类组成的，一个 Java Applet 不再需要 main()方法，但必须有且只有一个类扩展了 Applet 类，即它是 Applet 类的子类，我们把这个类叫做这个 Java Applet 的主类，Java Applet 的主类必须是 public 的。Java Applet 属于嵌入到浏览器环境中的程序，必须由浏览器中的 JVM 负责执行。当 Java Applet 编译通过之后，我们必须编写一个超文本文件（含有 applet 标记的 Web 页）告诉浏览器来运行这个 Java Applet。假设 Applet 主类的名字是 Boy,下面是一个简单的 html 文件 like.html,

```
<applet code="Boy.class" height="180" width="300">
</applet>
```

like.html 文件告诉浏览器运行主类是 Boy 的 Java Applet。

网页的最终目的是让其他客户通过网络来访问，下载到客户端执行，可以用 Web 发布管理器（Windows 98 安装盘可以安装个人 Web 管理器）或 IIS 将含有 Java Applet 网页所在的目录设成 Web 共享。假如，我们将 like.html 所在的文件夹 C:\1000 设为 Web 共享目录，共享的名称是 hello, 那么其他用户就可以在其浏览器的地址栏中输入服务器的 IP 地址、共享 Web 目录和该目录中的 html 文件，来下载执行含有 Java Applet 程序的网页。也就是说，Java Applet 的字节码文件会下载到客户端，由客户端的浏览器负责运行。浏览器内置的 JVM 创建负责创建主类的对象，该对象立刻调用 init()方法完成必要的初始化工作。初始化的主要任务是创建所需要的对象、设置初始状态、装载图像、设置参数等。该对象仅接着自动调用 start()方法。在程序的执行过程中，init()方法只被调用执行一次

但 start()方法将多次被自动调用执行。除了进入执行过程时调用方法 start()外，当用户从 JavaApplet 所在的 Web 页面转到其他页面，然后又返回时，start()将再次被调用，但不再调用 init()方法。当浏览器离开 Java Applet 所在的页面转到其他页面时，主类创建的对象将调用 stop()方法。如果浏览器又回到此页，则 start()又被调用。在 Java Applet 的生命周期中，start 和 stop()方法可能被调用多次，但 init()方法只被调用一次。当用户关闭浏览器结束浏览时，主类创建的对象自动执行 destroy()方法，结束 Java Applet 的生命。paint(Graphics g)方法可以使一个 Java Applet 在容器上显示某些信息，如文字、色彩、背景或图像等。在 Java Applet 的生命周期内可以多次调用。例如，当 Java Applet 被其他页面遮挡，然后又重新放到最前面、改变浏览器窗口的大小以及 Java Applet 本身需要显示信息时，主类创建的对象都会自动调用 paint()方法。



### 3.9.4 实验内容

#### 题目 1 播放声音

通过 Java Applet 播放多个音频文件，要求用 Chioce 类创建一个选择控制组件，将声音文件放在控件的选择列表中，当客户选择列表中一个项目后，就启动一个创建音频对象的线程。另外，含有 Java Applet 超文本中使用若干个<Param···>标志把指传递到 Java Applet 中。下面是为运行 Java Applet 程序的 Html 文件：

```
<applet code =PlayAudioClip.class width=200 height=200>
<Param name="1" value="祝酒歌： 1.au">
<Param name="2" value="云的思念： 2.au">
<Param name="3" value="祝你平安： 3.au">
<Param name="4" value="难忘今宵： 4.au">
<Param name="5" value="">
<Param name="总数" value="">
</applet>
```

程序运行的效果如图 3-9 所示。



图 3-9 程序运行效果

#### [程序模版]

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class PlayAudioClip extends Applet implements ActionListener,Runnable,ItemListener{
    AudioClip clip;
    Choice choice;
    TextField text;
    Thread thread;
    String item=null;
    Button button_play,button_loop,button_stop;
    public void init(){
        choice=new Choice();
        thread=new Thread(this);
        int N=Integer.parseInt(getParameter("总数"));
        for(int i=1;i<=N;i++){
            choice.add(getParameter(String.valueOf(i)));
        }
        button_play=new Button("开始播放");
        button_loop=new Button("循环播放");
        button_stop=new Button("停止播放");
        text=new TextField(12);
        button_play.addActionListener(this);
        button_stop.addActionListener(this);
        button_loop.addActionListener(this);
        choice.addItemListener(this);
        add(choice);
        add(button_play);
        add(button_loop);
```

```
        add(button_stop);
        add(text);
        button_play.setEnabled(false);
        button_loop.setEnabled(false);
    }
    public void itemStateChanged(ItemEvent e) {
        item=choice.getSelectedItem();
        int index=item.indexOf(":");
        item=item.substring(index+1).trim();
        if(!(thread.isAlive())){
            thread=new Thread(this);
        }
        try {
            thread.start();
        }
        catch(Exception exp) {
            text.setText("正在下载音频文件");
        }
    }
}
public void stop(){
    【代码 1】        // clip 停止播放
}
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==button_play) {
        【代码 2】    // clip 开始播放，但不循环播放
    }
    else if(e.getSource()==button_loop) {
        【代码 3】    // clip 开始播放，并且循环播放
    }
    else if(e.getSource()==button_stop) {
        【代码 4】    // clip 停止播放
        button_play.setEnabled(false);
        button_loop.setEnabled(false);
    }
}
public void run() {
    clip=getAudioClip(getCodeBase(),item);
    text.setText("请稍等...");
    if(clip!=null) {
        button_play.setEnabled(true);
        button_loop.setEnabled(true);
        text.setText("您可以播放了");
    }
}
}
```

**题目 2 Applet 显示图形**

使用 Applet 类提供的 `getImage()` 方法将网络图形文件加载到 Applet 小程序中，通过 `Graphics` 类中 `drawImage()` 方法将它显示在浏览器中。要求完成 java 程序并编写 html 文件。

**[程序模版]**

```
import java.applet.*;
import java.awt.*;
public class DisplayImage extends Applet{
    Image image;
    public void init(){
        image=getImage(getCodeBase(),"Image.gif");
    }
    public void paint(Graphics g) {
        int w=【代码 1】 // 获取原图像宽度
        int h=【代码 2】 // 获取原图像高度
        【代码 3】 // 显示图像
        【代码 4】 // 显示缩小 1 半图像
        【代码 5】 // 显示扩大 2 倍图像
        【代码 6】 // 显示扩大 3 倍图像
    }
}
```

**[思考与扩展]**

1. 题目二中方法 `getCodeBase()` 的作用是什么？
2. Java Application 程序与 Java Applet 程序的不同之处有那些？
3. 构造函数和 `init()` 方法谁先被执行？
4. 编写 Applet，包含两个按钮，一个按钮用于放大 Applet 上的一串字符串，一个按钮用于缩小；连续点击可不断放大或缩小。

## 3.10 设计型实验

### 3.10.1 题目一 算术计算器

#### 【问题描述】

我们在日常生活中经常使用计算器，能买到的电子计算器也种类繁多。本题就是要模拟电子计算器的一些基本功能，实现算术计算。

#### 【基本要求】

本题使用 Swing 的图形界面实现计算器的按键、显示功能，主要实现实数的加减乘除的连续顺序计算（比如：连续在界面上点击“1”“+”“3”“\*”“5”“=”，则输出结果为 20）以及开方、清零等功能。另外，提供退出按钮供用户退出，或者用户可以点击由上角的关闭符号退出。

#### 【测试数据】

测试数据可以有以下几类：

1. 正常数据：正负整数的加减乘除计算，正负小数的加减乘除计算，正数的开方等
2. 边界数据：有 0 参与的加减乘除计算，大数减小数等
3. 错误数据：0 作除数，负数开方等

#### 【实现提示】

使用较为复杂的布局管理（盒式或边界布局+网格布局），可以考虑面板嵌套的设计，尽量做出较好的界面。对于连续顺序计算时，随加减乘除号的点击就开始计算，最后点击等号时显示结果。对于四则运算，可以使用堆栈或多个向量来完成计算。另外，可以提供一些图形用户界面的提示信息，增加界面友好性。

#### 【选做内容】

1. 增加更多的功能
2. 把加减乘除的连续顺序计算改为加减乘除的四则运算

### 3.10.2 题目二 日历查看器

#### 【问题描述】

日历是经常使用的工具，电子日历容量更大，能方便地查到前后几十年甚至更多年份的日历情况。本设计就是要使用图形设计，模拟日历本的界面，创建一个日历查看器，可以查看 100 年的日历情况。

#### 【基本要求】

用户通过下拉列表选择年份和月份，并使用表格显出该年该月下每个日期及其对应的星期几，其中，年份和月份的选择可以独立，也就是说，每选一个，日历就会发生相应的变化，显示相应的月份的情况。

日历反映公历和星期信息。程序运行起来，默认（用户没有选择年份和月份）显示当前的年份和月份。星期六和星期日用红色底的表格标出，以符合人们的使用习惯。

#### 【测试数据】

自行设计。

#### 【实现提示】

使用 Swing 组件完成界面设计，具体建议是使用表格显示日历主体内容，下拉列表组件实现年份和月份的选择。使用 ItemEvent 事件类和 ItemListener 接口，采用匿名类方式完成事件处理。

#### 【选做内容】

把当前日期高亮显示，增加用户手动输入任意年月查询功能。

### 3.10.3 题目三 简单画图程序

#### 【问题描述】

Windows 的画图程序、AutoCAD 等都是典型的画图程序，本题就是要模拟这些典型画图程序的一些基本功能，实现一个简单的能使用鼠标画出多种图形（线段、圆、椭圆、矩形等），有色彩功能的画图程序。

#### 【基本要求】

本题主要使用 Swing 的图形界面实现简单画图程序。主要完成菜单（包含四个菜单：选择图形、选择颜色、清除、退出）、图形绘制、坐标的实时显示、当前做操作的名称显示等功能。在实现菜单功能的同时，在界面上提供基本图形和常用颜色的快捷选择功能（即，鼠标点击后就可以直接在界面上画图，而不用打开菜单。）。)

#### 【测试数据】

自行设计测试数据和方法，主要测试鼠标能不能按照菜单或快捷选择的图形和颜色正确画出图形，同时看坐标是否能正确地实时显示。

#### 【实现提示】

使用较为复杂的布局管理（边界布局+面板嵌套）设计，尽量做出较好的界面。使用 Graphics 类中的相关画图方法，同时结合 Point 类、Color 等类实现画图功能。另外，可以提供一些图形用户界面的提示信息，增加界面友好性。

#### 【选做内容】

1. 增加更多图形的绘制功能
2. 使用文件保存功能

### 3.10.4 题目四 简单文本编辑器

#### 【问题描述】

模拟记事本程序，创建一个简单的文本编辑器，包含一些基本功能，如：文件打开、保存、关闭，文本的复制、粘贴等。

#### 【基本要求】

可打开文件对话框选择一个文件，并在文本区进行编辑，然后把它保存起来，能够关闭和退出。参考界面如图 3-21 所示。



图 3-21 文本编辑器

**【测试数据】**

自行设计。

**【实现提示】**

使用 Swing 组件完成界面设计，采用菜单实现功能选择，主要采用 `ActionEvent` 事件类和 `ActionListener` 接口实现事件处理。文本编辑区使用 `EditorPanel` 或者 `TextArea` 实现。

**【选做内容】**

实现字体大小和颜色的使用，并能够保存相应的值，打开新文件时可以使用这些值。

### 3.10.5 题目五 商店购物结算器

**【问题描述】**

商店在结账时常使用结算工具，本题就是设计一个程序，模拟购物结算的过程，实现结算功能。

**【基本要求】**

由收银员输入购买记录，单击“添加”按钮保存购买记录，显示所有记录；可以按记录号删除记录，并可给出购买物品总额，并可清空所有记录。

编写浏览Applet 的页面文件，在浏览器运行结果如图3-22所示。



图3-22

**【测试数据】**

自行设计。

**【实现提示】**

使用 Swing 设计图形用户界面，完成各个组件的事件处理。使用流式布局管理器布局各个组件、容器。完成购买商品的添加、删除以及购买数量的输入，完成应付款额的计算。对于添加、删除商品及其数量、应付的款额等消息均显示在文本框内，供收银员的检查和客户的查看。设计时可以使用了内部类进行事件处理，使用向量记录所购物品及价格，方便计算。运行程序后，

**【选做内容】**

输入品名编号之后立即可以查询处单价，便于计算总价。

## 3.11 综合型实验

### 3.11.1 题目一 学生成绩管理系统的设计

#### 【问题描述】

某地一大学的期末考试快临近了，教务处希望记录学生的成绩成后，可以立即输出成绩单。但是，有些已经选修课程的学生尚未支付他们的选课费。

如果某个学生的选课费已支付，他的成绩将会与平均学分绩点一起显示在学习成绩单上。

如果某个学生尚未支付学费，则不会输出成绩。对于这些学生，成绩单将会包含一则信息，指出由于未支付学费的原因成绩暂时扣留。同时，成绩单也将显示应支付的学费总额。

教务处想让你帮他们编写一个程序，该程序可以分析学生的数据并输出相应的成绩单。

#### 【基本要求】

本题目要求使用数据库存储需要管理的数据，使用 Swing 创建显示学生的成绩单的图形用户界面，该程序在图形界面中查询学生的成绩及相关信息，按“问题描述”部分的要求生成成绩单。

查询学生成绩信息并合理显示是主要功能，查询主要包括：对特定学号学生成绩情况的查询，对特定姓名学生成绩情况的查询，对某个班级学生成绩情况的查询和统计，对某个专业学生成绩情况的查询和统计。

如果某个学生尚未支付学费，则不会输出成绩，成绩单界面上将会显示提示信息，指出由于未支付学费的原因成绩暂时不能统计，同时显示应支付的学费总额。

#### 【测试数据】

测试数据可以有以下几类：

1. 正常数据：正确的学号、姓名、班级、专业等字符
2. 边界数据：不输入任何数据，输入全 0 或 1 等
3. 错误数据：输入超过学号位数的数据，随意输入汉字，输入带空格的字符串等

#### 【实现提示】

程序设计过程提示：

1. 应仔细分析题目的问题描述和要求，得出功能点以及这些功能点之间的联系，得出需要存储在数据库的数据信息（建议写成数据字典）；
2. 根据存储数据的特点和程序要求，识别出实体以及他们之间的联系，然后设计数据库，所设计的数据库应符合数据库设计的规范；
3. 根据面向对象的思想，进行类的设计（主要包括类的识别、属性和行为的设计）；
4. 设计合理的流程逻辑（即，问题的处理流程）；
5. 根据上述工作的结果，进行编码；
6. 设计合理的测试数据，并进行程序的测试。

每个学校包含很多学生，每位学生都要选修课程。因此，学生成绩和课程的信息是两个主要组成部分。对课程（Course）信息的说明：课程的主要属性可以是课程编号、课程名称、学分数、所属专业等。对学生成绩（StudentScore）信息部分的说明：学生成绩的主要属性可以是学号、姓名、选修的课程数目、选修的课程编号、每门课程的成绩、平均学分积点以及是否缴纳学费等。

使用数据库（MS-Accesse、MS-SqlServer 或 MySQL）存储需要管理的学生成绩情况数据，使用 ODBC-JDBC 桥接器访问数据库，使用 Swing 设计图形用户界面，使用菜单和文本框、按钮来完成相应的操作，提供退出按钮使用 Action 事件供用户退出，或者用户可以点击由上角的关闭符号，使用 Window 事件退出。

平均学分绩点 =  $\sum(\text{课程绩点} \times \text{课程学分}) / \sum \text{课程学分}$

**【提高要求】**

1. 使用 JDBC 访问数据库数据
2. 增加学生成绩录入功能
3. 增加对专业、院系的信息管理功能
4. 增加用户管理功能，即程序的使用者需要登录才能进行相关操作

**3.11.2 题目二 Socket 编程实现网络聊天室****【问题描述】**

通过 Socket 方式进行连接一个聊天程序，在服务器端指定端口上建立监听机制，监听客户端 Applet 小程序的请求，每个客户 Applet 都需要同服务器的监听管理程序建立通讯，通过服务器端程序将每个客户发送的信息发送给相应的在线客户。

**【基本要求】**

1. 实验的基本参考演示界面

客户端 Applet 小程序界面如图 3-23 所示。

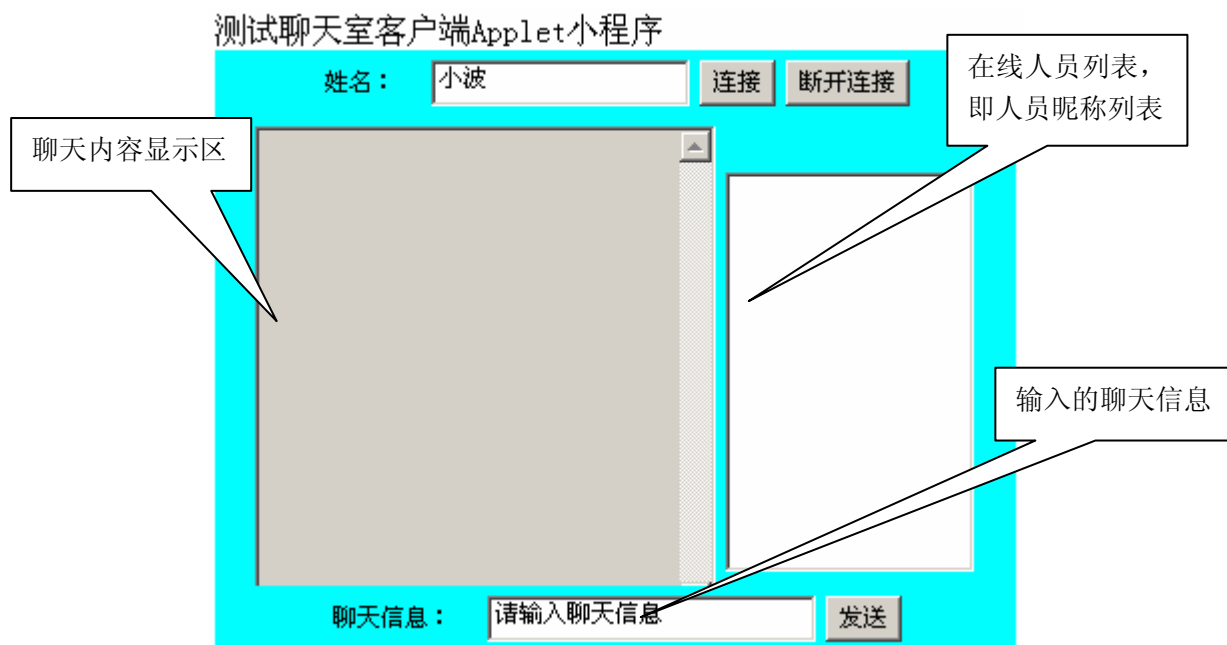


图 3-23

服务器端运行的程序，监听连接，并输出结果，如图 3-24 所示。

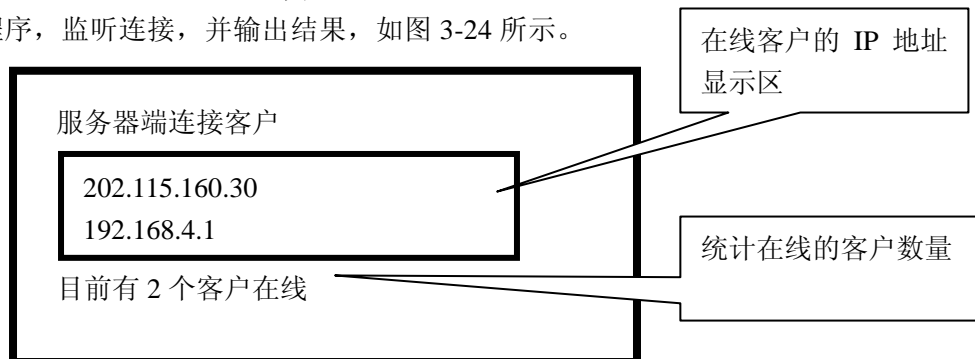


图 3-24

2. 多线程监听，即能够实现多个客户端的聊天。
3. 程序界面要求尽量美观，布局符合实际应用习惯。

**【测试数据】**



自行设计。要求在局域网中模拟实验环境

**【实现提示】**

程序设计过程提示：

1. 应仔细分析题目的问题描述和要求，得出功能点以及这些功能点之间的联系，找出载程序运行过程中需要的和产生的数据；
2. 根据数据的特点和程序要求，设计出数据的存储（可以是）；
3. 根据面向对象的思想，进行类的设计（主要包括类的识别、属性和行为的设计）；
4. 设计合理的流程逻辑（即，问题的处理流程）；
5. 根据上述工作的结果，进行编码；
6. 设计合理的测试数据，并进行程序的测试。

使用 TCP 编程，创建客户端对象的 `Socket` 类，创建服务器端对象使用 `ServerSocket` 类，设计适当流对象和方法发送信息。建议使用 `DataInputStream` 类和 `DataOutputStream` 类构造输入输出流。

使用 `Swing` 设计图形用户界面，完成相应组件的事件处理。提供退出按钮，使用 `Action` 事件供用户退出，或者用户可以点击由上角的关闭符号，使用 `Window` 事件退出。

实例的界面布局仅做参考，学生可以根据自己的情况设计界面，要求客户端界面尽量美化，功能实用，符合用户习惯。服务器端程序要记录在线用户（可以使用向量来实现），并在客户端显示出来。

客户端使用 `Applet` 实现，服务器端使用 `Application` 实现。

**【提高要求】**

1. 增加用户注册和登录，增加用户头像，增加聊天的背景音乐
2. 实现一对一的私下聊天（其他人看不见）
3. 使用数据库存储用户注册信息、头像等信息

## 附录 A Java 编码规范

### 1. Java 命名约定

#### 1.1 Java 标识符

Java 标识符是用来表示变量名、方法名、类名等的有效字符(Unicode,16 位)序列,定义的规则如下:

- ✧ 由字母、数字、下划线和美元符组成
- ✧ 第一个字符不能使数字
- ✧ 严格区分大小写,没有长度限制

Java 命名时应始终采用比较完整的英文描述符,即命名有一定的含义,尽量做到望名知意。一般应采用小写字母,但类名、接口名以及任何非初始单词的第一个字母要大写。

#### 1.2 一般性约定

在 Java 命名过程中应遵循以下约定:

- ✧ 尽量使用完整的英文描述符
- ✧ 采用适用于相关领域的术语
- ✧ 采用大小写混合使名字可读
- ✧ 尽量少用缩写,但如果用了,要明智地使用,且在整个工程中统一
- ✧ 避免使用长的名字(小于 15 个字符)
- ✧ 避免使用类似的名字,或者仅仅是大小写不同的名字
- ✧ 避免使用下划线(除静态常量等)

#### 1.3 示范

**包(Package)** 采用完整的英文描述符,应该都是由小写字母组成,如: java.awt。对于全局包,将你的 Internet 域名反转并接上包名,如: com.ambyssoft.www.persistence。

**类(Class)** 采用完整的英文描述符,所有单词的第一个字母大写,如: Customer, SavingsAccount。

**接口(Interface)** 采用完整的英文描述符说明接口封装,所有单词的第一个字母大写,与类相同。习惯上,名字后面加上后缀 able, ible 或者 er,但这不是必需的,如: Contactable,Prompter。

**组件/部件(Component)** 使用完整的英文描述来说明组件的用途,末端应接上组件类型,如: okButton, customerList, fileMenu。

**异常(Exception)** 通常采用字母 e 表示异常对象。对于自定义的异常,也应该以 Exception 结尾。

**类变量** 采用完整的英文描述,第一个字母小写,任何中间单词的首字母大写,如: firstName, lastName。

**获取型方法** 被访问属性的前面加上前缀 get,如: getFirstName(), getLastName()。

**布尔型方法** 所有的布尔型方法用单词 is 做前缀,如 isPersistent(), isString()。

**设置型方法** 被访问属性的前面加上前缀 set,如: setFirstName(), setLastName(), setWarpSpeed()。

**成员方法** 采用完整的英文描述说明该方法的功能,一般是动宾结构,第一个单词尽可能采用一个生动(能表达该方法的动作)的动词,第一个字母小写,如: openFile(), addAccount()。

**静态常量(static final)** 全部采用大写字母,单词之间用下划线分隔,如: MIN\_BALANCE, DEFAULT\_DATE。

**循环计数器** 通常采用字母 i, j, k 或者 counter 都可以接受。

**数组** 数组应该总是用下面的方式来命名: objectType[] theobject, byte[] buffer。

### 2. 有关注释的约定

在程序编写过程中使用注释能很好地提高程序的可读性与可理解性。一个很好的可遵循的有关注释的经验法则是：问问你自己，你如果从未见过这段代码，要在合理的时间内有效地明白这段代码，你需要哪些信息。

## 2.1 一般性约定

在注释的书写中，一般应遵循以下约定：

- ✧ 注释应该增加代码的清晰度
- ✧ 保持注释的简洁
- ✧ 在写代码之前写注释
- ✧ 注释出为什么做了一些事，而不仅仅是做了什么

## 2.2 示范

**文档注释** 在紧靠接口、类、成员方法和属性声明的前面注释它们，或者在一个源文件（通常是一个类）的开始位置书写文档注释。如：`/** 客户：客户是我们将服务和产品卖给的人或机构。*/`

**多行注释** 多行注释去掉不再使用但你仍想保留的代码。仍想保留是因为用户万一会改变想法，或者在调试过程中想让它暂时失效。如：`/* 这部分代码因为已被它之前的代码取代，由 B.Gustafsson, 于 1999 年 6 月 4 日注释掉。如果一年之后还未使用，将其删除。...（源代码）*/`

**单行注释** 在成员方法内采用单行注释，来说明业务逻辑、代码段和局部变量的声明。注释符`/**`后必须紧跟一个空格，然后才是注释信息。如：`// sum 清零。`

## 2.3.哪些部分需要注释

在编写程序时，并不是每个地方、每行都要写注释，要根据具体的情况来判断，应注意以下情况需要写注释：

**类** 类的目的、即类所完成的功能，还要注释出采用的不变量，如果是一个类文件，则要在源文件的开始注释出编写者、时间、版权等信息。

**接口** 设置接口的目的、它应如何被使用以及如何不被使用。

**成员方法** 对于设置与获取成员方法，在属性已有说明的情况下，可以不加注释；普通成员方法要求说明完成什么功能，参数含义是什么，返回什么。

**普通成员方法内部注释** 主要有：控制结构，代码做了些什么以及为什么这样做，处理顺序等。

**参数** 参数含义、及其它任何约束或前提条件。

**属性** 属性描述。

**局部变量** 无特别意义的情况下可以不加注释。

## 3. Java 源文件样式约定

所有的 Java(\*.java) 源文件都必须遵守如下的样式（书写顺序）约定：

### 1) 版权信息

版权信息必须在 java 源文件的开头，比如：

```
/** Copyright@2000 Shanghai XXX Co. Ltd. All right reserved. */
```

其他不需要出现在 javadoc 的信息也可以包含在这里。

### 2) Package/Imports

package 行要在 import 行之前，import 中标准的包名要在本地的包名之前，而且按照字母顺序排列。如果 import 行中包含了同一个包中的不同子目录，则应该用\*来处理。如：

```
package hotlava.net.stats;import java.io.*;
import java.util.Observable;
import hotlava.util.Application;
```

### 3) Class

接下来的是类的注释，一般是用来解释类的。

```
/** A class representing a set of packet and byte counters. It is observable to allow it to be watched,
but only reports changes when the current set is complete */
```

接下来是类定义，包含了在不同的行的 `extends` 和 `implements`，如：

```
public class CounterSet extends Observable implements Cloneable{
    .....
}
```

以下均是在类体里面书写

#### 4) main() 方法

主类(原文件中由 `public` 修饰的 `class`)需要包含 `main()`方法，普通类不需要 `main()`方法，如果有 `main()`方法，那么它应该写在类的前部。并且参数最好写成：`String[] args` 的形式。

#### 5) 构造方法

接下来是构造方法，它应该用递增的重载方式写（参数多的写在后面）。如：

```
public CounterSet(int size){
    this.size = size;
}
```

#### 6) 存取方法（类的设置与获取成员方法）

接下来是类的属性的存取的方法。如类的成员变量已经有注释，类变量的存取方法可以没有注释。如：

```
public int[] getPackets(){
    return this.packets;
}
```

#### 7) 克隆方法

如果这个类是可以被克隆的，那么下一步就是 `clone` 方法：

```
public Object clone() {
    try {
        .....
    }catch(CloneNotSupportedException e) { ..... }
}
```

#### 8)类的普通成员方法

接下来写普通的方法，如：

```
public void throwException() throws MyException {
    throw new MyException("Originated in throwException()"); // 抛出自定义异常
}
```

#### 9) toString 方法

接下来定义 `toString` 方法：

```
public String toString() {
    .....
}
```

#### 10) 成员属性

接下来是类的成员属性，如：

```
int sum;
```

`public` 的成员变量必须生成文档 (JavaDoc)。`protected`、`private` 和 `package` 定义的成员变量如果名字含义明确的话，可以没有注释。同时应注意常量定义在变量定义之前。

### 4. Java 编码其它约定

## 4.1 文档化

必须用 javadoc 来为类生成文档。不仅因为它是标准，这也是被各种 Java 编译器都认可的方法。使用 @author 标记是不被推荐的，因为代码不应该是被个人拥有的。

## 4.2 缩进

缩进应该是每行 2 个空格。不要在源文件中保存 Tab 字符，在使用不同的源代码管理工具时 Tab 字符将因为用户设置的不同而扩展为不同的宽度。

如果你使用 UltrEdit 作为你的 Java 源代码编辑器的话，你可以通过如下操作来禁止保存 Tab 字符：通过 UltrEdit 中先设定 Tab 使用的长度为 2 个空格，然后用 Format|Tabs to Spaces 菜单将 Tab 转换为空格。

## 4.3 页宽

页宽应该设置为 80 字符。源代码一般不会超过这个宽度，但这一设置也可以灵活调整。在任何情况下，超长的语句应该在一个逗号或者一个操作符后换行。一条语句换行后，应该比原来的语句再缩进 2 个字符。

## 4.4 {} 对

{ } 中的语句应该单独作为一行。例如，下面的第 1 行是错误的，第 2 行是正确的：

```
if (i>0) { i ++ }; // 错误, { 和 } 在同一行
if (i>0) {
    i ++; // 正确, 单独作为一行
}
```

## 4.5 括号

左括号和后一个字符之间不应该出现空格；同样，右括号和前一个字符之间也不应该出现空格。如：

```
CallProc( AParameter ); // 错误
CallProc(AParameter); // 正确
```

不要在语句中使用无意义的括号，括号只应该为达到某种目的而出现在源代码中。

# 5. 一些编程建议

## 5.1 使用 StringBuffer 对象

在处理 String 的时候要尽量使用 StringBuffer 类，StringBuffer 类是构成 String 类的基础。String 类将 StringBuffer 类封装了起来，（以花费更多时间为代价）为开发人员提供了一个安全的接口。当我们在构造字符串的时候，我们应该用 StringBuffer 来实现大部分的工作，当工作完成后将 StringBuffer 对象再转换为需要的 String 对象。比如：如果有一个字符串必须不断地在其后添加许多字符来完成构造，那么我们应该使用 StringBuffer 对象和它的 append() 方法。如果我们用 String 对象代替 StringBuffer 对象的话，会花费许多不必要的创建和释放对象的 CPU 时间。

## 5.2 避免太多的使用 synchronized 关键字

避免不必要的使用关键字 synchronized，应该在必要的时候再使用它，这是一个避免死锁的好方法。必须使用时，也尽量控制范围，最好在块级控制。

## 5.3 尽量使用接口而不是一个具体的类

假设有如下需求，给定一个 SQL 语句，返回一个对象的列表，实现中用 java.util.ArrayList 实现，于是定义方法为：

```
public java.util.ArrayList getObjectItems(String sql)
```

上面的方法存在一个问题，当 getObjectItems 内改用 Vector 或 LinkedList 实现，外部类必须做相应更改。一个更好的方法是定义返回值为 java.util.AbstractList 更合适：

```
public java.util.AbstractList getObjectItems(String sql)
```

这样即使更改实现，外部类也不必做相应更改。

#### 5.4 避免使用索引来调用数据库中间层组件返回的结果集

如: `for(int i=1; i<=dt.getRowCount(); i++){ String field1 = dt.getField(i, 0).toString(); .....}`

而应用字段名来存取结果集:

`for(int i=1; i<=dt.getRowCount(); i++){ String field1 = dt.getField(i, "field1").toString(); .....}`

这样在数据库设计更改或查询的 SQL 语句发生变化时, 不会影响到程序的执行。

## 附录 B NetBeans 集成开发环境

NetBeans IDE 是为软件开发者提供的一个免费、开放源代码的集成开发环境（IDE）。它可以在多种平台上运行，其中包括 Windows、Linux、Solaris 和 MacOS。NetBeans IDE 易于安装和使用。它为开发者创建专业的跨平台桌面、企业、Web 和 Mobile 应用程序提供了所需的全部工具。

### NetBeans 的下载

NetBeans IDE 可以从 Sun 网站上下载，网址如下：

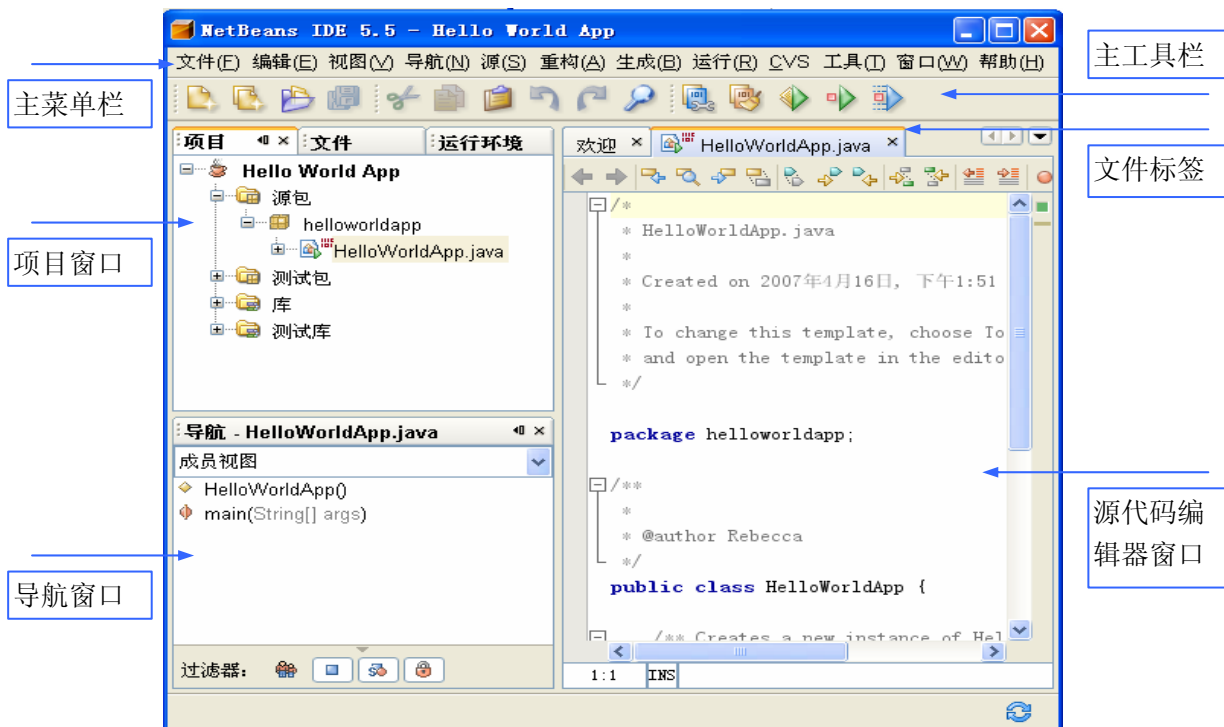
1. <http://www.netbeans.info/downloads/index.php>（NetBeans 官方网站）
2. <http://gceclub.sun.com.cn/download.html>（Sun 中国技术社区）

目前最新正式版本为 V5.5，在下载时注意选择中文版本进行下载。

注意：NetBeans5.5 的安装和使用需要 J2SE(TM) Development Kit (JDK)版本 5.0 以上。

### NetBeans 集成开发环境简介

NetBeans 集成开发环境是一个使用单窗口界面，集编辑、管理文件与工程、可视界面设计、浏览、编译、调试和其他操作等多功能于一体的开发平台。NetBeans 集成开发环境如图附 B-1 所示。



图附 B-1 NetBeans 集成开发环境图

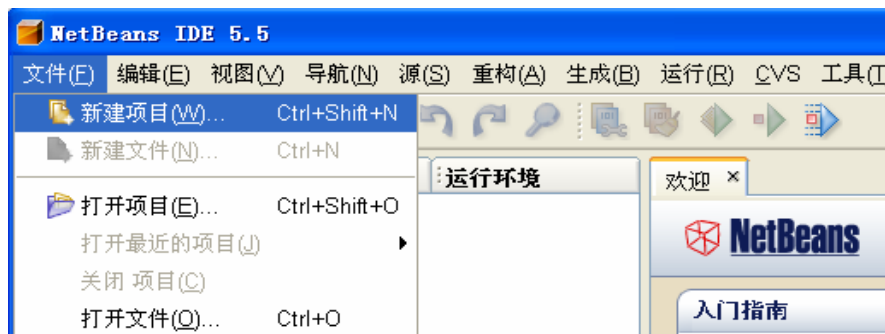
常用窗口简介如下：

- ✧ 主菜单栏，包含了几乎所有的操作菜单，NetBeans 所有的功能都可以在菜单上实现；
- ✧ 主工具栏，包含了常用的操作，避免频繁使用菜单，使用更简便；
- ✧ 文件标签，在 IDE 中打开的文件都将体现在文件标签中；
- ✧ 项目窗口，包含了该项目的组件树视图，其中含有源文件、代码依赖的库等；
- ✧ 导航窗口，使用该窗口在选定类中的元素之间进行快速导航；
- ✧ 源代码编辑器窗口（在此窗口中打开了一个名为 HelloWorldApp 的文件），用来编写源代码。

在全中文状态下编辑环境的使用是比较方便的。下面将介绍编辑环境的使用。

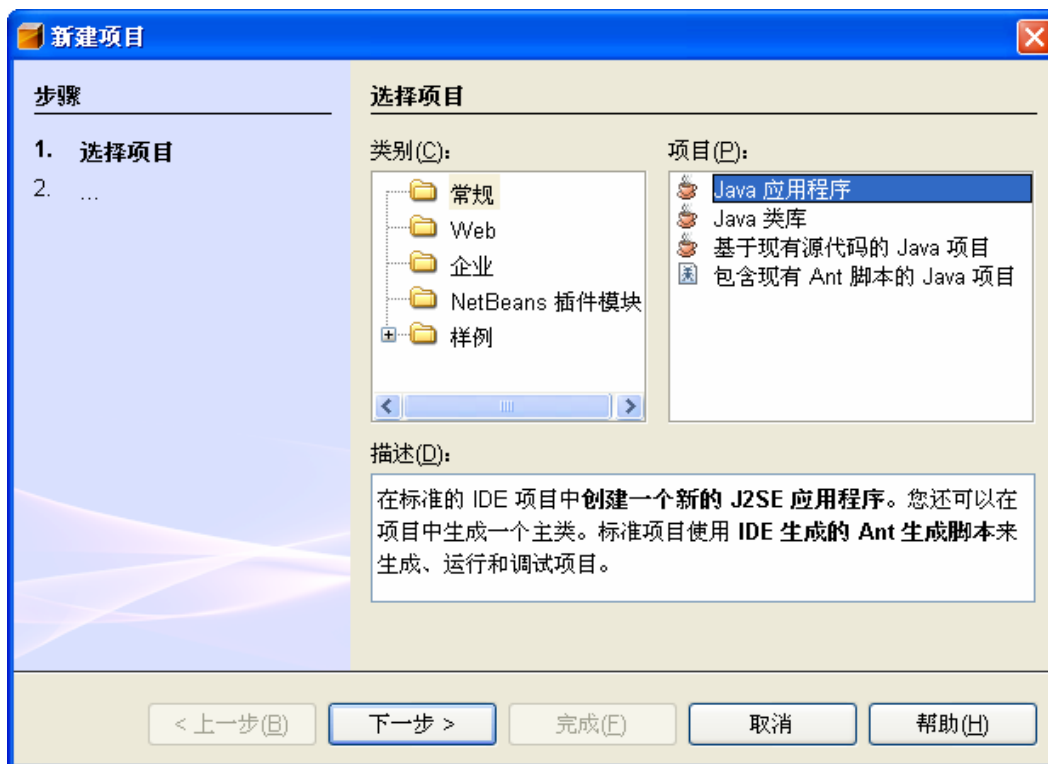
## 创建 IDE 项目

1. 启动 NetBeans IDE。
2. 在 IDE 中，选择“文件”>“新建项目”，如图附 B-2 所示。



图附 B-2 新建项目菜单示意图

3. 在“新建项目”向导中，展开“常规”类别，然后选择“Java 应用程序”，如图附 B-3 所示。然后，单击“下一步”。



图附 B-3 选择项目示意图

4. 在向导的“名称和位置”页中，执行以下操作，如图附 B-4 所示：
  - ✧ 在“项目名称”字段中，键入 Hello World App;
  - ✧ 在“创建主类”字段中，键入 helloworldapp.HelloWorldApp;
  - ✧ 将“设置为主项目”复选框保留为选中状态。





图附 B-4 设置项目名称和位置示意图

5. 单击“完成”，这样就建立了一个项目。在下次打开 IDE 时，就会看到如图附 B-1 所示的界面（默认情况下，上次编辑过的项目，在下次打开 IDE 时，会出现在当前窗口中。）。

#### 在生成的源文件中添加代码

如果在“新建项目”向导中将“创建主类”复选框保留为选中状态，那么 IDE 会自动创建了一个框架类。编码时，只需要在框架中空白的地方加上适当的代码即可。比如，此处可在 `main()` 方法里面加上代码：

```
System.out.println("Hello World!");
```

即完成一个简单的 Java 程序“HelloWorld.java”。

#### 编译源文件

要编译源文件，则从 IDE 的主菜单中选择“生成”>“生成主项目”。编译时将打开“输出”窗口，显示与图附 B-5 类似的输出内容。

```
输出 - Hello_World_App (jar)
init:
deps-jar:
Created dir: C:\MyProjects\Hello World App\build\classes
Compiling 1 source file to C:\MyProjects\Hello World App\build\classes
compile:
Created dir: C:\MyProjects\Hello World App\dist
Building jar: C:\MyProjects\Hello World App\dist\Hello_World_App.jar
To run this application from the command line without Ant, try:
java -jar "C:\MyProjects\Hello World App\dist\Hello_World_App.jar"
jar:
生成成功 (总时间: 2 秒) |
```

图附 B-5 编译时输出窗口

如果在生成输出中出现“生成成功”结束语句，则表明已经成功编译了该程序！

如果生成输出中出现“生成失败”结束语句，则表明代码中可能包含语法错误。其中“输出窗口”

报告的错误将以超级链接文本的形式出现。可以双击此类超级链接以导航至错误源代码。然后修复该错误，并再次选择“生成”>“生成主项目”。

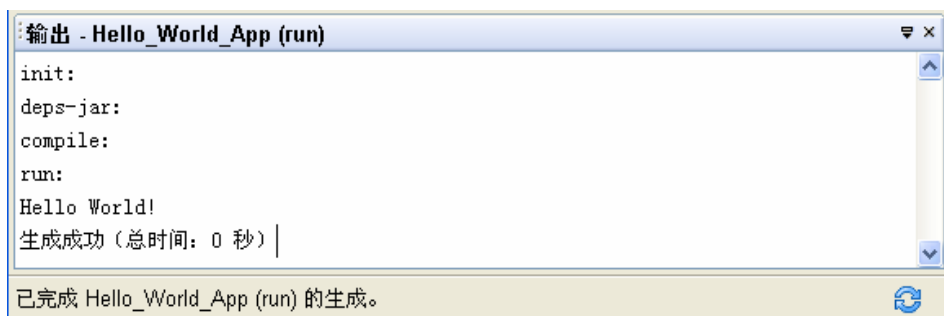
生成项目时，将会生成字节代码文件 `HelloWorldApp.class`，可通过打开“文件”窗口并展开 `Hello World App/build/classes/helloworldapp` 节点来查看生成的新文件，如图附 B-6 所示。



图附 B-6 项目文件结构示意图

### 运行程序

经过前面的编译，就可以开始运行程序了。从 IDE 的菜单栏中，选择“运行”>“运行主项目”。运行的结果将在输出窗口中显示，如图附 B-7 所示。



图附 B-7 运行时输出窗口示意图

### 调试程序

调试是检查应用程序是否存在错误的过程。可以使用以下方法进行调试：在代码中设置断点和监视，然后在调试器中运行代码。还可以通过逐行执行代码并检查应用程序状态来查找任何问题。

#### 1. 基本调试

##### (1) 启动调试会话

在 IDE 中启动调试会话时，IDE 将编译所调试的文件，在调试模式下运行它们，并在调试器窗口中显示调试器输出。要启动调试会话，要选择要调试的文件，然后从“运行”菜单中选择以下某个命令：

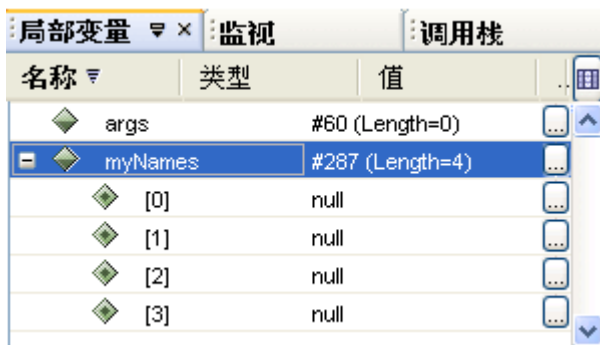
- ✧ 调试主项目 (F5)。运行主项目，直至遇到第一个断点。
- ✧ 步入 (F7)。开始运行主项目的主类，并在第一条可执行语句处停止。
- ✧ 运行至光标 (F4)。启动调试会话，使应用程序运行至源代码编辑器中的光标位置，然后暂停应用程序。

如果在 IDE 中打开了多个项目，则要确保使用以下方法将当前要调试项目的设置为主项目：在“项目”窗口中右键单击 当前要调试项目的节点，然后从上下文菜单中选择“设置主项目”。按 F7 键

步入主项目的主类。如果未设置项目的主类，则 IDE 将提示设置它。然后，IDE 将在源代码编辑器中打开文件，显示输出窗口和调试器窗口，并刚好在 `main` 方法内部停止。

### (2) 调试器窗口

调试器窗口在每次启动调试会话时自动打开，并在完成会话时自动关闭。默认情况下，IDE 将打开三个调试器窗口：“局部变量”窗口、“监视”窗口和“调用栈”窗口，如图附 B-8 所示。



图附 B-8 “局部变量”窗口位于前端的调试器窗口

通过从“窗口”>“调试”菜单中进行选择，可以打开其他调试器窗口。如果在调试会话过程中打开了调试器窗口，该窗口将在完成会话后自动关闭。如果在未启动调试会话的情况下打开调试器窗口，该窗口将一直保持打开状态，直到手动关闭它。也可以通过将调试器窗口拖动到所需位置来对其进行排列。

### (3) 逐步执行代码

可以使用“运行”菜单中的以下命令来控制代码在调试器中的执行方式：

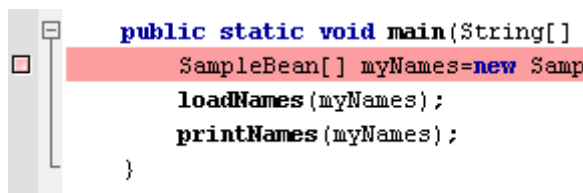
- ✧ 越过 (F8)。执行一行源代码。如果源代码行包含调用，则执行整个例程而不逐步执行单个指令；
- ✧ 步入 (F7)。执行一行源代码。如果源代码行包含调用，则会刚好在执行例程的第一条语句之前停止；
- ✧ 步出 (Alt-Shift-F7)。执行一行源代码。如果源代码行是某个例程的一部分，则会执行该例程的其余各行，然后将控制权返回给例程的调用者；
- ✧ 暂停。暂停执行应用程序；
- ✧ 继续 (Ctrl-F5)。继续执行应用程序。应用程序将在下一个断点处停止；
- ✧ 运行至光标 (F4)。将当前会话运行到源代码编辑器中的光标位置，并暂停应用程序。

## 2. 使用断点

大多数应用程序都太大，从而无法逐行进行检查。通常，在认为会出现问题的位置设置一个断点，然后将应用程序运行到该位置。还可以设置更为专用的断点（如仅当指定条件为 `true` 时停止执行的条件断点）或适用于某些线程或方法的断点。

### (1) 设置断点

如果仅希望设置简单的行断点，则可以单击所需行的左旁注处。在旁注中将会出现行断点图标 (■)，如图附 B-9 所示。再次单击此行断点可以将其删除。



图附 B-9 源代码编辑器中设置断点示意图

对于更复杂的断点，则使用“运行”菜单中的“新建断点”(Ctrl-Shift-F8) 命令。“新建断点”对话框

框将使您能够选择要创建的断点类型并设置断点选项，如中断条件或断点输出到输出窗口的信息。

#### (2) 设置断点条件

仅当指定的布尔表达式为 `true` 时，条件断点才停止执行。如果要设置条件断点，请打开“新建断点”对话框，然后在“条件”字段中输入表达式。

#### (3) 定制断点输出

在“新建断点”对话框中，可以指定到达断点时打印的信息。在对话框底部的“打印文本”字段中输入任何消息。还可以使用变量引用要显示的某些类型的信息。

#### (4) 断点类型

可用的不同断点类型如表附 B-1 所示。

表附 B-1 断点类型表

类型	描述
行	可以在到达行时或在行中的元素满足某些条件时中断执行。
方法	如果在方法名称上设置了断点，则在每次执行该方法时都会停止执行应用程序。
异常	具有多个用于设置异常断点的选项。在捕获到特定异常、源代码中未处理特定异常或遇到任何异常（无论应用程序是否处理错误）时都可以中断执行应用程序。
变量	可以在访问（例如，将变量作为参数来调用方法）或修改特定类和字段中的变量时停止应用程序的执行。
线程	可以在启动和/或停止线程时中断执行应用程序。
类	如果设置了类断点，则可以在将类装入虚拟机和/或从虚拟机中卸载类时停止调试器。

### 3. 设置监视

通过使用监视，可以在应用程序执行期间跟踪变量或表达式值的变化。要设置监视，在源代码编辑器中选择要设置监视的变量或表达式，然后单击鼠标右键并选择“新建监视”(Ctrl-Shift-F7)。

还可以在“监视”视图中创建固定监视。常规监视描述的是变量的内容，而固定监视描述的则是当前为变量指定的对象。要创建固定监视，请右键单击“局部变量”或“监视”视图中的任意项，然后选择“创建固定监视”。

## 附录 C 验证型实验报告格式

# Java 程序设计实验报告

实验名称: \_\_\_\_\_

指导教师: \_\_\_\_\_

专业班级: \_\_\_\_\_

姓 名: \_\_\_\_\_

学 号: \_\_\_\_\_

电子邮件: \_\_\_\_\_

实验日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

报告日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

成绩: \_\_\_\_\_

## 一、实验目的

（目的要明确，在理论上验证 Java 的语言基础和语言使用，并使实验者获得深刻和系统的理解；在实践上，掌握使用实验设备、实验环境的技能技巧以及程序的调试方法。可以着重参考实验指导书）

## 二、实验过程

（这是实验报告极其重要的内容。要抓住重点，可以从理论和实践两个方面考虑。这部分要写明依据什么知识点以及使用方法进行实验以及实验步骤。不要简单照抄实习指导，更不可写一大堆源代码）

## 三、实验结果

（应先列出测试数据，要写明实验的现象，实验数据的处理等。对于实验结果的表述，一般有三种方法：文字叙述（根据实验目的将实验结果系统化、条理化，用准确的专业术语客观地描述实验现象和结果，要有时间顺序以及各项指标在时间上的关系），图表（用表格或坐标图的方式使实验结果突出、清晰，便于相互比较，尤其适合于分组较多，且各组观察指标一致的实验，使组间异同一目了然；每一图表应有表目和计量单位，应说明一定的中心问题），屏幕截图（实验结果也可以是屏幕截图，充分表明实验的实际情况）。在实验报告中，可任选其中一种或几种方法并用，以获得最佳效果。）

## 四、讨论与分析

（这个为主要部分。根据相关的知识点以及编程规范和经验对所得到的实验结果进行解释和分析。如果所得到的实验结果和预期的结果一致，那么它可以验证什么知识点，可以验证语言的什么使用方法？实验结果有什么意义？说明了什么问题？这些是实验报告应该讨论的；如果所得到的实验结果和预期的结果不相符，那么误差在什么地方，是什么原因造成的，准备怎么改进；如果实验根本就进行不下去，那么原因在什么地方。不要简单地复述课本上的理论而缺乏自己主动思考的内容。本部分还应该回答实验指导书的扩展与思考部分提出的问题。）

**五、实验自评**（主要从实验态度、方法、效果上给一个客观公正的自我评价，可以对前面讨论与分析部分，做进一步的主观原因的分析。）

**六、附录：关键代码**（给出适当注释，可读性高）

**【注】：**每个部分要保证层次清晰，逻辑性强，通俗易懂。

## 附录 D 设计型、综合型实验报告格式

# Java 程序设计设计报告

题目名称: \_\_\_\_\_

指导教师: \_\_\_\_\_

专业班级: \_\_\_\_\_

姓 名: \_\_\_\_\_

学 号: \_\_\_\_\_

电子邮件: \_\_\_\_\_

设计时间: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

至 \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

报告日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

成绩: \_\_\_\_\_

目录，每个设计报告都要有目录，示例如图附 D-1 所示。

目 录	
一. 问题描述及设计思路.....	1
1.1 [节标题].....	1
1.1.1.....	
1.1.2.....	
1.2 .....	
1.3 .....	
1.4 .....	
二. 详细设计过程.....	
2.1 .....	
2.2 .....	
2.3 .....	
2.4 .....	
2.5 .....	
三. 结论及体会.....	
3.1 .....	
3.2 .....	
四. 附录 .....	
附录 A.....	
附录 B.....	
五. 参考文献.....	

图附 D-1 参考录目示意图

目录完毕后，要分页。



正文（主要包括问题描述及设计思路、详细设计过程、结论及体会等。格式同目录清单相对应，要求正文部分层次分明，逻辑性强，避免使用口语化词语。详细设计过程不能写程序代码，重点要写方法设计、消息设计，配以相应的图表，必要的地方用程序流程图和伪码描述问题。结论及体会主要有设计结果及分析，设计过程中遇到的关键问题，怎么解决的，通过本次设计的收获等。正文字数不少于 0.3 万字，每一章要分页）

附录，主要显示一些正文中没有出现的图表，主要程序代码等。

参考文献，格式如图附 D-2 所示，要求：五号字，宋体，单倍行距。按作者、书名、出版社、出版时间格式逐一列出

- |     |  |
|-----|--|
| [1] | 耿祥义，张跃平. JAVA 2 实用教程. 第二版. 清华大学出版社, 2004 |
| [2] | .....                                    |

图附 D-2 参考文献格式

上交设计结果和文档的磁盘或光盘要求：

1. 上交前要检查磁盘或光盘是否完好，保证文件的完好性；
2. 文件夹组织（每个学生上交的内容放在一个文件夹里面，文件夹的命名格式是：设计题目-班级-姓名-学号，如算术运算器-软件 0501-李明-20051234。示例如下），如图附 D-3 所示。

设计题目-班级-学生姓名-学号

名称	大小	类型	修改时间
源代码		文件夹	2006-3-21 9:23
其他相关说明文档1	0 KB	文本文档	2006-3-21 9:17
其他相关说明文档2	0 KB	文本文档	2006-3-21 9:18
设计报告-设计题目	11 KB	Microsoft Word 文档	2006-3-21 9:19

图附 D-3 文件夹组织示意图

## 计算机科学与技术学院自编实验指导书

软件项目管理

计算机基本技能训练

计算机网络

计算机组成原理

C++程序语言设计

计算机图形学

编译原理

软件测试技术

汇编语言程序设计

网络程序设计

反病毒技术

Linux 系统及程序设计

Windows 程序设计

软件开发综合实验

Java 程序设计

网络攻防对抗

计算机制图

微机原理及应用

嵌入式系统实验

数据库应用设计

单片机与接口应用设计

信息电器分析与设计

数据结构应用设计

计算机操作系统综合实验

动态网页制作技术

数学实验

面向对象技术应用设计

信息安全综合实验

