

第 1 章

进销存管理系统

(Swing+SQL Server 2000 实现)

实现企业信息化管理是现代社会中小企业稳步发展的必要条件，它可以提高企业的管理水平和工作效率，最大限度地减少手工操作带来的失误。进销存管理系统正是一个信息化管理软件，可以实现企业的进货、销售、库存管理等各项业务的信息化管理。本章将介绍如何使用 Java Swing 技术和 SQL Server 2000 数据库开发跨平台的应用程序。通过阅读本章，可以学习到：

- 如何进行项目的可行性分析
- 如何系统设计
- 如何进行数据库分析和数据库建模
- 企业进销存主要功能模块的开发过程
- 如何设计公共类
- 如何将程序打包

1.1 开发背景

加入 WTO 之后，随着国内经济的高速发展，中小型的商品流通企业越来越多，其所经营的商品种类繁多，难以管理，而进销存管理系统逐渐成为企业经营和管理中的核心环节，也是企业取得效益的关键。×××有限公司是一家以商业经营为主的私有企业，为了完善管理制度，增强企业的竞争力，公司决定开发进销存管理系统，以实现商品管理的信息化。现需要委托其他单位开发一个企业进销存管理系统。

1.2 系统分析

1.2.1 需求分析

通过与×××有限公司的沟通和需求分析，要求系统具有以下功能。

- ☒ 系统操作简单，界面友好。
- ☒ 规范、完善的基础信息设置。
- ☒ 支持多人操作，要求有权限分配功能。
- ☒ 为了方便用户，要求系统支持多条件查询。
- ☒ 对销售信息提供销售排行。
- ☒ 支持销售退货和入库退货功能。
- ☒ 批量填写进货单及销售单。
- ☒ 支持库存价格调整功能。
- ☒ 当外界环境（停电、网络病毒）干扰本系统时，系统可以自动保护原始数据的安全。

1.2.2 可行性分析

根据《GB8567—88 计算机软件产品开发文件编制指南》中可行性分析的要求，制定可行性研究报告如下。

1·引言

☒ 编写目的

以文件的形式给企业的决策层提供项目实施的参考依据，其中包括项目存在的风险、项目需要的投资和能够收获的最大效益。

☒ 背景

×××有限公司是一家以商业经营为主的私有企业。为了完善管理制度、增强企业的竞争力、实现信息化管理，公司决定开发进销存管理系统。



2·可行性研究的前提

☒ 要求

企业进销存管理系统必须提供商品信息、供应商信息和客户信息的基础设置；提供强大的多条件搜索功能和商品的进货、销售和库存管理功能；可以分不同权限、不同用户对该系统进行操作。另外，该系统还必须保证数据的安全性、完整性和准确性。

☒ 目标

企业进销存管理系统的目标是实现企业的信息化管理，减少盲目采购、降低采购成本、合理控制库存、减少资金占用并提升企业市场竞争力。

☒ 条件、假定和限制

为实现企业的信息化管理，必须对操作人员进行培训，而且将原有的库存、销售、入库等信息转换为信息化数据，需要操作员花费大量时间和精力来完成。为了不影响企业的正常运行，进销存管理系统必须在两个月的时间内交付用户使用。

系统分析人员需要2天内到位，用户需要5天时间确认需求分析文档。去除其中可能出现的问题，例如用户可能临时有事，占用6天时间确认需求分析。那么程序开发人员需要在1个月零15天的时间内进行系统设计、程序编码、系统测试、程序调试和网站部署工作。其间，还包括了员工每周的休息时间。

☒ 评价尺度

根据用户的要求，项目主要以企业进货、销售和查询统计功能为主，对于库存、销售和进货的记录信息应该及时、准确地保存，并提供相应的查询和统计。由于库存商品数量太多，不易盘点，传统的盘点方式容易出错，系统中的库存盘点功能要准确地计算出每种商品的损益数量，减少企业不必要的损失。

3·投资及效益分析

☒ 支出

根据系统的规模及项目的开发周期（两个月），公司决定投入7个人。为此，公司将直接支付9万元的工资及各种福利待遇。在项目安装及调试阶段，用户培训、员工出差等费用支出需要2万元。在项目维护阶段预计需要投入4万元的资金。累计项目投入需要15万元资金。

☒ 收益

用户提供项目资金32万元。对于项目运行后进行的改动，采取协商的原则根据改动规模额外提供资金。因此从投资与收益的效益比上，公司可以获得18万元的利润。

项目完成后，会给公司提供资源储备，包括技术、经验的积累，其后再开发类似的项目时，可以极大地缩短项目开发周期。

4·结论

根据上面的分析，在技术上不会存在问题，因此项目延期的可能性很小。在效益上公司投入7个人、2个月的时间获利18万元，效益比较可观。在公司今后发展上，可以储备网站开发的经验和资源。因此认为该项目可以开发。

1.2.3 编写项目计划书

根据《GB8567—88 计算机软件产品开发文件编制指南》中的项目开发计划要求，结合单位实际情况，设计项目计划书如下。

1·引言

☒ 编写目的

为了保证项目开发人员按时保质地完成预定目标，更好地了解项目实际情况，按照合理的顺序开展工作，现以书面的形式将项目开发生命周期中的项目任务范围、项目团队组织结构、团队成员的工作责任、团队内外沟通协作方式、开发进度、检查项目工作等内容描述出来，作为项目相关人员之间的共识和约定以及项目生命周期内的所有项目活动的行动基础。

☒ 背景

企业进销存管理系统是由×××有限公司委托我公司开发的大型管理系统，主要功能是实现企业进销存的信息化管理，包括统计查询、进货、销售、库存盘点及系统管理等功能。项目周期两个月。项目背景规划如表 1.1 所示。

表 1.1 项目背景规划

项 目 名 称	项目委托单位	任务提出者	项目承担部门
企业进销存管理系统	×××有限公司	陈经理	策划部门 研发部门 测试部门

2·概述

☒ 项目目标

项目目标应当符合 SMART 原则，把项目要完成的工作用清晰的语言描述出来。企业进销存管理系统的项目目标如下：

企业进销存管理系统的主要目的是实现企业进销存的信息化管理，主要的业务就是商品的采购、销售和入库，另外还需要提供统计查询功能，其中包括商品查询、供应商查询、客户查询、销售查询、入库查询和销售排行等。项目实施后，能够降低采购成本、合理控制库存、减少资金占用并提升企业市场竞争力，整个项目需要在两个月的时间内交付用户使用。

☒ 产品目标

时间就是金钱，效率就是生命。项目实施后，企业进销存管理系统能够为企业节省大量人力资源，减少管理费用，从而间接为企业节约成本，提高企业效益。

☒ 应交付成果

- 在项目开发完后，交付内容有企业进销存管理系统的源程序、系统的数据库文件、系统使用说明书。
- 将开发的进销存管理系统打包并安装到企业的网络计算机中。
- 企业进销存管理系统交付用户之后，进行系统无偿维护和服务 6 个月，超过 6 个月进行系统有偿维护与服务。

☑ 项目开发环境

操作系统为 Windows XP 或 Windows 2003 均可，使用集成开发工具 Eclipse，数据库采用 SQL Server 2000，项目运行环境为 JDK 6.0。

☑ 项目验收方式与依据

项目验收分为内部验收和外部验收两种方式。在项目开发完成后，首先进行内部验收，由测试人员根据用户需求和项目目标进行验收。项目在通过内部验收后，交给客户进行验收，验收的主要依据为需求规格说明书。

3·项目团队组织

☑ 组织结构

为了完成进销存管理系统的项目开发，公司组建了一个临时的项目团队，由公司副经理、项目经理、系统分析员、软件工程师、美工设计师和测试人员构成，如图 1.1 所示。

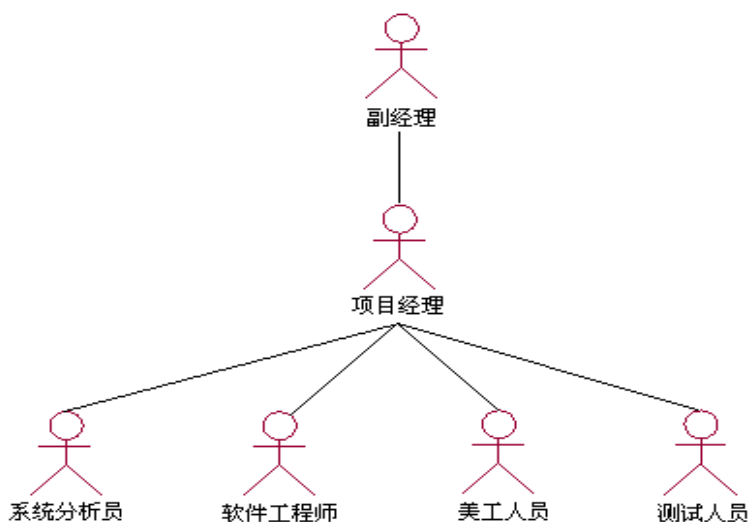


图 1.1 项目团队组织结构图

☑ 人员分工

为了明确项目团队中每个人的任务分工，现制定人员分工，如表 1.2 所示。

表 1.2 人员分工

姓 名	技 术 水 平	所 属 部 门	角 色	工 作 描 述
陈××	MBA	经理部	副经理	负责项目的审批、决策的实施
侯××	MBA	项目开发部	项目经理	负责项目的前期分析、策划、项目开发进度的跟踪、项目质量的检查
钟××	高级系统分析员	项目开发部	系统分析员	负责系统功能分析、系统框架设计
李××	高级美术工程师	美工设计部	美术工程师	负责软件美术设计
梁××	高级软件工程师	项目开发部	系统分析员	负责软件设计与编码
马××	高级软件工程师	项目开发部	软件工程师	负责软件设计与编码
王××	中级软件工程师	软件评测部	测试人员	负责软件测试与评定

1.3 系统设计

1.3.1 系统目标

根据需求分析的描述以及与用户的沟通，现制定系统实现目标如下。

- ☑ 界面设计简洁、友好、美观大方。
- ☑ 操作简单、快捷方便。
- ☑ 数据存储安全、可靠。
- ☑ 信息分类清晰、准确。
- ☑ 强大的查询功能，保证数据查询的灵活性。
- ☑ 提供销售排行榜，为管理员提供真实的数据信息。
- ☑ 提供灵活、方便的权限设置功能，使整个系统的管理分工明确。
- ☑ 对用户输入的数据，系统进行严格的数据检验，尽可能排除人为的错误。

1.3.2 系统功能结构

本系统包括基础信息、进货管理、销售管理、库存管理、查询统计、系统管理等6大部分。系统结构如图1.2所示。

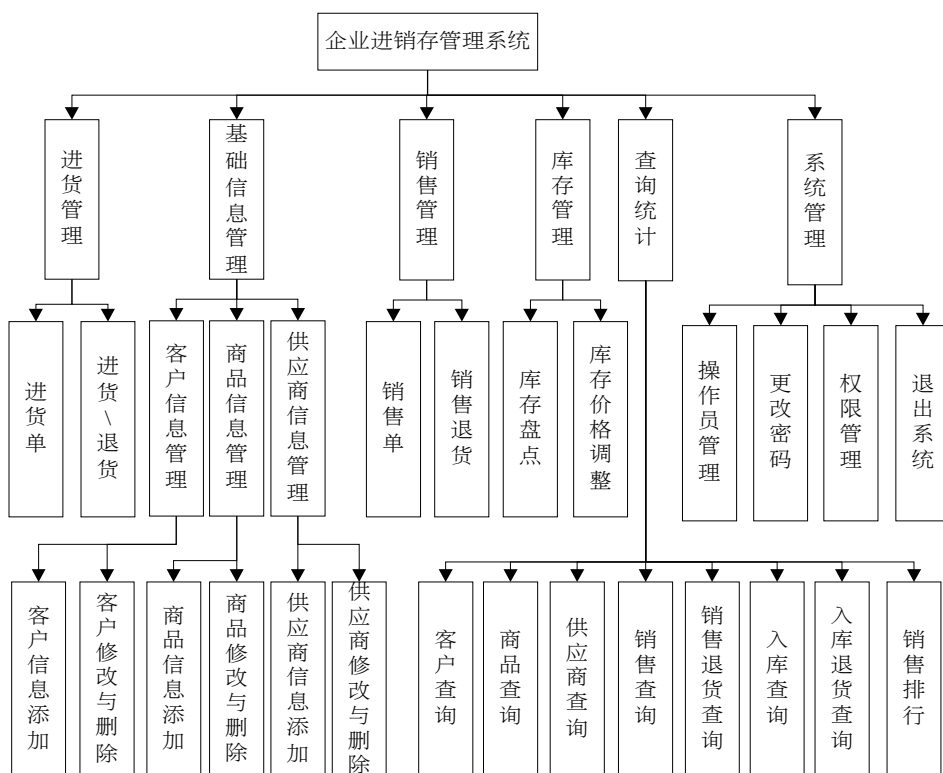


图 1.2 企业进销存管理系统功能结构

1.3.3 业务逻辑编码规则

遵守程序编码规则所开发的程序，代码清晰、整洁、方便阅读，并可以提高程序的可读性，要做到见其名知其意才能编写出优雅的程序代码。本节从数据库设计和程序编码两个方面介绍程序开发中的编码规则。

1·数据库对象命名规则

☒ 数据库命名规则

数据库命名以字母“db”开头（小写），后面加数据库相关英文单词或缩写。下面将举例说明，如表 1.3 所示。

表 1.3 数据库命名

数据库名称	描 述
db_JXC	企业进销存管理系统数据库
db_library	图书馆管理系统数据库

注意：在设计数据库时，为使数据库更容易理解，数据库命名时要注意大小写。

☒ 数据表命名规则

数据表以字母“tb”开头（小写），后面加数据库相关英文单词或缩写和数据表名，多个单词间用“_”分隔。下面将举例说明，如表 1.4 所示。

表 1.4 数据表命名

数据表名称	描 述
tb_sell_main	销售主表
tb_sell_detail	销售明细表

☒ 字段命名规则

字段一律采用英文单词或词组（可利用翻译软件）命名，如找不到专业的英文单词或词组可以用相同意义的英文单词或词组代替。下面将举例说明，如表 1.5 所示。

表 1.5 字段命名

字 段 名 称	描 述
ID	流水号
Name	名称
ProductInfo	商品信息

注意：在命名数据表的字段时，应注意字母的大小写。

2. 业务编码规则

☑ 供应商编号

供应商的 ID 编号是进销存管理系统中供应商的唯一标识，不同的供应商可以通过该编号来区分。该编号是供应商信息表的主键。在本系统中对该编号的编码规则：以字符串“gys”为编号前缀，加上 4 位数字作编号的后缀，这 4 位数字从 1000 开始。例如（gys1001）。

☑ 客户编号

和供应商编号类似，客户的 ID 编号也是客户的唯一标识，不同的客户将以该编号进行区分。该编号作为客户信息表的主键，有数据的唯一性的约束条件，所以，在客户信息表中不可能有两个相同的客户编号。企业进销存管理系统对客户编号的编码规则：以字符串“kh”为编号的前缀，加上 4 位数字作编号的后缀，这 4 位数字从 1000 开始。例如（kh1002）。

☑ 商品编号

商品编号是商品的唯一标识，它是商品信息表的主键，用于区分不同的商品。即使商品名称、单价、规格等信息相同，其 ID 编号也是不可能相同的，因为主键约束不可以存在相同的 ID 值。商品编号的编码规则和客户编号、供应商编号的编码规则相同，但是前缀使用了“sp”字符串。例如（sp2045）。

☑ 销售单编号

销售单编号用于区分不同的销售凭据。销售单编号的命名规则：以“XS”字符串为前缀，加上销售单的销售日期，再以 3 位数字作后缀。例如（XS20071205001）。

☑ 入库编号

入库编号用于区分不同的商品入库信息。入库编号的命名规则：以“RK”字符串为前缀，加上商品的入库日期，再以 3 位数字作后缀。例如（RK20071109003）。

☑ 入库退货编号

入库退货编号用于区分不同的入库退货信息。入库退货编号的命名规则：以“RT”字符串为前缀，加上商品入库的退货日期，再以 3 位数字作后缀。例如（RT20071109001）。

1.3.4 系统流程图

进销存管理系统的系统流程如图 1.3 所示。

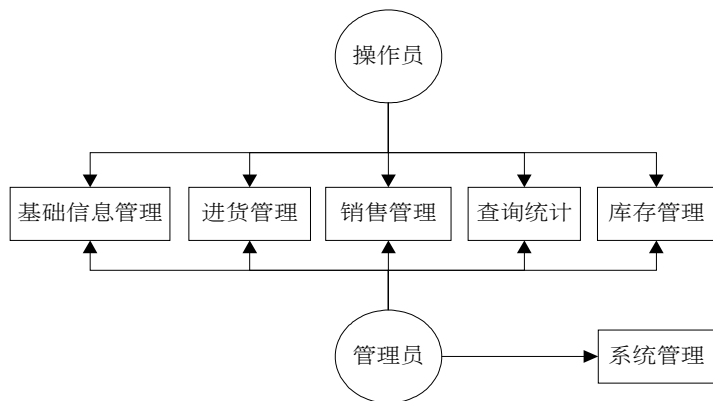


图 1.3 系统流程图

1.3.5 构建开发环境

在开发企业进销存管理系统时，使用了下面的软件环境。

- ☒ 操作系统：Windows 2003（SP1）。
- ☒ Java 开发包：JDK 1.6。
- ☒ 数据库：SQL Server 2000。
- ☒ 分辨率：最佳效果为 1024×768 像素。

注意：SP（Service Pack）为 Windows 操作系统补丁。

1.3.6 系统预览

企业进销存管理系统由多个程序界面组成，下面仅列出几个典型界面的预览，其他界面参见光盘中的源程序。

进销存管理系统的主界面如图 1.4 所示，该界面是所有功能模块的父窗体，其中包含调用所有功能模块的导航面板。商品进货单界面如图 1.5 所示，该界面将记录进货单据添加到数据库，其中进货单的编号由系统自动生成。



图 1.4 主窗体（光盘\...\com\lzw\JXCFrame.java）



图 1.5 进货单界面（光盘\...\

internalFrame\JinHuoDan.java）

操作员管理界面如图 1.6 所示，该界面由系统管理调用，主要用于操作员的添加、查看和删除。商品管理界面如图 1.7 所示，该界面包括商品的添加、修改和删除等功能。



图 1.6 操作员管理界面（光盘\...\internalFrame\CzyGL.java） 图 1.7 商品管理界面（光盘\...\internalFrame\ShangPinGuanLi.java）

说明：由于路径太长，因此省略了部分路径，省略的路径是“TM\01\JXCManager\src”。

1.3.7 文件夹组织结构

在进行系统开发之前，需要规划文件夹组织结构，也就是说，建立多个文件夹，对各个功能模块进行划分，实现统一管理。这样做的好处在于：易于开发、管理和维护。本系统的文件夹组织结构如图 1.8 所示。



图 1.8 文件夹组织结构图

1.4 数据库设计

1.4.1 数据库分析

本系统是一个桌面应用程序，它可以直接在本地计算机运行，而不需要像 Web 应用那样部署到指定的服务器中，所以这个进销存管理系统在本地计算机安装了 SQL Server 2000 数据服务器，将数据库和应用程序放在同一个计算机中，可以节省开销、提升系统安全性。另外，本系统也可以在网络内的其他计算机中运行，但是这需要将数据库对外开放，会降低数据安全性。其数据库运行环境如下：

☒ 硬件平台：

- CPU: P4 3.2GHz。
- 内存: 512MB 以上。
- 硬盘空间: 80GB。

☒ 软件平台：

- 操作系统: Windows 2003。
- 数据库: SQL Server 2000。

1.4.2 进销存管理系统的 E-R 图

企业进销存管理系统主要实现从进货、库存到销售的一体化信息管理，涉及商品信息、商品的供应商、购买商品的客户等多个实体。下面简单介绍几个关键的实体 E-R 图。

☒ 客户实体 E-R 图

企业进销存管理系统将记录所有的客户信息，在销售、退货等操作时，将直接引用该客户的实体属性。客户实体包括客户编号、客户名称、简称、地址、电话、邮政编码、联系人、联系人电话、传真、开户行和账号等属性，客户实体 E-R 图如图 1.9 所示。

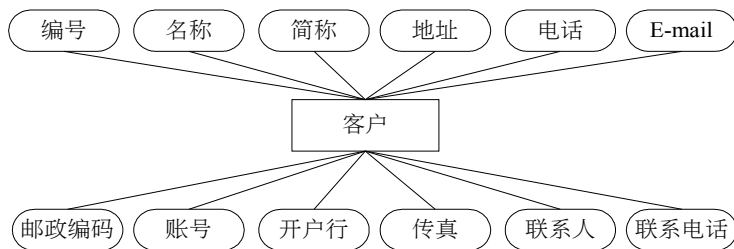


图 1.9 客户实体 E-R 图

☒ 供应商实体 E-R 图

不同的供应商可以为企业提供不同的商品，在商品信息中将引用商品供应商的实体属性。供应商实体包括编号、名称、简称、地址、电话、邮政编码、传真、联系人、联系电话、开户行和 E-mail 属性，供应商实体 E-R 图如图 1.10 所示。

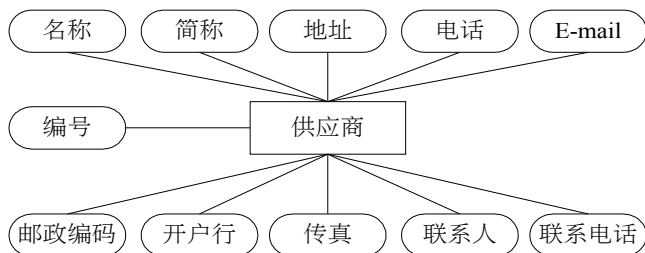


图 1.10 供应商实体 E-R 图

☑ 商品实体 E-R 图

商品信息是进销存管理系统中的基本信息，系统将维护商品的进货、退货、销售、入库等操作。商品实体包括编号、商品名称、商品简称、产地、单位、规格、包装、批号、批准文号、商品简介和供应商属性，商品实体 E-R 图如图 1.11 所示。

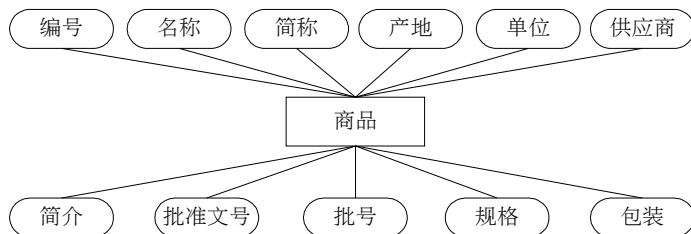


图 1.11 商品实体 E-R 图

1.4.3 使用 PowerDesigner 建模

在数据库概念设计中已经分析了本系统中主要的数据应实体对象，通过这些实体可以得出数据表结构的基本模型，最终实施到数据库中，形成完整的数据结构。本系统将使用 PowerDesigner 工具完成数据库建模，使用的版本为 12.5。使用该工具建模的步骤如下：

(1) 运行 PowerDesigner，并在 PowerDesigner 主窗口中选择主菜单中的 File/New 命令，在打开的 New 对话框左侧 Model type 列表框中选择 Physical Data Model（物理数据模型，简称 PDB）选项，在右侧的 Model name 文本框中输入模型名称 JXCManager，在 DBMS 下拉列表框中选择数据库管理系统。PowerDesigner 支持的数据库管理系统非常多，例如常用的 MySQL 5.0、Microsoft SQL Server 2005、Oracle Version 10gR2 等。企业进销存管理系统选择 Microsoft SQL Server 2000 作为数据库服务器，单击“确定”按钮，如图 1.12 所示。

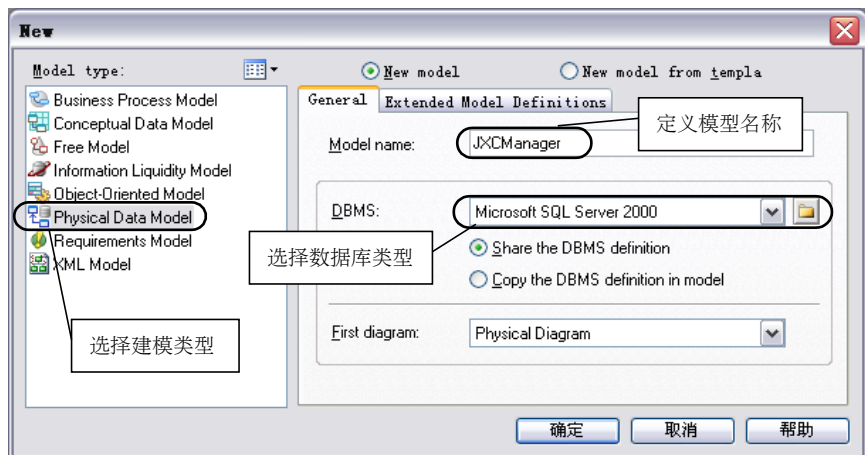




图 1.12 New 对话框

(2) 打开新建的 PDM 窗口。在该窗口的中心空白区域是模型编辑器，下方为输出窗口。另外还有一个浮动的工具面板，其中包括常用的建表工具、建视图工具和主外键引用工具，如图 1.13 所示。

(3) 在图 1.13 中单击“建表工具”按钮，这时鼠标指针将显示为，在模型编辑器的合适位置单击，此时在图形窗口中将显示如图 1.14 所示的数据表模型。

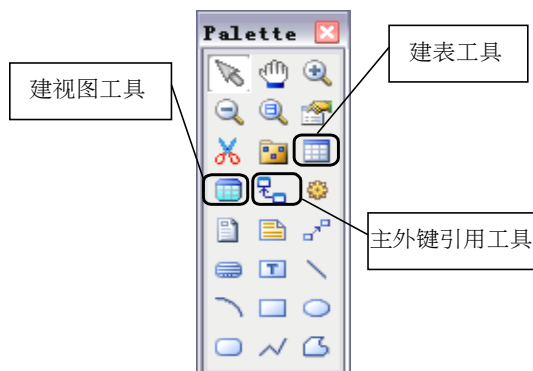


图 1.13 工具面板

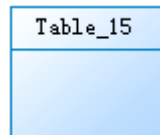




图 1.14 表符号

注意：细心的读者可以发现，此时的鼠标指针仍然是。如果再次单击还将出现类似图 1.14 所示的表符号。如果想取消该指针，可以单击工具面板中的按钮或单击鼠标右键。

(4) 在图 1.14 所示的表符号上双击鼠标左键，将打开 Table Properties（表属性）对话框。默认情况下选中的是 General 选项卡，在该选项卡的 Name 文本框中，输入表的名称 tb_manager，此时在 Code 文本框中也将自动显示 tb_manager，其他选择默认即可。

(5) 选择 Columns 选项卡，首先单击列输入列表的第一行，将自动转换第一行为编辑状态，然后在 Name 列输入字段名称为 ID，同时 Code 列也将自动显示为 ID，再在 Data Type 列中选择 int 选项，最后选中 P 列的复选框将该数据表字段设置为主键，此时 M 列的复选框也将自动被选中，它约束字段值不能为空。

(6) 按照步骤 (5) 的方法再添加两个列 name 和 PWD，但是不需要选中 P 列复选框设置主键，

如图 1.15 所示。

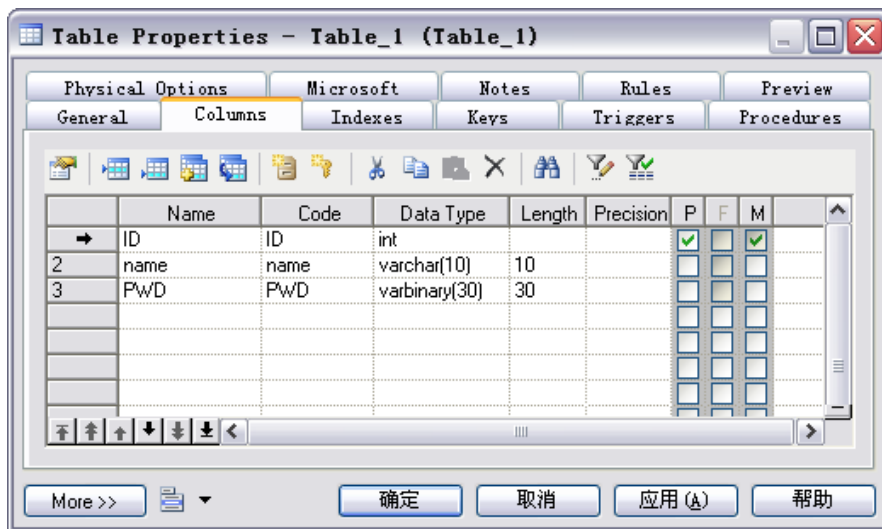



图 1.15 Columns（列）选项卡

(7) 在图 1.15 中单击“应用”按钮后，选择 ID 字段，单击左上角的  按钮，将打开 Column Properties（列属性）对话框，默认选中 General 选项卡，在其中选中 Identity 复选框，此项操作用于设置 ID 字段使用自动编号。

(8) 单击“应用”按钮后，再单击“确定”按钮，关闭 Column Properties 对话框。

(9) 单击“确定”按钮，关闭 Table Properties 对话框，完成 tb_manager 表的创建。

(10) 按照步骤 (3)~步骤 (9) 的方法创建本系统中的其他数据表，并通过主外键引用工具建立各表间的依赖关系。创建完成的模型如图 1.16 所示。

- 技巧：在默认情况下，创建后的表模型中的全部文字均为常规样式的宋体 8 号字，如果想修改文字的格式，可以选中全部表符号，按 Ctrl+T 键，在打开的 Symbol Format 对话框中选择 Font 选项卡，从中设置相关内容的字体及样式和字号等。

(11) 选择 PowerDesigner 主菜单中的 Database/Generate Database 命令，将打开 Database Generation 对话框。在该对话框中设置导出的脚本文件的名称（如 jxc.sql）及保存路径（如 D:\JXC），选中 Script generation 单选按钮，如图 1.17 所示。单击“确定”按钮，会在指定的路径中生成数据库脚本文件。

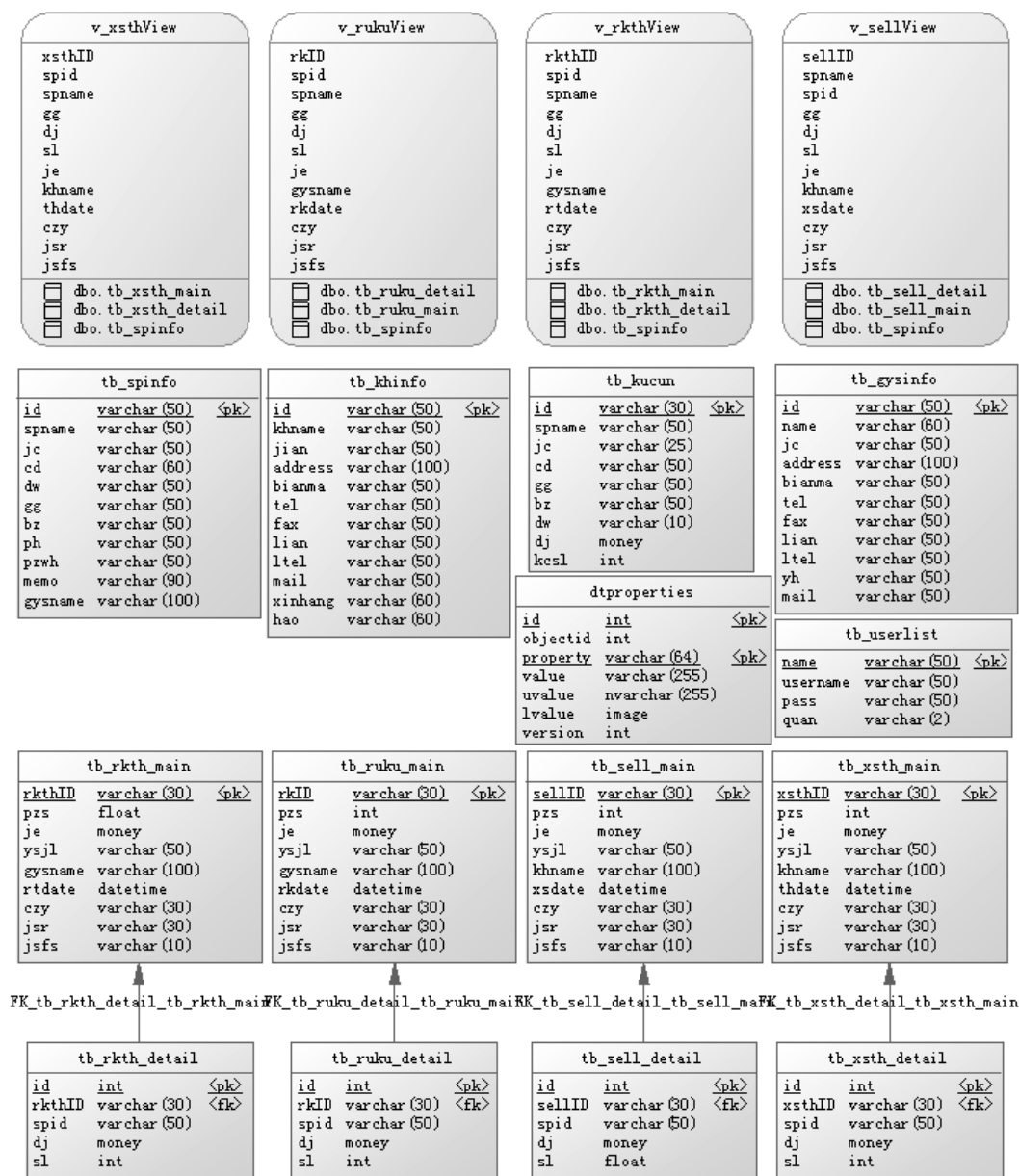


图 1.16 企业进销存管理系统的模型

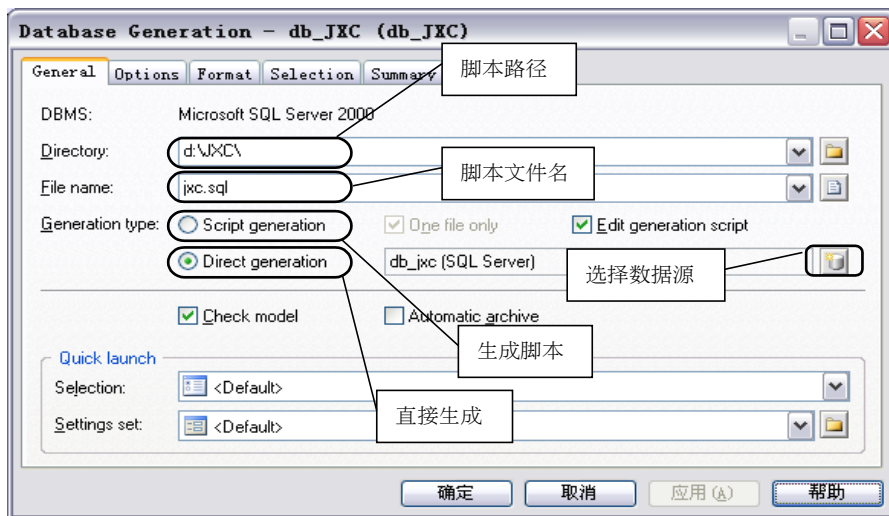



图 1.17 Database Generation 对话框

(12) 在图 1.17 所示的对话框中选择 **Direct generation** 单选按钮，可以使用 ODBC 数据源直接在数据库管理系统中生成数据表和视图。但是，必须先创建数据库的数据源，然后单击  按钮选择指定的数据源，并单击“确定”按钮。


1.4.4 创建数据库

在导出数据库脚本文件后，就可以在查询分析器中执行该脚本来创建数据库及数据表了。具体步骤如下：

(1) 选择“开始”/“所有程序”/Microsoft SQL Server/“查询分析器”命令，在弹出的“连接到 SQL Server”对话框中输入访问数据库的用户名和密码，如图 1.18 所示。单击“确定”按钮。



图 1.18 “连接到 SQL Server”对话框

(2) 在打开的 SQL 查询分析器中选择“文件”/“打开”命令，在弹出的对话框中选择数据库脚本文件，然后单击“打开”按钮，返回查询分析器，选择执行脚本的数据库，然后单击  按钮执行脚本。

本中的命令创建数据库的表结构，如图 1.19 所示。

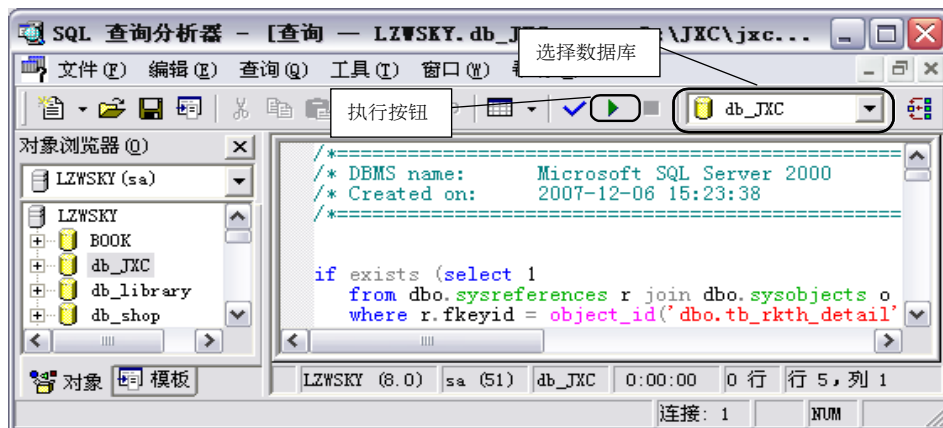


图 1.19 查询分析器运行效果

1.5 主窗体设计

主窗体界面也是该系统的欢迎界面。应用程序的主窗体必须设计层次清晰的系统菜单和工具栏，其中系统菜单包含系统中所有功能的菜单项，而工具栏主要提供常用功能的快捷访问按钮。企业进销存管理系统采用导航面板综合了系统菜单和工具栏的优点，而且导航面板的界面更加美观，操作更快捷。主窗体的运行结果如图 1.20 所示。



图 1.20 程序主窗体界面效果

1.5.1 创建主窗体

创建主窗体的步骤如下：

(1) 创建 JXCFrame 类，在类中创建并初始化窗体对象，为窗体添加桌面面板，并设置背景图片。关键代码如下：

例程01 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JDesktopPane desktopPane;
private JFrame frame;
private JLabel backLabel;
private Preferences preferences;
//创建窗体的Map类型集合对象
private Map<String, JInternalFrame> ifs = new HashMap<String, JInternalFrame>();
public JXCFrame() {
    frame = new JFrame("企业进销存管理系统");           //创建窗体对象
    frame.addComponentListener(new FrameListener());      //添加窗体事件监听器
    frame.getContentPane().setLayout(new BorderLayout());  //设置布局管理器
    frame.setBounds(100, 100, 800, 600);                 //设置窗体位置和大小
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //设置窗体默认的关闭方式
    backLabel = new JLabel();                             //背景标签
    backLabel.setVerticalAlignment(SwingConstants.TOP);    //设置背景标签垂直对齐方式
    backLabel.setHorizontalAlignment(SwingConstants.CENTER); //设置背景标签水平对齐方式
    updateBackImage();                                   //调用初始化背景标签的方法
    desktopPane = new JDesktopPane();                    //创建桌面面板
    desktopPane.add(backLabel, new Integer(Integer.MIN_VALUE)); //将背景标签添加到桌面面板中
    frame.getContentPane().add(desktopPane);              //添加桌面面板到窗体中
    JTabbedPane navigationPanel = createNavigationPanel(); //创建导航面板
    frame.getContentPane().add(navigationPanel, BorderLayout.NORTH); //添加导航面板到窗体中
    frame.setVisible(true);                               //显示窗体
}
```

(2) 编写 updateBackImage()方法，在该方法中初始化背景标签，背景标签使用 HTML 超文本语言设置了主窗体的背景图片，该图片将随主窗体的大小自动缩放。关键代码如下：

例程02 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private void updateBackImage() {
    if (backLabel != null) {
        int backw = JXCFrame.this.frame.getWidth();
        int backh = frame.getHeight();
        backLabel.setSize(backw, backh);                //初始化背景标签的大小
        backLabel.setText("<html><body><image width=\"" + backw
            + "\" height=\"" + (backh - 110) + "\" src=\""
            + JXCFrame.this.getClass().getResource("welcome.jpg")
            + "\"></img></body></html>");                //设置背景标签的图像
    }
}
```

(3) 在类的静态代码段中设置进销存管理系统的外观样式。Swing 支持跨平台特性，它可以在不同的操作系统中保持一致的外观风格，但是本系统使用 UIManager 类的 setLookAndFeel() 方法设置程序界面使用本地外观，这样可以使程序更像本地应用程序。关键代码如下：

例程03 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
static {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

(4) 编写主窗体的 main() 入口方法，在该方法中创建登录窗体对象，登录窗体会验证登录信息，并显示主窗体界面。关键代码如下：

例程04 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new Login();
        }
    });
}
```

1.5.2 创建导航面板

创建导航面板的步骤如下：

(1) 在 JXCFrame 类中编写 createNavigationPanel() 方法，在该方法中创建 JTabbedPane 选项卡面板对象。为突出选项卡的立体效果，设置该选项卡使用 BevelBorder 边框效果，然后依次创建基础信息管理、库存管理、销售管理、查询统计、进货管理和系统管理的选项卡。关键代码如下：

例程05 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JTabbedPane createNavigationPanel() { //创建导航面板的方法
    JTabbedPane tabbedPane = new JTabbedPane();
    tabbedPane.setFocusable(false);
    tabbedPane.setBackground(new Color(211, 230, 192));
    tabbedPane.setBorder(new BevelBorder(BevelBorder.RAISED));
    JPanel baseManagePanel = new JPanel(); //基础信息管理面板
    baseManagePanel.setBackground(new Color(215, 223, 194));
    baseManagePanel.setLayout(new BorderLayout(baseManagePanel, BorderLayout.X_AXIS));
    baseManagePanel.add(createFrameButton("客户信息管理", "KeHuGuanLi"));
    baseManagePanel.add(createFrameButton("商品信息管理", "ShangPinGuanLi"));
    baseManagePanel.add(createFrameButton("供应商信息管理", "GysGuanLi"));
    JPanel depotManagePanel = new JPanel(); //库存管理面板
    depotManagePanel.setBackground(new Color(215, 223, 194));
```

```

depotManagePanel.setLayout(new BorderLayout(depotManagePanel, BorderLayout.X_AXIS));
depotManagePanel.add(createFrameButton("库存盘点", "KuCunPanDian"));
depotManagePanel.add(createFrameButton("价格调整", "JiaGeTiaoZheng"));
JPanel sellManagePanel = new JPanel(); //销售管理面板
sellManagePanel.setBackground(new Color(215, 223, 194));
sellManagePanel.setLayout(new BorderLayout(sellManagePanel, BorderLayout.X_AXIS));
sellManagePanel.add(createFrameButton("销售单", "XiaoShouDan"));
sellManagePanel.add(createFrameButton("销售退货", "XiaoShouTuiHuo"));
JPanel searchStatisticPanel = new JPanel(); //查询统计面板
searchStatisticPanel.setBounds(0, 0, 600, 41);
searchStatisticPanel.setName("searchStatisticPanel");
searchStatisticPanel.setBackground(new Color(215, 223, 194));
searchStatisticPanel.setLayout(new BorderLayout(searchStatisticPanel, BorderLayout.X_AXIS));
searchStatisticPanel.add(createFrameButton("客户信息查询", "KeHuChaXun"));
searchStatisticPanel.add(createFrameButton("商品信息查询", "ShangPinChaXun"));
searchStatisticPanel.add(createFrameButton("供应商信息查询", "GongYingShangChaXun"));
searchStatisticPanel.add(createFrameButton("销售信息查询", "XiaoShouChaXun"));
searchStatisticPanel.add(createFrameButton("销售退货查询", "XiaoShouTuiHuoChaXun"));
searchStatisticPanel.add(createFrameButton("入库查询", "RuKuChaXun"));
searchStatisticPanel.add(createFrameButton("入库退货查询", "RuKuTuiHuoChaXun"));
searchStatisticPanel.add(createFrameButton("销售排行", "XiaoShouPaiHang"));
JPanel stockManagePanel = new JPanel(); //进货管理面板
stockManagePanel.setBackground(new Color(215, 223, 194));
stockManagePanel.setLayout(new BorderLayout(stockManagePanel, BorderLayout.X_AXIS));
stockManagePanel.add(createFrameButton("进货单", "JinHuoDan"));
stockManagePanel.add(createFrameButton("进货退货", "JinHuoTuiHuo"));
JPanel sysManagePanel = new JPanel(); //系统管理面板
sysManagePanel.setBackground(new Color(215, 223, 194));
sysManagePanel.setLayout(new BorderLayout(sysManagePanel, BorderLayout.X_AXIS));
sysManagePanel.add(createFrameButton("操作员管理", "CzyGL"));
sysManagePanel.add(createFrameButton("更改密码", "GengGaiMiMa"));
sysManagePanel.add(createFrameButton("权限管理", "QuanManager"));
//将所有面板添加到导航面板中
tabbedPane.addTab(" 基础信息管理 ", null, baseManagePanel, "基础信息管理");
tabbedPane.addTab(" 进货管理 ", null, stockManagePanel, "进货管理");
tabbedPane.addTab(" 销售管理 ", null, sellManagePanel, "销售管理");
tabbedPane.addTab(" 查询统计 ", null, searchStatisticPanel, "查询统计");
tabbedPane.addTab(" 库存管理 ", null, depotManagePanel, "库存管理");
tabbedPane.addTab(" 系统管理 ", null, sysManagePanel, "系统管理");
return tabbedPane;
}

```

(2) 编写 `createFrameButton()` 方法，该方法负责创建 `Action` 对象，该对象用于创建并显示窗体对象，另外，它还包含图标、文本等属性，如果将 `Action` 对象添加到系统菜单栏或者工具栏中，会直接创建相应的菜单项和工具按钮，而且这些菜单项和工具按钮将显示 `Action` 对象中的文本和图标属性。本系统没有使用系统菜单，所以该方法直接创建按钮对象。关键代码如下：

例程06 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
private JButton createFrameButton(String fName, String cname) {           //为内部窗体添加 Action 的方法
    String imgUrl = "res/ActionIcon/" + fName + ".png";
    String imgUrl_roll = "res/ActionIcon/" + fName + "_roll.png";
    String imgUrl_down = "res/ActionIcon/" + fName + "_down.png";
    Icon icon = new ImageIcon(imgUrl);                                   //创建按钮图标
    Icon icon_roll = null;
    if (imgUrl_roll != null)
        icon_roll = new ImageIcon(imgUrl_roll);                       //创建鼠标经过按钮时的图标
    Icon icon_down = null;
    if (imgUrl_down != null)
        icon_down = new ImageIcon(imgUrl_down);                       //创建按钮按下的图标
    Action action = new openFrameAction(fName, cname, icon);           //用 openFrameAction 类创建 Action 对象
    JButton button = new JButton(action);
    ■ button.setMargin(new Insets(0, 0, 0, 0));
    ■ button.setHideActionText(true);
    ■ button.setFocusPainted(false);
    ■ button.setBorderPainted(false);
    ■ button.setContentAreaFilled(false);
    if (icon_roll != null)
    ■     button.setRolloverIcon(icon_roll);
    if (icon_down != null)
    ■     button.setPressedIcon(icon_down);
    return button;
}
```

■ 代码贴士

- setMargin()：该方法用于设置按钮的四周边界大小。
- setHideActionText()：该方法用于设置按钮隐藏 Action 对象中的文本信息，例如一个只显示图标的按钮可以取消文本使按钮更加美观。
- setFocusPainted()：该方法用于设置按钮获取焦点时，是否绘制焦点样式。导航面板取消了这个焦点样式，因为它破坏了按钮图标美观性。
- setBorderPainted()：该方法设置是否绘制按钮的边框样式，导航面板取消了边框样式，因为按钮的图标需要覆盖整个按钮。
- setContentAreaFilled()：该方法设置是否绘制按钮图形，在不同的操作系统，甚至系统不同的皮肤样式中都有不同的图形。导航面板取消了按钮的图形效果，因为导航面板要使用图标绘制整个按钮。
- setRolloverIcon()：该方法用于设置鼠标经过按钮时，按钮所使用的图标。
- setPressedIcon()：该方法用于设置鼠标按下按钮时，按钮所使用的图标。

(3) 编写内部类 openFrameAction，它必须继承 AbstractAction 类实现 Action 接口。该类用于创建导航按钮的 Action 对象，并为每个导航按钮定义创建并显示不同窗体对象的动作监听器，这个监听器在按钮被按下时，调用 getIFrame()方法获取相应的窗体对象，并显示在主窗体中。关键代码如下：

例程07 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
protected final class openFrameAction extends AbstractAction {           //主窗体菜单项的单击事件监听器
    private String frameName = null;
```

```

private openFrameAction() {
}
public openFrameAction(String cname, String frameName, Icon icon) {
    this.frameName = frameName;
    putValue(Action.NAME, cname); //设置 Action 的名称
    putValue(Action.SHORT_DESCRIPTION, cname); //设置 Action 的提示文本框
    putValue(Action.SMALL_ICON, icon); //设置 Action 的图标
}
public void actionPerformed(final ActionEvent e) {
    JInternalFrame jf = getIFrame(frameName); //调用 getIFrame() 方法
    //在内部窗体关闭时，从内部窗体容器 ifs 对象中清除该窗体
    jf.addInternalFrameListener(new InternalFrameAdapter() {
        public void internalFrameClosed(InternalFrameEvent e) {
            ifs.remove(frameName);
        }
    });
    if (jf.getDesktopPane() == null) {
        desktopPane.add(jf); //将窗体添加到主窗体中
        jf.setVisible(true); //显示窗体
    }
    try {
        jf.setSelected(true); //使窗体处于被选择状态
    } catch (PropertyVetoException e1) {
        e1.printStackTrace();
    }
}
}

```

(4) 编写 `getIFrame()` 方法，该方法负责创建指定名称的窗体对象，在方法中使用了 Java 的反射技术，调用不同窗体类的默认构造方法创建窗体对象。关键代码如下：

例程08 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```

private JInternalFrame getIFrame(String frameName) { //获取内部窗体的唯一实例对象
    JInternalFrame jf = null;
    if (!ifs.containsKey(frameName)) {
        try {
            Class fClass = Class.forName("internalFrame." + frameName);
            Constructor constructor = fClass.getConstructor(null);
            jf = (JInternalFrame) constructor.newInstance(null);
            ifs.put(frameName, jf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    } else {
        jf = ifs.get(frameName);
    }
    return jf;
}

```

1.6 公共模块设计

在本系统的项目空间中，有部分模块是公用的，或者是多个模块甚至整个系统的配置信息，它们被多个模块重复调用完成指定的业务逻辑，本节将这些公共模块提出来作单独介绍。

1.6.1 编写 Dao 公共类

Dao 类主要负责有关数据库的操作，该类在静态代码段中驱动并连接数据库，然后将所有的数据库访问方法定义为静态的。本节将介绍 Dao 类中有关数据库操作的关键方法。Dao 类的定义代码如下：

例程09 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
public class Dao {
    protected static String dbClassName = "com.microsoft.jdbc.sqlserver.SQLServerDriver";
    protected static String dbUrl = "jdbc:microsoft:sqlserver://localhost:1433;"
        + "DatabaseName=db_JXC;SelectMethod=Cursor";
    protected static String dbUser = "sa";
    protected static String dbPwd = "";
    protected static String second = null;
    public static Connection conn = null;
    static {
        try {
            if (conn == null) {
                Class.forName(dbClassName).newInstance();           //加载数据库驱动类
                conn = DriverManager.getConnection(dbUrl, dbUser, dbPwd); //获取数据库连接
            }
        } catch (Exception ee) {
            ee.printStackTrace();
        }
    }
}
```

代码贴士

dbClassName：该成员变量用于定义数据库驱动类的名称。

dbUrl：该成员变量用于定义访问数据库的 URL 路径。

dbUser：该成员变量用于定义访问数据库的用户名称。

dbPwd：该成员变量用于定义访问数据库的用户密码。

conn：该成员变量用于定义连接数据库的对象。

1 · addGys()方法

该方法用于添加供应商的基础信息，它接收供应商的实体类 TbGysinfo 作方法的参数，然后把实体对象中的所有属性存入供应商数据表中。关键代码如下：

例程10 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```

//添加供应商信息的方法
public static boolean addGys(TbGysinfo gysInfo) {
    if (gysInfo == null) //如果供应商实体对象为空
        return false; //则返回false
    return insert("insert tb_gysinfo values('" + gysInfo.getId() + "','" //执行供应商添加
        + gysInfo.getName() + "','" + gysInfo.getJc() + "','"
        + gysInfo.getAddress() + "','" + gysInfo.getBianma() + "','"
        + gysInfo.getTel() + "','" + gysInfo.getFax() + "','"
        + gysInfo.getLian() + "','" + gysInfo.getLtel() + "','"
        + gysInfo.getMail() + "','" + gysInfo.getYh() + "')");
}

```

2 · getGysInfo()方法

该方法将根据 Item 对象中封装的供应商 ID 编号和供应商名称获取指定供应商的数据，并将该供应商的数据封装到实体对象中，然后返回该实体对象。关键代码如下：

例程11 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```

//读取指定供应商信息
public static TbGysinfo getGysInfo(Item item) {
    String where = "name='" + item.getName() + "'"; //默认的查询条件以供应商名称为主
    if (item.getId() != null) //如果Item对象中存有ID编号
        where = "id='" + item.getId() + "'"; //则以ID编号为查询条件
    TbGysinfo info = new TbGysinfo();
    ResultSet set = findForResultSet("select * from tb_gysinfo where " + where);
    try {
        if (set.next()) {
            info.setId(set.getString("id").trim()); //封装供应商数据到实体对象中
            info.setAddress(set.getString("address").trim());
            info.setBianma(set.getString("bianma").trim());
            info.setFax(set.getString("fax").trim());
            info.setJc(set.getString("jc").trim());
            info.setLian(set.getString("lian").trim());
            info.setLtel(set.getString("ltel").trim());
            info.setMail(set.getString("mail").trim());
            info.setName(set.getString("name").trim());
            info.setTel(set.getString("tel").trim());
            info.setYh(set.getString("yh").trim());
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return info; //返回供应商实体对象
}

```

3 · updateGys()方法

该方法用于更新供应商的基础信息，它接收供应商的实体类 TbGysinfo 作方法参数，在方法中直接解析供应商实体对象中的属性，并将这些属性更新到数据表中。关键代码如下：

例程12 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

//修改供应商信息的方法

```
public static int updateGys(TbGysinfo gysInfo) {
    return update("update tb_gysinfo set jc=" + gysInfo.getJc()
        + ",address=" + gysInfo.getAddress() + ",bianma="
        + gysInfo.getBianma() + ",tel=" + gysInfo.getTel()
        + ",fax=" + gysInfo.getFax() + ",lian=" + gysInfo.getLian()
        + ",ltel=" + gysInfo.getLtel() + ",mail="
        + gysInfo.getMail() + ",yh=" + gysInfo.getYh()
        + " where id=" + gysInfo.getId() + "");
}
```

4 · insertRukuInfo()方法

该方法负责完成入库单信息的添加，它涉及到库存表、入库主表和入库详细表等多个数据表的操作。为保证数据的完整性，该方法将入库信息的添加操作放在事务中完成，方法将接收入库主表的实体类 TbRukuMain 作参数，该实体类中包含了入库详细表的引用。关键代码如下：

例程13 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```
public static boolean insertRukuInfo(TbRukuMain ruMain) { //在事务中添加入库信息
    try {
        boolean autoCommit = conn.setAutoCommit();
        conn.setAutoCommit(false); //取消自动提交模式
        insert("insert into tb_uku_main values(" + ruMain.getRkId() //添加入库主表记录
            + "," + ruMain.getPzs() + "," + ruMain.getJe() + ","
            + ruMain.getYsjl() + "," + ruMain.getGysname() + ","
            + ruMain.getRkdate() + "," + ruMain.getCzy() + ","
            + ruMain.getJsrr() + "," + ruMain.getJsfs() + ")");
        Set<TbRukuDetail> rkDetails = ruMain.getTabRukuDetails();
        for (Iterator<TbRukuDetail> iter = rkDetails.iterator(); iter.hasNext();) {
            TbRukuDetail details = iter.next();
            insert("insert into tb_uku_detail values(" + ruMain.getRkId() //添加入库详细表记录
                + "," + details.getTabSpinfo() + "," + details.getDj() + "," + details.getSl() + ")");
            Item item = new Item();
            item.setId(details.getTabSpinfo());
            TbSpinfo spInfo = getSpInfo(item);
            if (spInfo.getId() != null && !spInfo.getId().isEmpty()) {
                TbKucun kucun = getKucun(item);
                if (kucun.getId() == null || kucun.getId().isEmpty()) { //添加或修改库存表记录
                    insert("insert into tb_kucun values(" + spInfo.getId()
                        + "," + spInfo.getSpname() + "," + spInfo.getJc() + "," + spInfo.getCd()
                        + "," + spInfo.getGg() + "," + spInfo.getBz() + "," + spInfo.getDw()
                        + "," + details.getDj() + "," + details.getSl() + ")");
                } else {
                    int sl = kucun.getKcsl() + details.getSl();
                    update("update tb_kucun set kcsl=" + sl + ",dj=" + details.getDj() + " where id=" + kucun.getId() + "");
                }
            }
        }
        conn.commit(); //提交事务
    } catch (Exception e) {
        //异常处理
    }
}
```

```

        conn.setAutoCommit(autoCommit); //恢复自动提交模式
    } catch (SQLException e) {
        try {
            conn.rollback(); //如果出错，回退事务
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
        e.printStackTrace();
    }
    return true;
}

```

■ 代码贴士

- getAutoCommit()：该方法用于获取事务自动提交模式。
- setAutoCommit()：该方法用于设置事务的自动提交模式。
- commit()：该方法用于执行事务提交。
- rollback()：该方法在事务执行失败时，执行回退操作。

5 · getKucun()方法

该方法用于获取指定商品 ID 编号或名称的库存信息。方法接收一个 Item 对象作参数，该对象中封装了商品的 ID 编号和商品名称信息，如果库存表中存在该商品的库存记录，就获取该记录并将记录中的数据封装到库存表的实体对象中，然后将该实体对象作为方法的返回值。关键代码如下：

例程14 代码位置：光盘\TM\01\JXCManager\src\com\lzw\dao\Dao.java

```

//获取库存商品信息
public static TbKucun getKucun(Item item) {
    String where = "spname=" + item.getName() + "";
    if (item.getId() != null)
        where = "id=" + item.getId() + "";
    ResultSet rs = findForResultSet("select * from tb_kucun where " + where);
    TbKucun kucun = new TbKucun();
    try {
        if (rs.next()) {
            kucun.setId(rs.getString("id"));
            kucun.setSpname(rs.getString("spname"));
            kucun.setJc(rs.getString("jc"));
            kucun.setBz(rs.getString("bz"));
            kucun.setCd(rs.getString("cd"));
            kucun.setDj(rs.getDouble("dj"));
            kucun.setDw(rs.getString("dw"));
            kucun.setGg(rs.getString("gg"));
            kucun.setKcsl(rs.getInt("kcsl"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```
}  
    return kucun;  
}
```

1.6.2 编写 Item 类

Item 类是系统的公共类之一，主要用于封装和传递参数信息，这是典型命令模式的实现。在 Dao 类中经常使用该类作为方法参数；另外，在各个窗体界面中也经常使用该类作组件数据，其 toString() 方法将返回 name 属性值，所以显示到各个组件上的内容就是 Item 类的对象所代表的商品、供应商或者客户等信息中的名称。定义该类的关键代码如下：

例程15 代码位置：光盘\TM\01\JXCManager\src\internalFrame\guanli\Item.java

```
public class Item {  
    public String id;           //定义id属性  
    public String name;        //定义名称属性  
    public String getId() {     //定义暴露ID属性的方法  
        return id;  
    }  
    public void setId(String id) {  
        this.id = id;  
    }  
    public String getName() {   //定义暴露名称属性的方法  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String toString() {  //定义该类的字符串表现形式  
        return getName();  
    }  
}
```

1.7 基础信息模块设计

基础信息模块用于管理企业进销存管理系统中的客户、商品和供应商信息，其功能主要是对这些基础信息进行添加、修改和删除。

1.7.1 基础信息模块概述

企业进销存管理系统中的基础信息模块主要包括客户管理、商品管理和供应商管理 3 部分，由于它们的实现方法基本相似，本节将以供应商管理部分为主，介绍基础信息模块对本系统的意义和实现的业务逻辑。

1 · 供应商添加



供应商添加功能主要负责为系统添加新的供应商记录。在企业进销存管理系统中，商品是主要的管理对象，而系统中所有的商品都由不同的供应商提供，这就需把不同的供应商信息添加到系统中，在商品信息中会关联系统中对应的供应商信息。供应商添加功能的程序界面如图 1.21 所示。

2· 供应商修改与删除

供应商的修改与删除功能主要用于维护系统中的供应商信息。在供应商的联系方式发生改变时，必须更新系统中的记录，以提供供应商的最新信息。另外，当不再与某家供应商合作时，需要从系统中删除供应商的记录信息。程序运行界面如图 1.22 所示。



图 1.21 供应商添加界面



图 1.22 供应商修改与删除功能界面

1.7.2 基础信息模块技术分析

基础信息模块中使用了 Java Swing 的 JTabbedPane 选项卡面板组件分别为客户信息管理、商品信息管理、和供应商信息管理提供了多个操作界面，例如供应商信息管理中分别存在供应商添加和供应商修改与删除界面，而这两个界面都存在于一个窗体中，可以通过选择顶部的两个选项卡，在不同的界面中来回切换。

1.7.3 供应商添加实现过程

■ 供应商添加使用的数据表：tb_gysinfo。

(1) 创建 GysTianJiaPanel 类，用于实现本系统的供应商添加功能。该类将在界面中显示多个用于输入供应商信息的文本框。界面中定义的主要控件如表 1.6 所示。

表 1.6 供应商添加界面中的主要控件

控件类型	控件名称	主要属性设置	用途
JtextField	quanChengF	无	供应商全称
	JianChengF	无	简称
	BianMaF	无	邮政编码

	DiZhiF	无	地址
	DianHuaF	无	电话
JtextField	ChuanZhenF	无	传真
	LianXiRenF	无	联系人
	lianXiRenDianHuaF	无	联系人电话
	YinHangF	无	开户银行
	EmailF	无	电子信箱

续表

控件类型	控件名称	主要属性设置	用途
Jbutton	TjButton	设置按钮文本为“添加” 设置动作监听器为TjActionListener类的实例对象	添加
	ResetButton	设置按钮文本为“重填” 设置动作监听器为ResetActionListener类的实例对象	重填

（2）创建 ResetActionListener 类，该类是“重填”按钮的事件监听器，它必须实现 ActionListener 接口，并在 actionPerformed()方法中清除界面中的所有文本框内容。关键代码如下：

例程16 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysTianJiaPanel.java

```

class ResetActionListener implements ActionListener {           //重填按钮的事件监听类
    public void actionPerformed(
        final ActionEvent e) {
        diZhiF.setText("");
        bianMaF.setText("");
        chuanZhenF.setText("");
        jianChengF.setText("");
        lianXiRenF.setText("");
        lianXiRenDianHuaF.setText("");
        EMailF.setText("");
        quanChengF.setText("");
        dianHuaF.setText("");
        yinHangF.setText("");
    }
}

```

代码贴士

ActionListener 接口：该接口是控件的动作监听器接口，实现该接口的类可以成为按钮和菜单项等控件的监听器。

actionPerformed()：该方法是监听器 ActionListener 接口定义的方法，当事件产生时，将调用监听器实现类的 actionPerformed()方法处理相应的业务逻辑。

ActionEvent：该类是动作事件类，当用户单击按钮时，将产生该事件，这个事件会被监听器捕获并执行相应的业务逻辑。

(3) 创建 `TjActionListener` 类，该类是“添加”按钮的事件监听器，它必须实现 `ActionListener` 接口，并在 `actionPerformed()` 方法中实现用户输入的验证和供应商信息的保存。关键代码如下：

例程17 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysTianJiaPanel.java


```
class TjActionListener implements ActionListener { //添加按钮的事件监听类
    public void actionPerformed(final ActionEvent e) {
        if (diZhiF.getText().equals("") || quanChengF.getText().equals("") //验证用户输入
            || chuanZhenF.getText().equals("") || jianChengF.getText().equals("")
            || yinHangF.getText().equals("") || bianMaF.getText().equals("")
            || diZhiF.getText().equals("") || lianXiRenF.getText().equals("")
            || lianXiRenDianHuaF.getText().equals("")
            || EMailF.getText().equals("") || dianHuaF.getText().equals("")) {
            JOptionPane.showMessageDialog(GysTianJiaPanel.this, "请填写全部信息");
            return;
        }
        try { //验证是否存在同名供应商
            ResultSet haveUser = Dao.query("select * from tb_gysinfo where name="
                + quanChengF.getText().trim() + "");
            if (haveUser.next()) {
                JOptionPane.showMessageDialog(GysTianJiaPanel.this,
                    "供应商信息添加失败，存在同名供应商", "供应商添加信息",
                    JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            ResultSet set = Dao.query("select max(id) from tb_gysinfo"); //获取供应商的最大 ID 编号
            String id = null;
            if (set != null && set.next()) { //创建新的供应商编号
                String sid = set.getString(1).trim();
                if (sid == null)
                    id = "gys1001";
                else {
                    String str = sid.substring(3);
                    id = "gys" + (Integer.parseInt(str) + 1);
                }
            }
            TbGysinfo gysInfo = new TbGysinfo(); //创建供应商实体对象
            gysInfo.setId(id); //初始化供应商对象
            gysInfo.setAddress(diZhiF.getText().trim());
            gysInfo.setBianma(bianMaF.getText().trim());
            gysInfo.setFax(chuanZhenF.getText().trim());
            gysInfo.setYh(yinHangF.getText().trim());
            gysInfo.setJc(jianChengF.getText().trim());
            gysInfo.setName(quanChengF.getText().trim());
            gysInfo.setLian(lianXiRenF.getText().trim());
            gysInfo.setLtel(lianXiRenDianHuaF.getText().trim());
        }
    }
}
```

```

        gysInfo.setMail(EMailF.getText().trim());
        gysInfo.setTel(dianHuaF.getText().trim());
        Dao.addGys(gysInfo); //调用addGys()方法存储供应商
        JOptionPane.showMessageDialog(GysTianJiaPanel.this, "已成功添加客户",
            "客户添加信息", JOptionPane.INFORMATION_MESSAGE);
        resetButton.doClick(); //触发“重填”按钮的单击动作
    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}
}

```

1.7.4 供应商修改与删除实现过程

 供应商删除与修改使用的数据表：tb_gysinfo。

(1) 创建 GysXiuGaiPanel 类，用于实现本系统的供应商修改功能。在程序界面中有多个用于输入供应商信息的文本框，这些文本框的内容会根据所选供应商自动填充内容，修改部分或全部内容后，单击“修改”按钮将修改供应商数据。界面中定义的主要控件如表 1.7 所示。

表 1.7 供应商修改与删除界面中的主要控件

控件类型	控件名称	主要属性设置	用途
JtextField	quanChengF	无	供应商全称
	jianChengF	无	简称
	bianMaF	无	邮政编码
	diZhiF	无	地址
	dianHuaF	无	电话
	chuanZhenF	无	传真
	lianXiRenF	无	联系人
	lianXiRenDianHuaF	无	联系人电话
	yinHangF	无	开户银行
	EMailF	无	电子信箱
JComboBox	Gys	设置初始大小为 (230, 21) 调用 initComboBox()方法初始化下拉列表 设置组件的选择事件调用 doGysSelectAction()方法	选择供应商
Jbutton	tjButton	设置按钮文本为“修改” 设置动作监听器为 ModifyActionListener 类的实例对象	修改供应商信息
	resetButton	设置按钮文本为“删除” 设置动作监听器为 DelActionListener 类的实例对象	删除供应商信息

(2) 编写 initComboBox()方法，用于初始化选择供应商的下拉列表框。该方法调用 Dao 类的

getGysInfos()方法获取数据库中所有的供应商信息，然后将供应商的 ID 编号和供应商名称封装成 Item 对象并添加到选择供应商的下拉列表框中，在下拉列表框中 Item 的 toString()方法将显示供应商的名称。initComboBox()方法的关键代码如下：

例程18 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
public void initComboBox() { //初始化供应商下拉列表框的方法
    List gysInfo = Dao.getGysInfos(); //调用getGysInfos()方法获取供应商列表
    List<Item> items = new ArrayList<Item>(); //创建Item列表
    gys.removeAllItems(); //清除下拉列表框中原有的选项
    for (Iterator iter = gysInfo.iterator(); iter.hasNext();) {
        List element = (List) iter.next();
        Item item = new Item(); //封装供应商信息
        item.setId(element.get(0).toString().trim());
        item.setName(element.get(1).toString().trim());
        if (items.contains(item)) //如果 Items 列表中包含该供应商的封装对象
            continue; //跳出本次循环
        items.add(item);
        gys.addItem(item); //否则添加该对象到下拉列表框中
    }
    doGysSelectAction(); //doGysSelectAction() 方法
}
```

(3) 编写 doGysSelectAction()方法，它在更改下拉列表框中的供应商信息时被调用，主要用于根据选择的供应商名称，把供应商的其他信息填充到相应的文本框中。关键代码如下：

例程19 代码位置：光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
private void doGysSelectAction() { //处理供应商选择事件
    Item selectedItem;
    if (!(gys.getSelectedItem() instanceof Item)) {
        return;
    }
    selectedItem = (Item) gys.getSelectedItem(); //获取 Item 对象
    TbGysinfo gysInfo = Dao.getGysInfo(selectedItem); //通过 Item 对象调用 getGysInfo() 方法获取供应商信息
    quanChengF.setText(gysInfo.getName()); //填充供应商信息到文本框中
    diZhiF.setText(gysInfo.getAddress());
    jianChengF.setText(gysInfo.getJc());
    bianMaF.setText(gysInfo.getBianma());
    dianHuaF.setText(gysInfo.getTel());
    chuanZhenF.setText(gysInfo.getFax());
    lianXiRenF.setText(gysInfo.getLian());
    lianXiRenDianHuaF.setText(gysInfo.getLtel());
    EmailF.setText(gysInfo.getMail());
    yinHangF.setText(gysInfo.getYh());
}
```


}

(4) 创建 `ModifyActionListener` 类, 该类是“修改”按钮的事件监听器, 它必须实现 `ActionListener` 接口, 并在 `actionPerformed()` 方法中获取所有文本框内容, 其中包括修改后的信息, 并通过调用 `updateGys()` 方法将这些供应商信息更新到数据库中。关键代码如下:

例程20 代码位置: 光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
class ModifyActionListener implements ActionListener { //修改按钮的事件监听器
    public void actionPerformed(ActionEvent e) {
        Item item = (Item) gys.getSelectedItem();
        TbGysinfo gysInfo = new TbGysinfo(); //创建供应商实体对象
        gysInfo.setId(item.getId()); //初始化供应商实体对象
        gysInfo.setAddress(diZhiF.getText().trim());
        gysInfo.setBianma(bianMaF.getText().trim());
        gysInfo.setFax(chuanZhenF.getText().trim());
        gysInfo.setYh(yinHangF.getText().trim());
        gysInfo.setJc(jianChengF.getText().trim());
        gysInfo.setName(quanChengF.getText().trim());
        gysInfo.setLian(lianXiRenF.getText().trim());
        gysInfo.setLtel(lianXiRenDianHuaF.getText().trim());
        gysInfo.setMail(EMailF.getText().trim());
        gysInfo.setTel(dianHuaF.getText().trim());
        if (Dao.updateGys(gysInfo) == 1) //更新供应商信息
            JOptionPane.showMessageDialog(GysXiuGaiPanel.this, "修改完成");
        else
            JOptionPane.showMessageDialog(GysXiuGaiPanel.this, "修改失败");
    }
}
```

(5) 创建 `DelActionListener` 类, 该类是“删除”按钮的事件监听器, 它必须实现 `ActionListener` 接口, 并在 `actionPerformed()` 方法中获取当前选择的供应商, 然后调用 `Dao` 类的 `delete()` 方法从数据库中把该供应商删除。关键代码如下:

例程21 代码位置: 光盘\TM\01\JXCManager\src\internalFrame\gysGuanLi\GysXiuGaiPanel.java

```
class DelActionListener implements ActionListener { //删除按钮的事件监听器
    public void actionPerformed(ActionEvent e) {
        Item item = (Item) gys.getSelectedItem(); //获取当前选择的供应商
        if (item == null || !(item instanceof Item))
            return;
        int confirm = JOptionPane.showConfirmDialog( //弹出确认删除对话框
            GysXiuGaiPanel.this, "确认删除供应商信息吗? ");
        if (confirm == JOptionPane.YES_OPTION) { //如果确认删除
            int rs = Dao.delete("delete tb_gysInfo where id=" //调用delete()方法
                + item.getId() + "");
            if (rs > 0) {
                JOptionPane.showMessageDialog(GysXiuGaiPanel.this, //显示删除成功对话框
                    "供应商: " + item.getName() + "。删除成功");
                gys.removeItem(item);
            }
        }
    }
}
```

```
        } else {  
            JOptionPane.showMessageDialog(GysXiuGaiPanel.this,  
                "无法删除客户: " + item.getName() + "。");  
        }  
    }  
}
```

1.7.5 单元测试

在现代软件开发过程中，测试不再作为一个独立的生命周期，单元测试成为与编写代码同步进行的开发活动。单元测试能够提高程序员对程序的信心，保证程序的质量，加快软件开发速度，使程序易于维护。

1 · 单元测试概述

单元测试是在软件开发过程中要进行的最低级别的测试活动，在单元测试活动中，软件的独立工作单元将在与程序的其他部分相隔离的情况下进行测试。

在一种传统的结构化编程语言中，如 Java 语言，要进行测试的工作单元一般是方法。在像 C++ 这样的面向对象的语言中，要进行测试的基本单元是类。单元测试不仅仅是作为无错编码的一种辅助手段，在一次性的开发过程中使用，单元测试还必须是可重复的，无论是在软件修改或是移植到新的运行环境的过程中。因此，所有的测试都必须在整个软件系统的生命周期中进行。

2 · 什么是单元测试

☑ 它是一种验证行为。

程序中的每一项功能都可以通过单元测试来验证其正确性。它为以后的开发提供支持。就算是开发后期，也可以轻松地增加功能或更改程序结构，而不用担心这个过程中会破坏重要的东西。而且它为代码的重构提供了保障。这样，我们就可以更自由地对程序进行改进。

☑ 它是一种设计行为。

编写单元测试将使我们从调用者的角度观察、思考。特别是先写测试（test-first），迫使我们把程序设计成易于调用和可测试的，即迫使我们解除软件中的耦合。

☑ 它是一种编写文档的行为。

单元测试是一种无价的文档，它是展示函数或类如何使用的最佳文档。这份文档是可编译、可运行的，它永远保持与代码同步。

3 · 越到项目后期，单元测试为何越难进行

在很多项目的初期，项目中的大部分程序员都能够自觉地编写单元测试。随着项目的进展、任务的加重，离交付时间越来越近，不能按时完成项目的风险越来越大，单元测试就往往成为牺牲品了。项目经理因为进度的压力也不重视了，程序员也因为编码的压力和无人看管而不再为代码编写单元测试了。笔者亲身经历的项目都或多或少地发生过类似这样的事情。越是在项目的后期，能够坚持编写单元测试的程序员在整个项目组中所占比例越来越低。

为了追赶项目进度，多数程序员将没有经过任何测试的程序代码上传到版本控制系统，项目经理

也不再追问，照单全收。这样做的结果就是在项目后期，技术骨干人员只好加班加点进行系统集成。集成完了之后，下发给测试人员测试时，Bug 的报告数量翻倍增长。程序员开始修改 Bug，但有非常多的 Bug 隐藏得很深，一直潜伏到生产环境中去。

4 · JUnit 单元测试工具的介绍与使用

☒ JUnit 使用介绍

JUnit 是一个单元测试框架，专门用于测试 Java 开发的程序，同类产品还包括 NUnit(.Net)，CPPUnit(C++)，都属于 xUnit 中的成员。目前 JUnit 的最新版本是 JUnit 4.0，在 Eclipse 开发工具中已经集成了 JUnit 的多个版本。

在正式讲解 JUnit 之前，先来看一下单元测试的运行效果，如图 1.23 和图 1.24 所示。

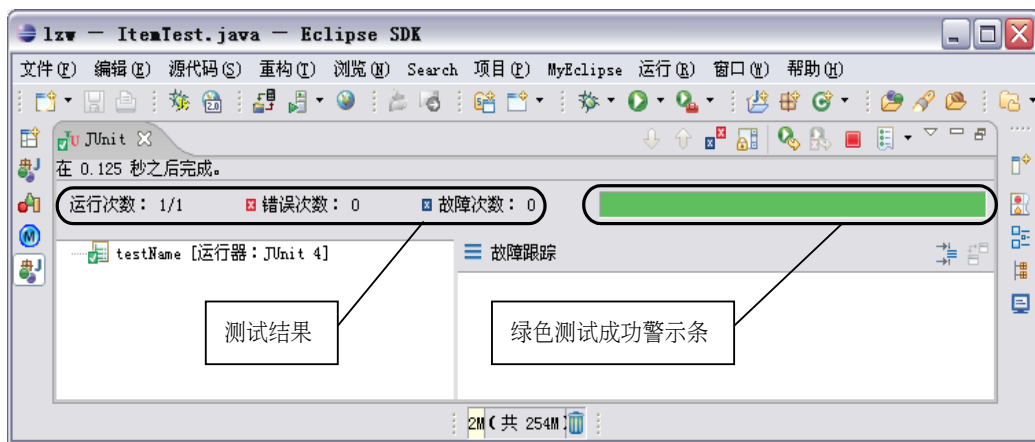


图 1.23 单元测试通过效果

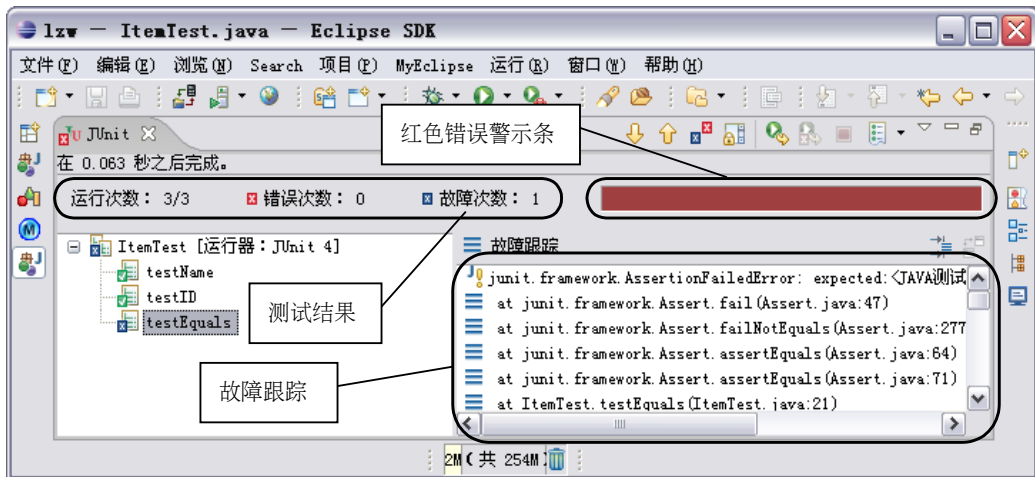


图 1.24 单元测试失败效果

在图 1.23 和图 1.24 中很容易发现不同颜色的警示条，图 1.23 所示是绿色的，图 1.24 所示是红色的。如果所有测试案例运行成功，就为绿色；反之，如果有一个不成功，则为红色。

☒ 使用 JUnit 进行单元测试

下面开始按步骤讲解如何在 Eclipse 中使用 JUnit 工具。

(1) 为单元测试代码创建一个 Java 项目，将其命名为 JUnitTest。

(2) 创建 ItemTest 类，该类用于测试公共类 Item 的行为（即方法）。在“创建 Java 类”对话框中设置该类的超类为 TestCase，也就是继承 JUnit 框架的测试用例编写单元测试，单击“完成”按钮，如图 1.25 所示。

(3) 在项目的构建路径中添加 JUnit 类库。右击项目名称，在弹出的快捷菜单中选择“构建路径”/“添加库”命令，在弹出的“添加库”对话框中选择 Junit 选项，单击“下一步”按钮，如图 1.26 所示。

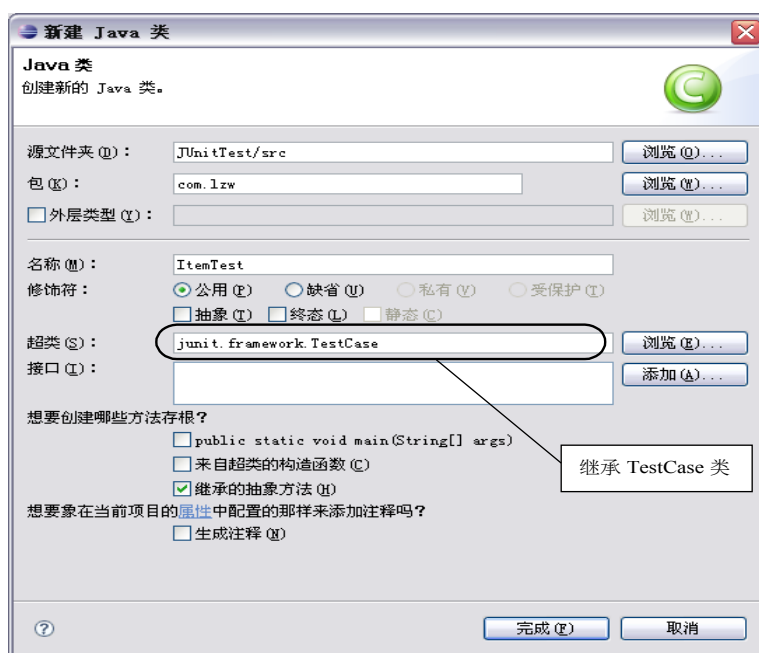


图 1.25 新建测试用例类

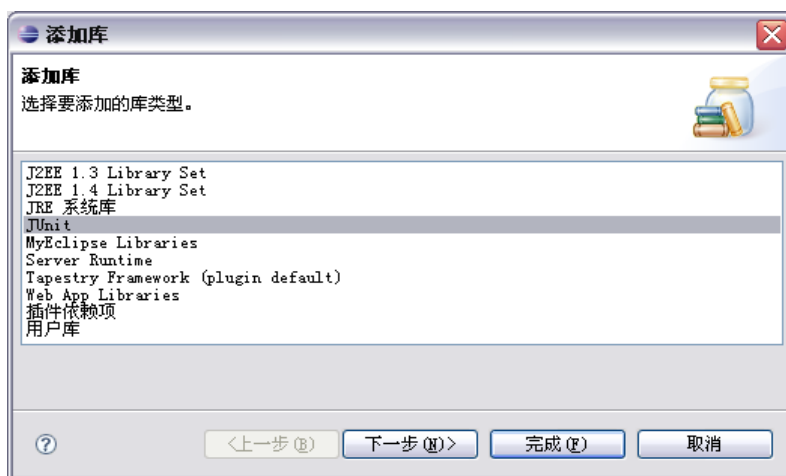


图 1.26 添加库对话框

(4) 在弹出的对话框中选择 JUnit 的版本为 JUnit 4，单击“完成”按钮。

(5) 在创建的 ItemTest 中，对 Item 类进行单元测试。Item 是本系统的公共类之一，要实现该类的单元测试，需要编写以 test 作方法名称的前缀，创建 testName()方法、testID()方法和 testEquals()方法。另外还要重写父类的 setUp()方法，在该方法中创建并初始化测试用例中需要的数据。

完整代码如下：

```
import internalFrame.guanli.Item;
import junit.framework.TestCase;
public class ItemTest extends TestCase{
    private Item item;
    protected void setUp() throws Exception {
        item=new Item();
        item.setId("007");
        item.setName("JAVA测试");
    }
    public void testID(){
        assertEquals(item.getId(), "007");
    }
    public void testName(){
        assertEquals(item.getName(), "JAVA测试");
    }
    public void testEquals(){
        Item newItem=new Item();
        newItem.setId("007");
        newItem.setName("JAVA测试");
        assertEquals(item, newItem);
    }
}
```

■ 代码贴士

- TestCase：该类是 JUnit 框架的测试用例类，所有的单元测试都需要继承该类。
- setUp()：该方法将在单元测试之前，为本类的所有单元测试提供测试数据。
- testID()：该方法用于测试 Item 类的 getId()方法。
- testName()：该方法用于测试 Item 类的 getName()方法。
- testEquals()：该方法用于测试 Item 类的相等性。

(6) 在该类上单击鼠标右键，在弹出的快捷菜单中选择“运行方式”/“JUnit 测试”命令，运行 Item 类的单元测试，根据警示条中的颜色，即可判断单元测试的成功与失败，如图 1.24 所示。因为在本系统中不需要判断 Item 实例的相等性，所以 Item 类没有实现父类的 equals()方法，可以不进行该测试，否则在判断两个 Item 类的实例对象是否相等时，将出现判断失败。

1.8 进货管理模块设计

进货管理模块是进销存管理系统中不可缺少的重要组成部分，它主要负责为系统记录进货单及其退货信息，相应的进货商品会添加到库存管理中。

1.8.1 进货管理模块概述

企业进销存管理系统中的进货管理模块主要包括进货单和进货退货两个部分。由于它们的实现方法基本相似，本节将以进货单功能为主，介绍进货管理模块对本系统的意义和实现的业务逻辑。

1· 进货单

进货单功能主要负责记录企业的商品进货信息，可以单击“添加”按钮，在商品表中添加进货的商品信息。在“供应商”下拉列表框中选择合适的供应商，将会改变商品表中可以添加的商品。进货单的程序界面如图 1.27 所示。

商品名称	商品编号	产地	单位	规格	包装	单价	数量	批号
木吉地	sp1002	中**海	把	H#2100	盒	120	3	32286
碧**莉香...	sp1003	厂**保洁公司	袋	350 g	塑包	2	100	2005091101
铅笔	sp1007	长春	铅笔厂	9*	沿看	0.5	100	14**

图 1.27 进货单程序界面

2· 进货退货

进货退货功能主要负责记录进货管理中的退货信息，界面效果如图 1.28 所示。在选择退货的商品之后，单击“退货”按钮，将把表格中的商品退货信息更新到数据库中。



图 1.28 进货退货程序界面

1.8.2 进货管理模块技术分析

进货管理模块使用 JDBC 实现事务操作，因为进货和退货的业务逻辑涉及到 3 个数据表，为保证数据的完整性，将 3 个数据表的操作放在事务中实现，如果对任何一个数据表的操作出现错误或是不可执行的操作，那么整个事务中的所有操作都将取消，并恢复到事务执行之前的数据状态；否则 3 个数据表的操作全部执行。下面介绍使用 JDBC 实现事务操作的关键方法。

1 · setAutoCommit()方法

该方法用于设置连接对象的自动提交模式。如果连接处对象的自动提交模式为 `True`，则它的所有 SQL 语句将被执行并作为单个事务提交；否则，该连接对象的 SQL 语句将聚集到事务中，直到调用 `commit()`方法或 `rollback()`方法为止。默认情况下，新连接的自动提交模式为 `True`。

语法：

```
void setAutoCommit(boolean autoCommit)
```

`autoCommit`: 该参数为 `True` 表示启用连接对象的自动提交模式；为 `False` 表示禁用连接对象的自动提交模式。

2 · getAutoCommit()

判断此连接对象是否启用了自动提交模式。

语法：

```
boolean getAutoCommit()
```

3 · commit()方法

该方法将执行提交 SQL 语句执行数据库操作，并释放此连接对象当前持有的所有数据库锁。此方

法只在禁用自动提交模式情况下使用。

语法:


```
void commit()
```

4 · rollback()方法

该方法将取消在当前事务中进行的所有更改，并释放此连接对象当前持有的所有数据库锁。此方法只在禁用自动提交模式情况下使用。

```
void rollback()
```

1.8.3 进货单实现过程

 进货单使用的数据表：tb_ruku_main、tb_ruku_detail、tb_kucun。

(1) 创建 JinHuoDan 类，用于实现本系统的进货单功能的界面和业务逻辑。界面中定义的主要控件如表 1.8 所示。

表 1.8 进货单界面中的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用 途
JTextField	PiaoHao	设置该控件不接受输入焦点	进货单票号
	Pzs	设置该控件不接受输入焦点	品种数
	Hpzs	设置该控件不接受输入焦点	货品总数
	Hjje	设置该控件不接受输入焦点	合计金额
	Ysjl	无	验收结论
	Czy	设置该控件不接受输入焦点	操作员
	Jhsj	设置该控件不接受输入焦点	进货时间
	Jsr	无	经手人
	Lian	设置该控件不接受输入焦点	联系人
JComboBox	Jsfs	添加“现金”和“支票”两个下拉列表项	结算方式
	Gys	从数据库中获取供应商并添加到下拉列表中	供应商
	Sp	从数据库中获取商品并添加到下拉列表中	商品
JTable	Table	调用initTable()方法初始化表格 取消表格列大小的自动调整	商品表格
JButton	TjButton	设置按钮文本为“添加” 设置动作监听器为TjActionListener类的实例对象	添加
	RkButton	设置按钮文本为“入库” 设置动作监听器为RkActionListener类的实例对象	入库

(2) 编写 `initTable()` 方法, 该方法用于初始化商品表格的表头、列编辑器等。设置表格中第一个列的编辑器使用下拉列表框样式的编辑器, 通过该编辑器选择商品的名称, 其他的商品信息将自动填充。关键代码如下:

例程22 代码位置: 光盘\TM\01\JXCManager\src\internalFrame\JinHuoDan.java

```
private void initTable() { //初始化表格
    String[] columnNames = {"商品名称", "商品编号", "产地", "单位", "规格", "包装", "单价",
        "数量", "批号", "批准文号"};
    ((DefaultTableModel) table.getModel())
        .setColumnIdentifiers(columnNames); //设置表格的表头
    TableColumn column = table.getColumnModel().getColumn(0); //获取第一个表格列
    final DefaultCellEditor editor = new DefaultCellEditor(sp); //创建表格列编辑器
    editor.setClickCountToStart(2);
    column.setCellEditor(editor);
}
```

(3) 编写 `initSpBox()` 方法, 该方法用于初始化表格中的商品下拉列表框。它首先调用 `Dao` 类的 `query()` 方法获取指定供应商所提供的所有商品信息, 然后将这些商品信息封装成商品对象, 并把这些对象添加到商品下拉列表框中。关键代码如下:

例程23 代码位置: 光盘\TM\01\JXCManager\src\internalFrame\JinHuoDan.java

```
private void initSpBox() { //初始化商品下拉列表框
    List list = new ArrayList();
    ResultSet set = Dao.query("select * from tb_spinfo where gysName="
        + gys.getSelectedItemId() + ""); //调用query()方法
    sp.removeAllItems();
    sp.addItem(new TbSpinfo());
    for (int i = 0; table != null && i < table.getRowCount(); i++) {
        TbSpinfo tmpInfo = (TbSpinfo) table.getValueAt(i, 0);
        if (tmpInfo != null && tmpInfo.getId() != null)
            list.add(tmpInfo.getId());
    }
    try {
        while (set.next()) {
            TbSpinfo spinfo = new TbSpinfo(); //创建商品对象
            spinfo.setId(set.getString("id").trim()); //初始化商品对象
            //如果表格中已存在同样商品, 商品下拉列表框中就不再包含该商品
            if (list.contains(spinfo.getId()))
                continue;
            spinfo.setSpname(set.getString("spname").trim()); //封装商品信息
            spinfo.setCd(set.getString("cd").trim());
            spinfo.setJc(set.getString("jc").trim());
            spinfo.setDw(set.getString("dw").trim());
            spinfo.setGg(set.getString("gg").trim());
            spinfo.setBz(set.getString("bz").trim());
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```

▪        spinfo.setPh(set.getString("ph").trim());
▪        spinfo.setPzwh(set.getString("pzwh").trim());
▪        spinfo.setMemo(set.getString("memo").trim());
▪        spinfo.setGysname(set.getString("gysname").trim());
        sp.addItem(spinfo);                                //将商品对象添加到下拉列表框
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

■ 代码贴士

- setSpname ()：该方法用于设置商品实体类的商品名称。
- setCd ()：该方法用于设置商品实体类的商品产地。
- setJc ()：该方法用于设置商品实体类的商品简称。
- setDw ()：该方法用于设置商品实体类的商品单位。
- setGg ()：该方法用于设置商品实体类的商品规格。
- setBz ()：该方法用于设置商品实体类的商品包装。
- setPh ()：该方法用于设置商品实体类的商品批号。
- setPzwh ()：该方法用于设置商品实体类的商品批准文号。
- setMemo ()：该方法用于设置商品实体类的商品简介信息。
- setGysname ()：该方法用于设置商品实体类的供应商名称。

(4) 编写“入库”按钮的事件监听器 `RkActionListener` 类，该类必须实现 `ActionListener` 接口和接口中的 `actionPerformed()` 方法，并在 `actionPerformed()` 方法中获取界面中的商品表格数据，然后将这些数据封装到进货数据表的实体对象中，最后调用 `Dao` 类的 `insertRukuInfo()` 方法在事务中保存进货单数据。关键代码如下：

例程24 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JinHuoDan.java

```

class RkActionListener implements ActionListener {           //入库按钮的事件监听器
    public void actionPerformed(ActionEvent e) {
        stopTableCellEditing();                             //结束表格中没有编写的单元
        clearEmptyRow();                                     //清除空行
        String hpzsStr = hpzs.getText();                     //货品总数
        String pzsStr = pzs.getText();                       //品种数
        String jeStr = hjje.getText();                       //合计金额
        String jsfsStr = jsfs.getSelectedItem().toString(); //结算方式
        String jsrStr = jsr.getText().trim();                 //经手人
        String czyStr = czy.getText();                       //操作员
        String rkDate = jhsjDate.toLocaleString();           //入库时间
        String ysjlStr = ysjl.getText().trim();              //验收结论
        String id = piaoHao.getText();                       //票号
        String gysName = gys.getSelectedItem().toString();  //供应商名称
        if (jsrStr == null || jsrStr.isEmpty()) {
            JOptionPane.showMessageDialog(JinHuoDan.this, "请填写经手人");
            return;
        }
    }
}

```

```

    }
    if (ysjlStr == null || ysjlStr.isEmpty()) {
        JOptionPane.showMessageDialog(JinHuoDan.this, "添写验收结论");
        return;
    }
    if (table.getRowCount() <= 0) {
        JOptionPane.showMessageDialog(JinHuoDan.this, "添加入库商品");
        return;
    }
    TbRukuMain ruMain = new TbRukuMain(id, pzsStr, jeStr, ysjlStr,
        gysName, rkDate, czyStr, jsrStr, jsfsStr); //创建入库主表实体对象
    Set<TbRukuDetail> set = ruMain.getTabRukuDetails(); //获取入库主表的详细表集合
    int rows = table.getRowCount();
    for (int i = 0; i < rows; i++) {
        TbSpinfo spinfo = (TbSpinfo) table.getValueAt(i, 0); //获取商品对象
        String djStr = (String) table.getValueAt(i, 6); //获取商品的单价
        String slStr = (String) table.getValueAt(i, 7); //获取商品的数量
        Double dj = Double.valueOf(djStr);
        Integer sl = Integer.valueOf(slStr);
        TbRukuDetail detail = new TbRukuDetail(); //创建商品详细表实体对象
        detail.setTabSpinfo(spinfo.getId()); //初始化商品详细表对象
        detail.setTabRukuMain(ruMain.getRkId());
        detail.setDj(dj);
        detail.setSl(sl);
        set.add(detail);
    }
    boolean rs = Dao.insertRukuInfo(ruMain); //调用insertRukuInfo()方法保存进货单
    if (rs) {
        JOptionPane.showMessageDialog(JinHuoDan.this, "入库完成");
        DefaultTableModel dftm = new DefaultTableModel();
        table.setModel(dftm);
        initTable();
        pzs.setText("0");
        hpzs.setText("0");
        hjje.setText("0");
    }
}
}

```

1.9 查询统计模块设计

查询统计模块是进销存管理系统中不可缺少的重要组成部分，它主要包括基础信息、进货信息、销售信息、退货信息的查询和销售排行功能。

1.9.1 查询统计模块概述

企业进销存管理系统中的查询统计模块包括客户查询、商品查询、供应商查询、销售查询、销售退货查询、入库查询、入库退货查询和销售排行功能。由于本书的篇幅所限，本节将以销售查询功能为主，介绍查询统计模块对本系统的意义和实现的业务逻辑。

销售查询功能主要用于查询系统中的销售信息，其查询方式可以按照客户全称、销售票号进行匹配查询和模糊查询。另外，还可以指定销售日期查询。程序界面如图 1.29 所示。



图 1.29 销售查询界面

1.9.2 查询统计模块技术分析

查询统计模块主要以丰富的查询条件为主要技术，当查询一个商品销售或者退货等信息时，需要提供按客户全称、销售票号、退货票号、指定日期等多种查询条件和查询对象，进行普通查询或者模糊查询。对于普通查询条件可以简单地使用 SQL 语句的“=”进行判断，但是模糊查询稍微复杂一些，需要使用 SQL 语句中的 LIKE 关键字。LIKE 关键字需要使用通配符在字符串内查找指定的模式，所以读者需要了解通配符及其含义。通配符的含义如表 1.9 所示。

表 1.9 LIKE 关键字中的通配符及其含义

通 配 符	说 明
%	由零个或多个字符组成的任意字符串
_	任意单个字符
[]	用于指定范围，例如[A~F]，表示A~F范围内的任何单个字符
[^]	表示指定范围之外的，例如[^A~F]，表示A~F范围以外的任何单个字符

☒ “%”通配符

“%”通配符能匹配0个或多个字符的任意长度的字符串。

☒ “_”通配符

“_”号表示任意单个字符，该符号只能匹配一个字符，利用“_”号可以作为通配符组成匹配模式进行查询。


☒ “[]”通配符

在模糊查询中可以使用 “[]”符号来查询一定范围内的数据。“[]”符号用于表示一定范围内的任意单个字符，它包括两端数据。

☒ “[^]”通配符

在模式查询中可以使用 “[^]”符号来查询不在指定范围内的数据。“[^]”符号用于表示不在某范围内的任意单个字符，它包括两端数据。

1.9.3 销售查询实现过程

 销售查询使用的数据表：v_sellView。

(1) 创建 XiaoShouChaXun 类，用于实现本系统的销售查询功能界面和业务逻辑。界面中定义的主要控件如表 1.10 所示。

表 1.10 进货单界面中的主要控件

控件类型	控件名称	主要属性设置	用途
JtextField	StartDate	设置文本内容为当年的1月1日	起始日期
	EndDate	设置文本内容为当前日期	截止日期
	Content	添加按键监听器，当按下Enter键时执行查询	查询内容
续表			
控件类型	控件名称	主要属性设置	用途
JcomboBox	Operation	添加“等于”和“包含”两项内容	条件方式
	Condition	添加“客户全称”和“销售票号”两项内容	查询条件
Jtable	Table	设置表头 取消表格列大小的自动调整	商品表格
Jbutton	queryButton	设置按钮文本为“查询” 设置动作监听器为QueryActionListener类的实例对象	查询
	showAllButton	设置按钮文本为“显示全部数据” 设置动作监听器为ShowAllActoinListener类的实例对象	显示全部数据

(2) 编写 updateTable ()方法，用于更新表格数据。该方法必须接收一个 Iterator 迭代器对象，通过遍历该迭代器中的数据来初始化界面中的表格。关键代码如下：

例程25 代码位置：光盘\TM\01\JXCManager\src\internalFrame\XiaoShouChaXun.java

```
private void updateTable(Iterator iterator) { //更新表格数据
```

```

int rowCount=dfm.getRowCount();
for(int i=0;i<rowCount;i++) {                                //清除原内容
    dfm.removeRow(0);
}
while(iterator.hasNext()) {                                  //更新表格数据
    Vector vector=new Vector();
    List view=(List) iterator.next();
    vector.addAll(view);
    dfm.addRow(vector);
}
}

```

(3) 创建 `ShowAllActoinListener` 类，使该类实现 `ActionListener` 接口，并实现该接口的 `actionPerformed()` 方法。该方法在用户单击“显示全部数据”按钮时，执行无条件的数据查询，也就是说，该按钮将读取数据库中所有的销售信息，并显示到表格中。关键代码如下：

例程26 代码位置：光盘\TM\01\JXCManager\src\internalFrame\XiaoShouChaXun.java

```

class ShowAllActoinListener implements ActionListener {        //“显示全部数据”按钮的动作监听器
    public void actionPerformed(final ActionEvent e) {
        content.setText("");
        List list=Dao.findForList("select * from v_sellView"); //调用findForList()方法执行查询
        Iterator iterator=list.iterator();
        updateTable(iterator);                                //调用updateTable()方法更新表格
    }
}

```

(4) 创建“查询”按钮的事件监听器 `QueryActionListener` 类，该类必须实现 `ActionListener` 接口，并实现该接口的 `actionPerformed()` 方法。在该方法中编写查询销售信息的业务逻辑，并将查询结果更新到表格控件中，其查询条件由 `condition`、`operation` 下拉列表框和一个 `content` 文本框组成。关键代码如下：

例程27 代码位置：光盘\TM\01\JXCManager\src\internalFrame\XiaoShouChaXun.java

```

class QueryActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        boolean selDate = selectDate.isSelected();
        if(content.getText().equals("")) {
            JOptionPane.showMessageDialog(getContentPane(), "请输入查询内容！");
            return;
        }
        if(selDate) {
            if(startDate.getText()==null||startDate.getText().equals("")) {
                JOptionPane.showMessageDialog(getContentPane(), "请输入查询的开始日期！");
                return;
            }
            if(endDate.getText()==null||endDate.getText().equals("")) {
                JOptionPane.showMessageDialog(getContentPane(), "请输入查询的结束日期！");
                return;
            }
        }
    }
}

```

```

    }
    List list=null;
    String con = condition.getSelectedIndex() == 0           //获取查询字段
        ? "khname "
        : "sellId ";
    int oper = operation.getSelectedIndex();                //定义查询方式
    String opstr = oper == 0 ? "=" : "like ";
    String cont = content.getText();                       //获取查询内容
    list = Dao.findForList("select * from v_sellView where " //调用findForList()方法查询数据
        + con
        + opstr
        + (oper == 0 ? ""+cont+"" : "%" + cont + "%")
        + (selDate ? " and xsdate>" + startDate.getText()
            + " and xsdate<=" + endDate.getText()+" 23:59:59" : ""));
    Iterator iterator = list.iterator();
    updateTable(iterator);                                //调用updateTable()方法更新表格
}
}

```

1.10 库存管理模块设计

1.10.1 库存管理模块概述

企业进销存管理系统中的库存管理模块包括库存盘点和价格调整两个功能。由于本书的篇幅所限，本节将以价格调整功能为主，介绍库存管理模块对本系统的意义和实现的业务逻辑。

价格调整功能主要用于调整库存中指定商品的单价，当用户选择了指定的商品，价格调整功能的界面会显示该商品在库存中的单价、库存数量、库存金额、单位、产地等信息。程序界面如图 1.30 所示。用户可以修改商品价格并单击“确定”按钮，调整该商品在库存中的单价。



图 1.30 价格调整界面

1.10.2 库存管理模块技术分析

企业进销存管理系统中的库存管理模块包括库存盘点和价格调整两个功能，其中库存盘点涉及的技术比较简单，它将库存信息显示在表格中，由操作员输入盘点的商品数量，然后程序自动计算损益值。价格调整功能涉及下拉列表框的选择事件监听和事件处理技术，这在使用 Java Swing 技术进行程序开发的过程中，非常重要。为防止用户的错误输入，程序界面经常需要将可枚举的输入内容封装在下拉列表框中，限制用户的输入。但是，要知晓下拉列表框的改变，还需要为下拉列表框添加相应的事件监听器。下面就来介绍一下相关的语法。

`addItemListener()`方法可以为下拉列表框添加 `ItemListener` 监听器。当更改下拉列表框中的选项时，将产生相应的事件，这个事件会被添加的 `ItemListener` 监听器捕获，并处理相应的业务逻辑。


语法：

```
public void addItemListener(ItemListener aListener)
```

参数：

`aListener`：要通知的 `ItemListener` 监听器

1.10.3 价格调整实现过程

 价格调整使用的数据表：tb_kucun。

(1) 创建 `JiaGeTiaoZheng` 类，用于实现本系统的价格调整功能界面和业务逻辑。界面中定义的主要控件如表 1.11 所示。

表 1.11 进货单界面中的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用 途
JTextField	KuCunJinE	取消编辑状态	库存金额
	KuCunShuLiang	取消编辑状态	库存数量
	DanJia	添加按键监听器，当输入改变时调用 <code>updateJinE()</code> 方法更新库存金额	库存单价
JComboBox	ShangPinMingCheng	读取库存表数据并初始化该控件内容	商品名称
JLabel	GuiGe	设置前景色为蓝色	规格
	ChanDi	设置前景色为蓝色	产地
	JianCheng	设置前景色为蓝色	简称
	BaoZhuang	设置前景色为蓝色	包装
	DanWei	设置前景色为蓝色	单位
Jbutton	OkButton	设置按钮文本为“确定” 设置动作监听器为 <code>OkActionListener</code> 类的实例对象	确定

	CloseButton	设置按钮文本为“关闭” 设置动作监听器为 CloseActionListener 类的实例对象	关闭
--	-------------	--	----

(2) 编写 `updateJinE()` 方法，用于更新库存金额。该方法将“单价”文本框的内容转换为 `Double` 类型，将“库存数量”文本框的内容转换为 `Integer` 类型，然后用它们的乘积更新“库存金额”文本框的内容。关键代码如下：

例程28 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
private void updateJinE() {           //更新库存金额的方法
    Double dj = Double.valueOf(danJia.getText());
    Integer sl = Integer.valueOf(kuCunShuLiang.getText());
    kuCunJinE.setText((dj * sl) + "");
}
```

(3) 创建 `ItemActionListener` 类，它必须实现 `ItemListener` 接口和接口中的 `itemStateChanged()` 方法，成为下拉列表框的事件监听器。当改变界面中选择的商品时，相应的 `ItemEvent` 事件会通知该监听器处理业务逻辑，也就是根据选择的商品名称更新其他控件内容。关键代码如下：

例程29 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
class ItemActionListener implements ItemListener {           //商品选择事件监听器
    public void itemStateChanged(
        final ItemEvent e) {
        Object selectedItem = shangPinMingCheng.getSelectedItem();           //获取选择的商品对象
        if (selectedItem == null)
            return;
        Item item = (Item) selectedItem;
        kcInfo = Dao.getKucun(item);           //调用getKucun()方法
        if (kcInfo.getId() == null)
            return;
        int dj, sl;
        dj = kcInfo.getDj().intValue();
        sl = kcInfo.getKcsl().intValue();
        chanDi.setText(kcInfo.getCd());           //更新界面控件的内容
        jianCheng.setText(kcInfo.getJc());
        baoZhuang.setText(kcInfo.getBz());
        danWei.setText(kcInfo.getDw());
        danJia.setText(kcInfo.getDj() + "");
        kuCunShuLiang.setText(kcInfo.getKcsl() + "");
        kuCunJinE.setText(dj * sl + "");
        guiGe.setText(kcInfo.getGg());
    }
}
```

■ 代码贴士

`ItemListener`：下拉列表框的事件监听器必须实现的分接口。

`ItemStateChanged()`：当下拉列表框的选中项发生改变时将触发该方法。

▪ ItemEvent：这是选项事件类，在用户更改带有多项选择内容的组件选项时，将产生该事件。例如下拉选择框组件。

(4) 创建 OkActionListener 类，它必须实现 ActionListener 接口和接口中的 actionPerformed() 方法，在这个方法中获取新的库存商品价格，然后调用 Dao 类的 updateKucunDj() 方法更新库存价格。关键代码如下：

例程30 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
class OkActionListener implements ActionListener {
    public void actionPerformed(final ActionEvent e) {
        kcInfo.setDj(Double.valueOf(danJia.getText()));
        kcInfo.setKcsl(Integer.valueOf(kuCunShuLiang.getText()));
        int rs = Dao.updateKucunDj(kcInfo);
        if (rs > 0)
            JOptionPane.showMessageDialog(getContentPane(), "价格调整完毕。",
                kcInfo.getSpname() + "价格调整",
                JOptionPane.QUESTION_MESSAGE);
    }
}
```

1.10.4 单元测试

在价格调整界面中输入单价时，如果输入“1133”程序将抛出 NumberFormatException 异常，如图 1.31 所示。这是因为输入单价的数字格式不对，注意输入值“1133”的第二个“l”字符并不是数字，而是英文字母 L 的小写形式，字母当然不能用作数字，所以产生了这个错误，导致程序无法执行价格调整。

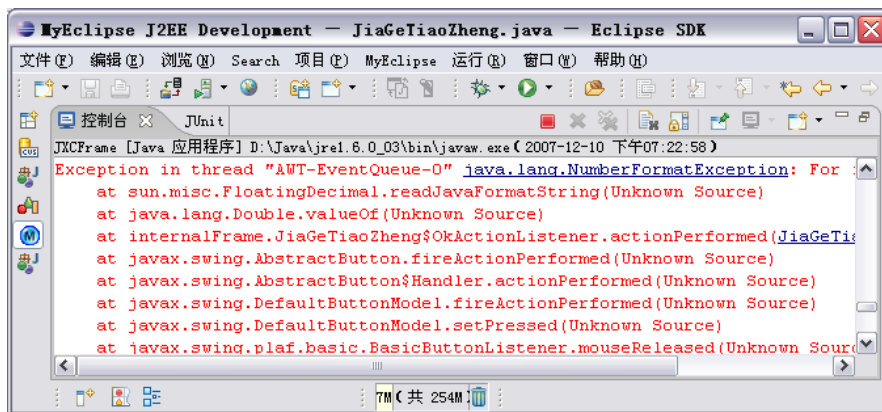


图 1.31 非数字单价产生的错误

解决这一问题的方法是在执行价格调整之前，对输入的单价进行数字格式验证。可是，非要等操作人员输入单价之后，再验证输入单价的正确与否吗？如果利用按键监听器，监听“单价”文本框中的

每一次按键，当按键是数字时，继续接收输入；反之，当按键不是数字或小数点时（那它就应该是字母或其他的什么，反正不是数字）就取消本次按键的输入。这样在用户输入时，就能够有效地屏蔽非数字格式的输入，它比之前的数字格式验证更有效。关键代码如下：

例程31 代码位置：光盘\TM\01\JXCManager\src\internalFrame\JiaGeTiaoZheng.java

```
danJia.addKeyListener(new KeyAdapter() {           //添加按键监听器
    public void keyTyped(KeyEvent e) {
        String numStr = "0123456789." + (char) 8; //数字格式的字符串，其中(char)8是回退键用于删除字符
        if (numStr.indexOf(e.getKeyChar()) < 0)    //如果按键字符不在数字格式字符串中
            e.consume();                           //销毁按键对象
        else                                       //否则
            updateJinE();                          //更新库存金额
    }
});
```

1.11 系统打包发布

Java 应用程序可以打包成 JAR 文件，JAR 文件是一个简单的 ZIP 格式的文件，它包含程序中的类文件和执行程序的其他资源文件。在程序发布之前，需要将所有编译好的 Java 文件封装到一个程序打包文件中，然后将这个程序的打包文件提交给客户使用。一旦程序打包之后，就可以使用简单的命令来执行它。另外，如果配置好 Java 环境或使用 JDK 的安装程序构建 Java 环境，那么就可以像运行本地可执行文件一样去执行 JAR 文件。本节将介绍如何使用 Eclipse 开发工具将程序打包成 JAR 文件。

(1) 创建描述文件。JAR 文件需要一个描述文件，该文件以 MANIFEST.MF 命名，它描述了 JAR 的配置信息，例如指定主类名称、类路径等。程序代码如下：

▪	Manifest-Version: 1.0	//指定描述文件的版本
▪	Main-Class: com.lzw.JXCFrame	//指定程序主类
▪	Class-Path: . lib\msbase.jar lib\mssqlserver.jar lib\msutil.jar	//配置类路径
▪		//添加空行结尾

代码贴士

描述文件的版本号是每个描述文件的基本信息。

Main-Class 用于指定程序执行的主类。

Class-Path 用于指定程序执行的类路径，多个路径使用“ ”空格符号分割。

在描述文件的结尾插入一个空行，这代表描述文件的结束。

注意：在“:”符号和后面的定义值之间一定要有一个“ ”空格作分隔符，否则程序会因为无法识别而导致程序出错。

(2) 在 Eclipse 的资源包管理器中右击项目的 src 文件夹，在弹出的快捷菜单中选择“导出”命令。

(3) 在弹出的“导出”对话框中选择 Java/“JAR 文件”子节点，单击“下一步”按钮。

(4) 在弹出的“JAR 导出”对话框中选择要导出的文件夹，本系统的程序代码都在 src 文件夹中，在步骤 (2) 中是右击 src 文件夹启动导出功能的，在该对话框中已经默认选取 src 文件夹中的所有内容，包括子文件夹。然后，在“JAR 文件”下拉列表框中输入生成的 JAR 文件名和路径，如图 1.32 所示。单击两次“下一步”按钮。



图 1.32 JAR 导出对话框

(5) 在弹出的对话框中选中“从工作空间中使用现有清单”单选按钮，在“清单文件”文本框的右侧单击“浏览”按钮，选择步骤 (1) 建立的清单文件 MANIFEST.MF，单击“完成”按钮。

(6) 现在 JAR 文件已经创建并保存在 C 盘 product 文件夹中，程序的清单描述文件中指定了连接 SQL Server 2000 数据库的 JDBC 驱动包放在 lib 文件夹中，所以，必须在 product 文件夹中创建 lib 文件夹，然后将相应的类包复制到 lib 文件夹中，最后将本系统所用到的 res 图片资源文件夹复制到 product 文件夹中，就可以双击 JXCManager.jar 文件运行程序了。

1.12 开发技巧与难点分析

本系统使用的是 MDI 窗体模式开发的程序界面，它使用一个主窗体包含多个子窗体，子窗体只能在主窗体规定的范围内移动。这些子窗体由导航面板上的按钮调用，这些按钮需要添加事件监听器，在单击该按钮时，由事件监听器创建并初始化相应的子窗体，然后显示该子窗体。

如果为每个按钮创建新的事件监听器对象，那至少需要 20 个事件监听器类，因为导航面板上定义的按钮总数和子窗体的数量是对应的，而子窗体的数量正好是 20 个，所以需要定义相应数量的按钮和事件监听器，这些繁琐的工作会占用大量的程序开发时间，影响工程进度。

从不同的按钮监听器所实现的业务逻辑中不难发现，它们所完成的工作基本相同，都是创建并初始化子窗体，然后显示它们。如果它们能够使用同一个事件监听器类就可以实现代码重用，同时也节省了代码工作量，提高程序开发速度。

这样的开发思路存在很多优点，但是实现起来并不容易，子窗体的名称、类名都可以获取，但是如何根据指定的类名去创建子窗体对象呢？

Java 的反射功能为这个思路提供了可行性。在 `java.lang.reflect` 包中有 `Field` 类、`Method` 类和 `Constructor` 类，这 3 个类分别描述类的字段、方法和构造方法。这里需要的就是类的构造方法，只有调用类的构造方法才能创建该类的实例对象。可以通过 `Class` 类的 `getConstructor()` 方法获取 `Constructor` 类的实例对象，然后调用该对象的 `newInstance()` 方法创建类的实例对象。关键代码如下：

例程32 代码位置：光盘\TM\01\JXCManager\src\com\lzw\JXCFrame.java

```
try {
    Class fClass = Class.forName("internalFrame." + frameName);
    Constructor constructor = fClass.getConstructor(null);
    jf = (JInternalFrame) constructor.newInstance(null);
    ifs.put(frameName, jf);
} catch (Exception e) {
    e.printStackTrace();
}
```

■ 代码贴士

- 调用 `Class` 类的 `forName()` 方法加载指定的 Java 类，该方法将返回该类的 `Class` 实例对象。
- 调用指定类的 `getConstructor()` 方法获取指定的构造器。方法中使用 `null` 作参数，是调用该类的默认构造器，因为类的默认构造器没有任何参数。
- 调用构造器的 `newInstance()` 方法，同样传递参数 `null`，这样就可以调用默认的构造方法创建子窗体对象。

1.13 使用 PowerDesigner 逆向生成数据库 E-R 图

在开发一个新的程序时，为提高开发速度，经常修改现有的与将要开发的程序相类似的旧程序。同样，功能相似的程序，它们的数据库也基本相似，甚至完全相同，那么可以直接使用原有的数据库从而节省数据库设计的时间和工作量。而要分析一个数据库的数据结构和连接关系，E-R 图是最好的数据库资料。但是原有的数据库也许是多年以前的，或者是借鉴同事的，资料不一定完整，也不一定存在 E-R 图。这就给数据库分析带来了很大的不便。

如果能够使用相应的设计工具将数据库的结构和关系：抽象成 E-R 图，就可以为系统分析员提供相应的数据库资料，从而分析或修改原有数据库。本节将介绍如何使用 `PowerDesigner` 工具实现数据库 E-R 图的逆向生成。

(1) 在开始逆向生成 E-R 图之前, 需要为指定的数据库创建 ODBC 数据源。以 Windows 2003 操作系统为例, 选择“开始”/“运行”命令, 在“运行”对话框中输入 `odbcad32.exe`, 单击“确定”按钮, 启动数据源管理器。

(2) 在“ODBC 数据源管理器”对话框中单击“添加”按钮。

(3) 在弹出的“创建新数据源”对话框中选择“SQL Server”选项, 单击“完成”按钮, 如图 1.33 所示。

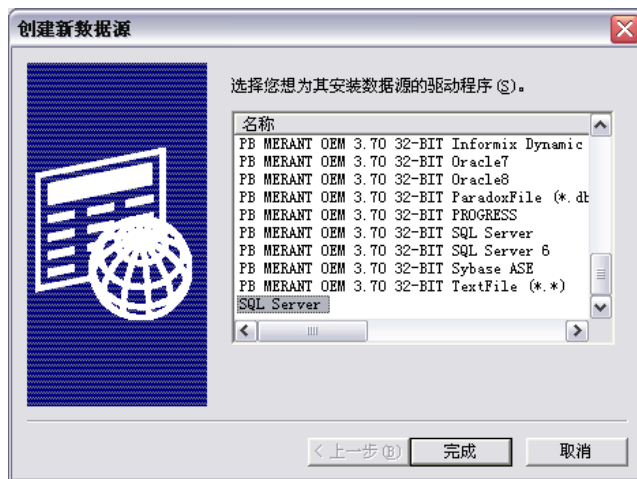


图 1.33 “创建新数据源”对话框

(4) 在弹出的“创建到 SQL Server 的新数据源”对话框的“名称”文本框中输入新建数据源的名称, 例如 `db_jxcOdbc`。在“描述”文本框中可以输入该数据源的描述信息, 因为数据源的名称经常使用单词的缩写形式, 随着时间的流逝很容易忘记其含义, 如果搭配相应的描述信息, 会使该数据源的含义更明确。在“服务器”下拉列表框中输入 `localhost`, 单击“下一步”按钮, 如图 1.34 所示。

(5) 在弹出的对话框中, 选中“使用用户输入登录 ID 和密码的 SQL Server 验证”单选按钮, 然后选中“连接 SQL Server 以获得其他配置选项的默认设置”复选框, 在“登录 ID”文本框中输入访问数据库的用户名, 例如 `sa`, 在“密码”文本框中输入访问数据库的密码, 单击“下一步”按钮, 如图 1.35 所示。

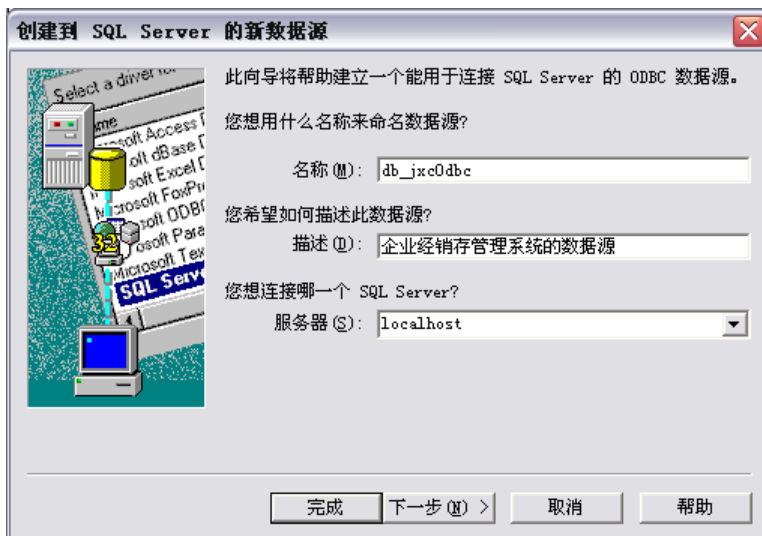



图 1.34 “创建到 SQL Server 的新数据源”对话框（1）



图 1.35 “创建到 SQL Server 的新数据源”对话框（2）

(6) 在弹出的对话框中的“更改默认的数据库”下拉列表框中，选择操作的数据库，例如本系统的 db_JXC，单击“下一步”按钮，然后在弹出的对话框中单击“完成”按钮创建数据源。

(7) 建立完数据源以后，启动 PowerDesigner，重复 1.4.3 节中的步骤（1）建立一个空的物理数据模型。

(8) 选择 Database/Reverse Engineer Database 命令，在弹出的对话框中选中 Using a data source 单选按钮，单击右侧的  按钮选择刚刚建立的 db_jxcOdbc 数据源，单击“确定”按钮，如图 1.36 所示。

(9) 在弹出的对话框中选择需要生成 E-R 图的数据表、视图、系统表等，单击“确定”按钮生成数据库的 E-R 图。本系统的数据库 E-R 图在 1.4.3 节中已经介绍过，效果如图 1.16 所示。

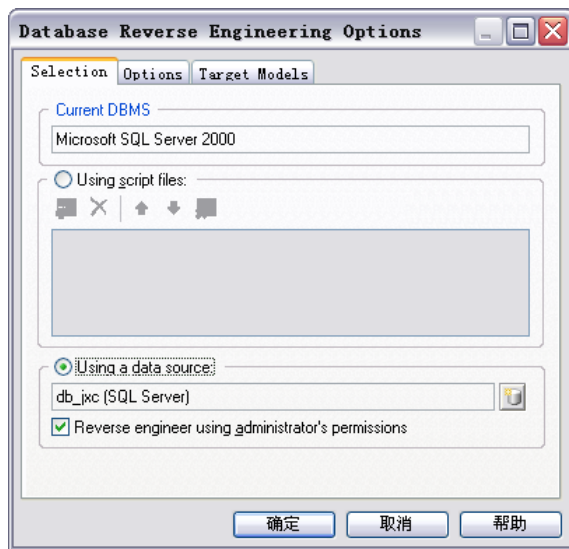


图 1.36 Database Reverse Engineering Options 对话框

1.14 本章小结

本章运用软件工程的设计思想，通过一个完整的企业进销存管理系统为读者详细讲解了一个系统的开发流程。通过本章的学习，读者可以了解 Java 应用程序的开发流程及 Java Swing 的窗体设计、事件监听等技术。另外，本章还介绍了 PowerDesigner 工具的数据库建模和逆向生成数据库 E-R 图以及 Java 应用程序的系统打包等技术，希望对读者日后的程序开发有所帮助。