进销存教案

版本号：JAVASE1.0

密 级：受控文档

课

程

标

准

化

2016年12月X日

1.文档属性

|  |  |
| --- | --- |
| 文档属性 | 内容 |
| 项目/任务名称： |  |
| 项目/任务编号： |  |
| 文档名称： |  |
| 文档编号： |  |
| 文档状态： |  |
| 作 者： | 实训部开发组 |
| 文档评审通过日期： |  |
| 评审负责人签字： |  |
| 参考模板： |  |

2.文档变更过程

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修正日期 | 修正人 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目 录

[第1章 Swing界面 4](#_Toc469385192)

[1.1 章节目标 4](#_Toc469385193)

[1.2 NetBeans的使用 4](#_Toc469385194)

[1.3 swing常用组件的应用 5](#_Toc469385195)

[1.4 反射动态生成子界面 13](#_Toc469385196)

[第2章 软件开发流程 14](#_Toc469385197)

[2.1 章节目标 14](#_Toc469385198)

[2.2 开发流程简介 14](#_Toc469385199)

[第3章 系统需求分析 14](#_Toc469385200)

[3.1 章节目标 14](#_Toc469385201)

[3.2 需求文档的编写 15](#_Toc469385202)

[3.3 用例图及用例描述 17](#_Toc469385203)

[3.4 进销存系统模块分析 20](#_Toc469385204)

[第4章 系统概要设计 25](#_Toc469385205)

[4.1 章节目标 25](#_Toc469385206)

[4.2 概要设计文档的编写 25](#_Toc469385207)

[4.3 流程图，E-R图设计 26](#_Toc469385208)

[4.4 powerdesigner的使用 28](#_Toc469385209)

[4.5 进销存系统数据库设计 31](#_Toc469385210)

[4.6 进销存系统概要设计 36](#_Toc469385211)

[第5章 系统详细设计 36](#_Toc469385212)

[5.1 章节目标 36](#_Toc469385213)

[5.2 详细设计文档的编写 36](#_Toc469385214)

[5.3 时序图设计 39](#_Toc469385215)

[5.4 通过UML类图的方式进行进销存系统接口设计 42](#_Toc469385216)

[5.5 进销存系统详细设计 45](#_Toc469385217)

[第6章 系统底层框架搭建 45](#_Toc469385218)

[6.1 章节目标 45](#_Toc469385219)

[6.2 三层架构说明 45](#_Toc469385220)

[6.3 进销存系统文件组织结构设计 45](#_Toc469385221)

[6.4 进销存公共类和接口的设计及编写 46](#_Toc469385222)

[第7章 系统重难点分析 55](#_Toc469385223)

[7.1 章节目标 55](#_Toc469385224)

[7.2 采购业务的实现 55](#_Toc469385225)

[7.3 导出excel及JfreeChart生成图表 57](#_Toc469385226)

# Swing界面

## 章节目标

了解NetBeans的使用

√ 掌握swing常用组件的应用

√ 掌握多文档界面的实现

√ 掌握利用反射动态生成子界面

## NetBeans的使用

1. 主要内容：

了解NetBeans的基本操作

1. 知识点：

(1)、NetBeans界面布局

(2)、NetBeans组件事件的添加

(3)、NetBeans使用其他注意事项

3、解析：

(1)、NetBeans界面布局：

简单介绍NetBeans工具的作用：由sun公司开发，支持各种应用，包括桌面应用、web应用、移动应用，甚至支持PHP、Ruby、C/C++等开发语言，是一款包括开源的开发环境和应用平台的IDE，而swing组件的应用可能需要很多的代码来填充实现界面布局，这样比较浪费时间，有悖于java项目桌面应用实训的意义，所以使用NetBeans能够实现拖拽式布局的方式，减少界面布局的精力浪费，让学员的实训更多的集中在业务的实现上面，同时又能更好的展现实训效果。

接下来进行NetBeans基本的布局操作示例：

1. 创建java项目swingdemo1
2. 创建视图包com.hpe.pss.view
3. 创建JFrame窗体：LoginFrame

放置组件--修改文本--修改组件变量名--->登录窗口

4.创建JFrame窗体(主界面)：MDIFrame

随意放几个组件即可

5.给组件添加事件

登录按钮实现窗体切换

4、代码实现：

|  |
| --- |
| private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  String username = this.txtUsername.getText();  String password = new String(this.txtPassword.getPassword());  if(username.equals("wangbaoqiang")&&password.equals("marong")){  MDIFrame mainFrame = new MDIFrame();  mainFrame.setVisible(true);  this.dispose();  }else{  JOptionPane.showMessageDialog(this, "用户名或密码错误");  }  } |

1. 、展示组件事件的基本添加及删除方式，可视化操作
2. 、讲解一些NetBeans的使用注意事项，比方说未编译则可以右键编译，窗体设计页包含预览窗体按钮等。

## swing常用组件的应用

1. 主要内容：

Swing常用组件的应用：JInternalFrame、JLabel、JButton、JTable、JTextField、JMenuBar、JRadioButton、JComboBox

1. 知识点：

多文档背景显示-自定义组件及动态子界面显示

JLabel:无需更改变量名，只做显示文字使用。

JButton:如上节登录窗口的按钮

JTable:动态添加数据

JTextField:修改文本显示内容，获取内容

JMenuBar:添加事件，添加子菜单

JRadioButton:添加按钮组实现单选

JComboBox:实现动态添加数据

1. 解析：

本节主要讲解一下常用的swing组件的一些用法，重难点是在JInternalFrame、JTable和JComboBox动态显示数据那里。

**多文档背景显示-自定义组件及多文档动态生成子界面**可以先研究如何自定义组件，设置主窗体背景，然后再演示如何生成多文档子界面，可根据指导文档进行讲述。

JLabel通过可视化操作更改一下显示内容即可

JButton在此节可略过

JTextField讲解设置内容和获取内容的方式，其中获取密码框需要字符数组接收。

JMenuBar可视化操作添加子菜单等操作即可

JRadioButton可以通过添加按钮组添加单选或多选按钮的可视化操作实现单选或多选

**JTable**可以先通过可视化的操作，添加部分数据，然后在后台创建集合的方式，通过代码先删除数据，然后再添加集合当中的数据到table里面，其中用到了DefaultTableModel和Vector集合，调用DefaultTableModel的getRowCount()和removeRow()方法进行删除数据，再利用addRow(Vector)方法添加数据。

**JComboBox**可以先通过可视化的操作，添加部分数据，然后在后台创建集合的方式，将对象集合添加到组件当中，然后更改对象的toString()方法用于显示对象的值，如名称等。

代码实现案例：

多文档背景设计-自定义组件



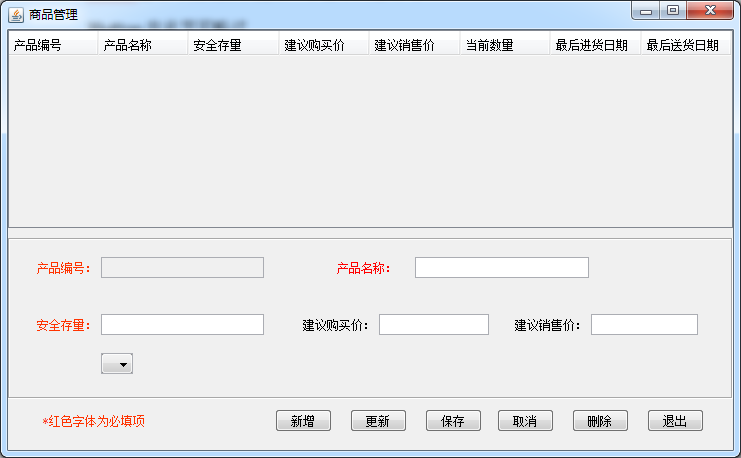
动态生成子界面



JTextField(登录界面)：

|  |
| --- |
| private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  String username = this.txtUsername.getText();  String password = new String(this.txtPassword.getPassword());  if(username.equals("wangbaoqiang")&&password.equals("marong")){  MDIFrame mainFrame = new MDIFrame();  mainFrame.setVisible(true);  this.dispose();  }else{  JOptionPane.showMessageDialog(this, "用户名或密码错误");  }  } |

JTable（产品管理）：



1. 创建dao、util、service、po包
2. 创建BaseDao、ProductDao、ProductService接口、ProductServiceImpl
3. BaseDao部分代码：

|  |
| --- |
| public List query(String sqlString, Class clazz) {  List beans = null;  Connection conn = null;  try {  conn = getConnection();  QueryRunner qRunner = new QueryRunner();  beans =  (List) qRunner.query(  conn,  sqlString,  new BeanListHandler(clazz));  } catch (SQLException e) {  e.printStackTrace();  } finally {  DbUtils.closeQuietly(conn);  }  return beans;  } |

ProductDao部分代码继承BaseDao即可：

|  |
| --- |
| public class ProductDao extends BaseDao {    } |

ProductService代码如下：

|  |
| --- |
| public interface ProductService {  public boolean add(Product p);  public boolean deleteById(int pid);  public boolean update(Product p);  public Product getProductById(int id);  public List<Product> getAll();  } |

ProductService部分代码如下：

|  |
| --- |
| public List<Product> getAll() {  String sql="select \* from product";  List<Product> productList = dao.query(sql, Product.class);  return productList;  } |

ProductFrame显示数据到JTable中代码实现：

|  |
| --- |
| private void initData() {  List<Product> list = productService.getAll();    DefaultTableModel dtm = (DefaultTableModel) this.tblProduct.getModel();  for(Product p:list){  this.cbProduct.addItem(p);  Vector v = new Vector();  v.add(p.getPid());  v.add(p.getPname());  v.add(p.getSafeamount());  v.add(p.getSuggestpurchaseprice());  v.add(p.getSuggestsaleprice());  v.add(p.getCurrentamount());  v.add(p.getLastpurchasetime());  v.add(p.getLastsaletime());  dtm.addRow(v);  }  } |

Product.java代码

|  |
| --- |
| public class Product {  private int pid;  private String pname;  private String ptype;  private int safeamount;  private double suggestpurchaseprice;  private double suggestsaleprice;  private int currentamount;  private Date lastpurchasetime;  private Date lastsaletime;  private int state;  public int getPid() {  return pid;  }  public void setPid(int pid) {  this.pid = pid;  }  public String getPname() {  return pname;  }  public void setPname(String pname) {  this.pname = pname;  }  public String getPtype() {  return ptype;  }  public void setPtype(String ptype) {  this.ptype = ptype;  }  public int getSafeamount() {  return safeamount;  }  public void setSafeamount(int safeamount) {  this.safeamount = safeamount;  }  public double getSuggestpurchaseprice() {  return suggestpurchaseprice;  }  public void setSuggestpurchaseprice(double suggestpurchaseprice) {  this.suggestpurchaseprice = suggestpurchaseprice;  }  public double getSuggestsaleprice() {  return suggestsaleprice;  }  public void setSuggestsaleprice(double suggestsaleprice) {  this.suggestsaleprice = suggestsaleprice;  }  public int getCurrentamount() {  return currentamount;  }  public void setCurrentamount(int currentamount) {  this.currentamount = currentamount;  }  public Date getLastpurchasetime() {  return lastpurchasetime;  }  public void setLastpurchasetime(Date lastpurchasetime) {  this.lastpurchasetime = lastpurchasetime;  }  public Date getLastsaletime() {  return lastsaletime;  }  public void setLastsaletime(Date lastsaletime) {  this.lastsaletime = lastsaletime;  }  public int getState() {  return state;  }  public void setState(int state) {  this.state = state;  }  @Override  **public String toString() {**  **return pname;**  **}**    } |

JComboBox数据动态显示代码实现：

|  |
| --- |
| for(Product p:list){  this.cbProduct.addItem(p);  } |

## 反射动态生成子界面

1. 主要内容：

利用反射动态生成子界面

1. 解析

运用反射的概念，替换掉上一节所讲的通过字符串创建相应的子界面，可以通过指导文档讲解，最终分析出它的灵活性即可

1. 代码实现：

指导文档



# 软件开发流程

## 章节目标

√ 掌握软件开发基本流程

## 开发流程简介

1. 主要内容

简单讲解软件开发的基本流程

1. 解析

本节可以简单讲解一下一个软件开发的整个过程大概是什么样子，并将其运用在进销存系统当中。

1. 首先要有需求，这也是软件开发的开端，往往需求都是文字性的描述。
2. 紧接着就需要根据用户需求做出原型，在这里可以提一下几种原型工具GUIDesigner（客户端原型设计工具）、Axure RP（web原型设计工具）、NetBeans（客户端和web皆可）等。同时需要和用户不断的沟通迭代原型，最后形成用户所希望东西，确认最终需求。
3. 确认需求之后就可以启动项目了，启动项目时需要制定项目任务书、组建项目团队、制定项目计划，这时候的项目计划可以粗略一些
4. 接下来就是开发阶段，开发阶段分为需求规格说明（需求分析，详细的开发计划）、概要设计（画出流程图和设计数据库）、详细设计（画出时序图、设计出项目接口）、编码、测试、修复、测试、
5. 实施阶段进行项目交付及后期产品维护。

讲解清楚以上的流程之后就可以说明我们的进销存系统将按照以上流程完成。

# 系统需求分析

## 章节目标

√ 掌握需求文档如何编写

√ 掌握UML用例图及事件流图如何绘制

√ 进销存系统的需求分析及文档编写

## 需求文档的编写

1. 主要内容：

本节主要讲解需求文档包含的内容及其各模块编写方式。

1. 解析：
2. 首先讲解需求文档包含的哪些模块哪些内容：

产品介绍、产品面向的用户群体、产品应当遵循的标准或规范、产品的范围、产品中的角色、产品的功能性需求（分为几个模块，各个模块有哪些功能等）、产品的非功能性需求（界面需求、软硬件环境需求，质量需求等）。

1. 将进销存系统的产品介绍、产品面向的用户群体、产品应当遵循的标准或规范、产品的范围、产品中的角色融入到进销存系统需求分析文档当中。
2. 产品的功能性需求里面可以以一个商品管理模块作为例子，编写商品管理的功能性需求描述。
3. 产品非功能性需求由导师结合进销存系统进行引导式讲解，学生自行编写即可
4. 具体实现：
5. 产品介绍：

|  |
| --- |
| 进销存（PSS）软件是一款通用性极强的商业企业进销存管理系统，软件囊括了商业企业日常经营管理的全过程。软件适用于食品、服装、保健品、电子、电器、物资、化妆品等商业领域的企业。软件覆盖商业企业商品的采购、销售、库存、管理决策支持等各个环节，软件的各个模块操作界面简单、实用，软件帮助系统图文并茂，让用户可以在最短的时间内掌握软件的使用方法，助用户生意早日更上一层楼。软件的采购管理、销售管理、库存管理功能强大，灵活，同时为用户提供方便实用的模糊查询功能，用户可以在最短时间内找到自己所需要的信息。软件系统采用流行的B/S结构，可以供多台电脑协同工作，数据共享。软件采用模块级安全管理，系统安全性高。软件对在使用过程中可能出现的外部意外情况，如断电、网络中断等均进行了妥善处理，可以确保用户数据的完整性，同时在外部故障消除后，系统仍能继续操作。系统软件采用Java平台进行开发并实现，用户数据选用mysql进行统一管理。 |

1. 用户群体

|  |
| --- |
| 进销存（PSS）主要是帮助商业企业全面有效管理采购、销售和库存，软件适用于需要进行采购管理，销售管理以及库存管理的所有商业企业。 |

1. 产品应当遵循的标准或规范

|  |
| --- |
| 本产品的是一套标准的进销存软件，符合国家财务标准，本产品遵循《计算机软件保护条例》的各项规定。 |

1. 产品范围

|  |
| --- |
| PSS进销存管理系统是集采购、销售及库存管理于一体的商业企业管理软件系统。本产品适用于商业贸易、批发行业以及个人用户，不适用于零售行业及行业标准较强的行业（如医疗药品的进销存管理）。 |

1. 产品中的角色

|  |  |
| --- | --- |
| **角色名称** | **职责描述** |
| 超级管理员 | 拥有使用系统所有功能的权限。 |
| 普通用户 | 在拥有相应的权限的情况下，可以维护各种资料，管理采购，销售及库存。 |

1. 功能性需求

例：商品管理模块

| 功能类别 | 功能名称、标识符 | 描述 |
| --- | --- | --- |
| **用户登录** | 用户登录 | 对用户输入的用户名，密码进行验证，验证通过后，该用户可以使用PSS系统中自己拥有权限的那部分功能，否则拒绝使用。 |
|  | 商品资料维护 | 用户修改，删除，新增或查询商品数据，系统根据用户的操作，对商品资料进行更新或显示。 |

1. 产品非功能性需求

此处内容可略

模板如下：

用户界面需求

|  |  |
| --- | --- |
| **需求名称** | **详细要求** |
|  |  |
|  |  |
|  |  |
|  |  |

软硬件环境需求

|  |  |
| --- | --- |
| 需求名称 | 详细要求 |
|  |  |
|  |  |
|  |  |

产品质量需求

|  |  |
| --- | --- |
| 主要质量属性 | 详细要求 |
| 正确性 |  |
| 健壮性 |  |
| 可靠性 |  |
| 性能，效率 |  |
| 易用性 |  |
| 清晰性 |  |
| 安全性 |  |
| 可扩展性 |  |
| 兼容性 |  |
| 可移植性 |  |

## 用例图及用例描述

1、主要内容：

本节主要讲述UML建模工具的使用及其中用例图的画法，用例图的作用，参与者、用例、关系线等元素分别代表的含义及如何去组合。建议使用工具starUML

2、解析

一、首先讲解需求分析文档当中需要有用例图对模块功能进行说明，然后解释UML的作用及UML里面用例图的概念及其作用。

Unified Modeling Language (UML)又称统一建模语言或[标准建模语言](http://baike.baidu.com/view/3458435.htm)，是始于1997年一个[OMG](http://baike.baidu.com/subview/138917/5088704.htm)标准，它是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持，包括由需求分析到规格，到构造和配置。

目前学习UML的目的是能够看懂UML图，并根据图进行设计与研发。

可以介绍一下UML的体系结构，如下图：



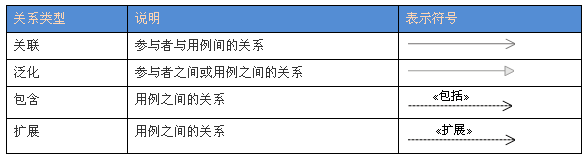
然后说明本学期只学习UML的用例图、类图、时序图、流程图。首先根据项目进行阶段先学习用例图。

用例图是指由参与者（Actor）、[用例](http://baike.baidu.com/view/706238.htm)（Use Case）以及它们之间的关系构成的用于描述系统功能的视图。用例图（User Case）是被称为参与者的外部用户所能观察到的系统功能的模型图，呈现了一些参与者和一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行[建模](http://baike.baidu.com/view/44500.htm)。

[用例](http://baike.baidu.com/view/706238.htm)图主要的作用有三个：（1）获取需求；（2）指导测试；（3）还可在整个过程中的其它[工作流](http://baike.baidu.com/view/60285.htm)起到指导作用。

二、用例图示例，这时候可以用用户登录作为示例，画一个基本的用例图，解释参与者、用例、关系线这些元素在用例图里面是如何体现的。

1. 用例关系解释，解释一下用例case之间各关系如何用关系线连接，如



3、具体实现案例：

用户登录用例图：

相关关系解释如下：

一个case表示一个功能模块，如登录和注册则是两个case

将基用例中一段相对独立并且可选的动作，用extend（扩展）线，如修改密码是登录的一个可选的动作。

包含用例，用include（包含）线，如校验用户名是用户注册的一个必然的动作。

子用例和父用例相似，但表现出更特别的行为；子用例将继承父用例的所有结构、行为和关系，用generalization（泛化）线。这里可以说明具体的泛化关系将在后面的类图的讲解当中有更详细的描述。

用例描述：参与者为实现目标与系统进行交互的过程。

主要包含以下几个方面：用例名称、参与者、基本事件流、前置条件、后置条件、备选事件流。

如用户登录用例描述：

|  |
| --- |
| 1.0 用例名称：  中文名称：用户登录  功能：验证用户的身份。  1.1 简要说明：  本用例的功能主要是用于确保用户在提供正确的验证信息之后，可以进一步使用本系统。  1.2 事件流：  1.2.1 基本流：  1 用户请求使用本系统。  2 系统显示用户登录信息输入界面。  3 用户输入登录名，密码并确认操作。  4 系统验证用户登录信息，如果登录信息验证没有通过，系统显示提醒信息，并转向基本流2，如果验证通过，系统显示系统操作主界面。  1.2.2 备选流：  1.2.2.1 备选流1：  1 客户可以在没有登录成功之前的任意时候要求放弃登录。  2 系统结束用户登录信息输入界面的显示。  3 退出系统。  1.3 特殊需求：  无  1.4 前置条件：  1 请求使用本系统。  1.5 后置条件：  1 用户登录成功,可以使用系统提供的功能。  1.6 附加说明：  无 |

## 进销存系统模块分析

1. 主要内容：

分析进销存系统商品管理示例模块需求分析和用例图及其描述

1. 解析：

本节主要是首先进行进销存系统的模块分析及划分，划分完成之后以商品管理为示例，进行系统模块的分析，并将分析的内容及用例图和事件流加入到系统需求分析文档当中。最后由学生去填充其他模块。

1. 具体实现：

进销存系统模块划分：

| **功能类别** | **功能名称、标识符** | **描述** |
| --- | --- | --- |
| **用户登录** | 用户登录 | 对用户输入的用户名，密码进行验证，验证通过后，该用户可以使用PSS系统中自己拥有权限的那部分功能，否则拒绝使用。 |
| **维护基本资料** | 销售员资料维护 | 用户修改，删除，新增或查询销售员数据，系统根据用户的操作，对销售员资料进行更新或显示。 |
| 商品资料维护 | 用户修改，删除，新增或查询商品数据，系统根据用户的操作，对商品资料进行更新或显示。 |
| 客户资料维护 | 用户修改，删除，新增或查询客户数据（其中包括对客户的联系人的修改，删除，新增与查询以及对客户交易记录的查询），系统根据用户的操作，对商品资料进行更新或显示。 |
| 供应商资料维护 | 用户修改，删除，新增或查询供应商数据（其中包括对供应商的联系人的修改，删除，新增与查询以及对供应商交易记录的查询），系统根据用户的操作，对供应商资料进行更新或显示。 |
| **采购** | 采购入库 | 用户通过录入采购入库单增加采购的货物，并可对采购入库单及其单据中的货物明细进行修改、删除与查询。 |
| 采购退货 | 用户录入通过采购退货单退回货物，并可对采购退货单及其单据中的货物明细进行修改、删除与查询。 |
| **销售** | 销售出库 | 用户通过录入销售出库单记录销售的货物，并可对销售出库单及其单据中的货物明细进行修改、删除与查询。 |
| 销售退货 | 用户通过录入销售退货单退回已销售货物，并可对销售退货单及其单据中的货物明细进行修改、删除与查询。 |
| **库存** | 库存查询 | 用户通过组合不同条件，对库存进行查询、盘点。 |
| **维护系统** | 维护用户资料 | 拥有系统维护权限的用户可以增加新用户，并可以对用户的资料进行修改、删除以及查询。 |
| 管理用户权限 | 拥有权限管理的用户可以设定其他用户对软件的访问权限。 |
| 修改密码 | 用户重新设定自己的密码 |
| **报表导出** | 导出库存预警 | 库存量低于安全存量的商品系统向用户提供警示信息，以提醒用户及时进货。用户也可将警示信息导出到Excel中。 |
| 导出商品资料 | 将商品资料信息导出到Excel中。 |
| 导出采购报表 | 按日期或用户定义的汇总方式对采购信息进行汇总，并将汇总结果导出到Excel中。 |
| 导出年采购报表 | 根据年度汇总采购支出金额，并按月分将其导出在Excel中。 |
| 导出销售报表 | 按日期或用户定义的汇总方式对销售信息进行汇总，并将汇总结果导出到Excel中。 |
| 导出年销售报表 | 根据年度汇总销售收入金额，并按月分将其导出在Excel中。 |

系统用例图示例：

|  |
| --- |
|  |

|  |
| --- |
|  |

商品管理用例描述示例：

|  |
| --- |
| 用例名称：  中文名称：维护商品资料  功能：用于维护公司经营的商品的信息资料。  1 简要说明：  本用例的功能主要是增加、删除、修改、查询公司所经营商品的信息资料。  2 事件流：  2.1 基本流：  1 用户请求维护公司商品资料。  2 系统显示公司商品信息资料。  3 根据用户的操作执行以下相应操作。   * + 用户修改已经存在的商品信息，系统执行修改商品信息子流。   + 用户选择增加商品操作，系统执行增加商品信息子流。   + 用户选择删除商品操作，系统执行删除商品子流。   + 用户选择查询符合指定条件的商品的信息，系统执行查询商品子流。   4 用户要求保存操作结果。  5 系统保存用户操作结果。  6 用户要求结束公司商品信息的维护。  7 系统结束公司商品资料的显示。  2.2 备选流：  2.2.1 备选流1：  1 如果在用户请求保存操作结果的时候，由于网络、数据库管理系统等外部原因造成操作结果不能保存，系统保证以恰当的方式通知用户，并维护用户的操作状态，在外部原因消除之后，用户仍能继续操作。  2.2.2 备选流2：  1 如果用户要求结束公司商品信息维护的时候，仍有未保存的信息，系统提醒用户。  2 根据用户的选择执行以下操作。  用户选择保存信息，系统保存信息，继续执行基本流7。  用户选择不保存信息，系统执行基本流7。  用户选择取消操作，系统执行基本流2。  3 特殊需求：  1 商品编号必须指定，输入不能超过8位字符。  2 商品名称必须指定，输入不能超过30个英文字符或15位中文字符。  3 安全存量必须指定，只能输入数值和小数点。  4 建议采购价和建议销售价，必须指定，只能输入数值和小数点。  5 商品的最后一次进货日期和最近一次销售日期，在添加商品时，不允许填写。  6商品的最后一次进货日期和最近一次销售日期，在修改商品信息时，不允许修改。  4 前置条件：  1 进入本系统的主界面。  2 拥有维护商品信息资料的权限。  5 后置条件:  1 系统保存修改过的商品资料。  6 附加说明:  1 操作的商品资料应包括：商品编号、商品名称、安全存量、当前数量、建议采购价、建议销售价、最后一次采购日期以及最后一次销售日期。 |

# 系统概要设计

## 章节目标

√ 掌握概要设计文档如何编写

√ 掌握UML流程图如何绘制

√ 掌握powerdesigner如何使用

√ 进销存系统数据库设计

√ 进销存系统概要设计文档编写

## 概要设计文档的编写

1. 主要内容：

本节主要讲解概要设计文档包含的内容及其各模块编写方式。

1. 解析：

首先讲解概要设计文档当中包含的模块及各模块的内容:

1. 引言：

编写目的、指出预期的读者、背景、定义、参考资料等

2.规格说明分析：

对系统需求进行简要说明。

3.系统体系结构：

根据选用的软件、硬件以及网络环境构造系统的整体框架，划分系统模块，并对系统内各模块之间的关系进行定义，确定已定义的对象以及其组件在系统内如何传输、通信。

4.界面设计定义：

各模块界面流，即各个功能所流向的界面。

5.接口定义：

人机交互接口，网络接口（服务器和网络拓扑图），系统与外部接口（导出excel之类的），系统内模块之间的接口，数据库接口。

6.模块设计：

系统功能结构图

7.数据库设计：

各模块流程图，E-R图，逻辑结构设计（powerdesigner使用），表结构设计

8.故障检测和处理机制：

故障检测机制如日志等，处理机制如事务的处理等。

1. 具体实现：

具体的实现内容将放在以下几节内容结合进销存系统实现。

## 流程图，E-R图设计

1. 主要内容：

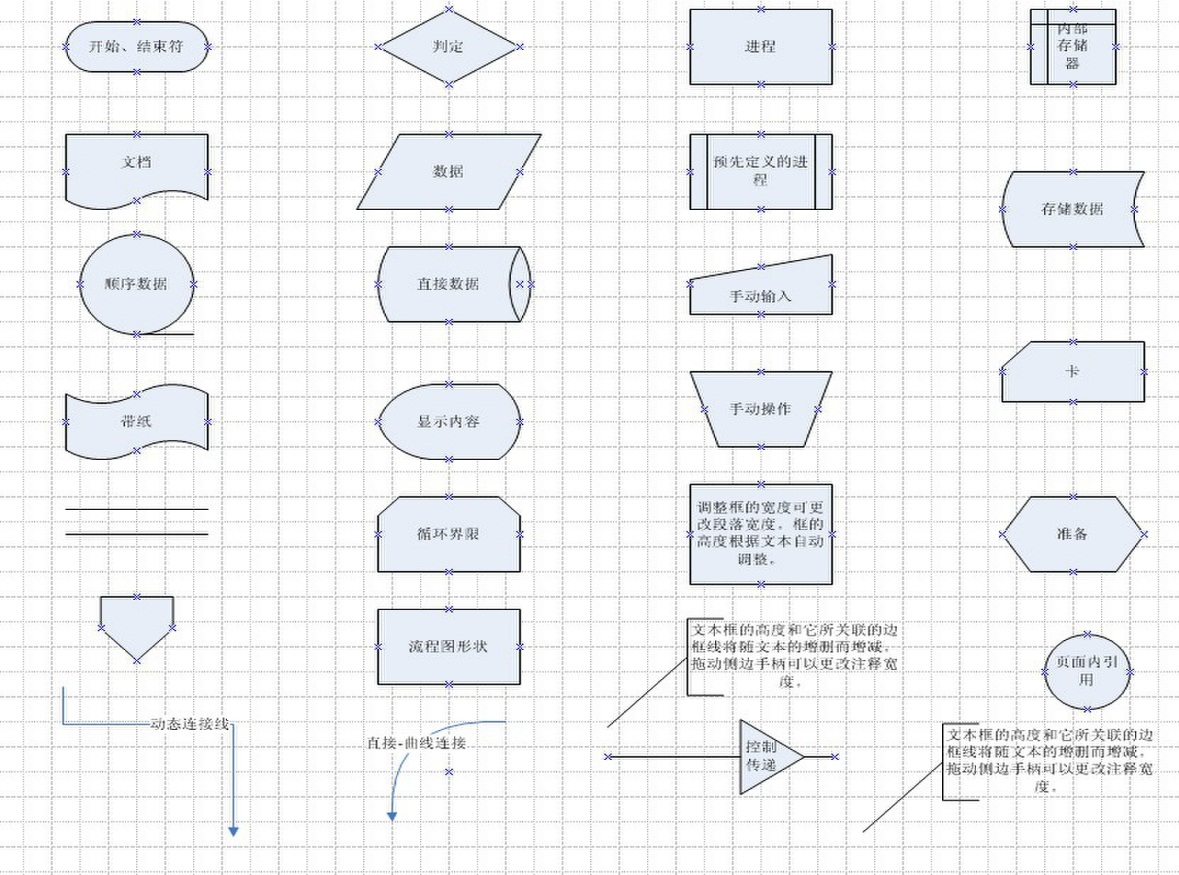
本节主要讲解系统流程图和E-R图的含义及其绘制方式

1. 解析：

首先讲述系统流程图的含义及其作用，然后再以案例的形式，绘制基本的系统流程图

系统流程图是描绘系统物理模型的传统工具。它的基本思想是用图形符号以黑盒子形式描绘系统里面的每个部件(程序、文件、数据库、表格、人工过程等），表达信息在各个部件之间流动的情况。

然后讲解系统流程图当中的基本组成要素有哪些？



以进销存系统为例进行基本的系统流程图的绘制，然后就引导学生进行补充。

然后系统分析完成之后，介绍E-R图的含义及其作用：

E-R图也称实体-联系图(Entity Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的[概念模型](http://baike.baidu.com/view/176360.htm)。

它是描述现实世界概念结构模型的有效方法。是表示概念模型的一种方式，用矩形表示实体型，矩形框内写明实体名；用椭圆表示实体的属性，并用无向边将其与相应的实体型连接起来；用菱形表示实体型之间的联系，在[菱形](http://baike.baidu.com/view/133728.htm)框内写明联系名，并用无向边分别与有关实体型连接起来，同时在无向边旁标上联系的类型（1:1,1:n或m:n）。

然后以进销存系统的用户管理为例进行用户模块的E-R图的绘制，然后引导学生进行后续模块的补充。

1. 具体实现：

进销存系统部分结构（管理员部分）流程图：

|  |
| --- |
|  |

进销存系统部分结构（用户实体）E-R图：

pid

## powerdesigner的使用

1. 主要内容：

本节主要是讲解如何用powerdesigner进行逻辑结构设计。

1. 解析：

首先介绍一下概要设计当中为何要用逻辑结构设计，然后介绍powerdesigner的作用

如：

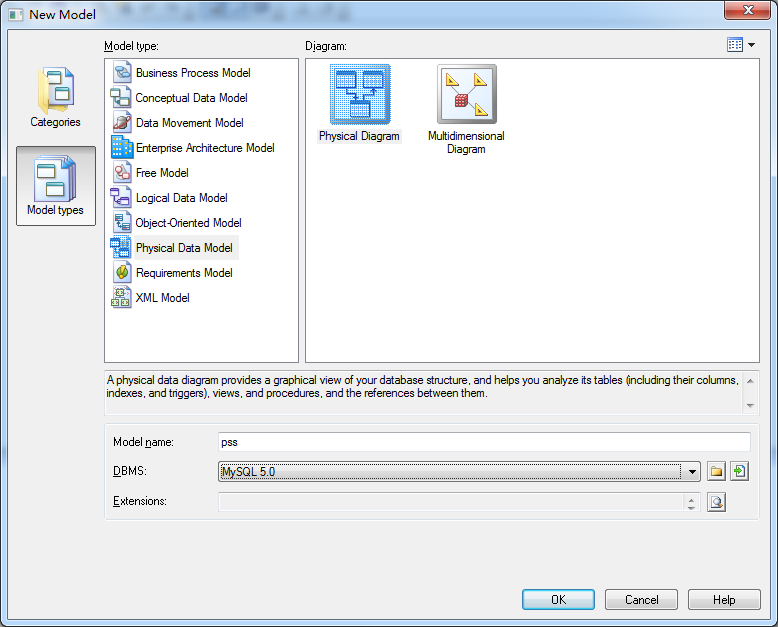
数据库的逻辑结构设计就是把概念结构设计阶段设计好的基本 E 一 R 图转换为与选用的 DBMS产品所支持的数据模型相符合的逻辑结构。

power designer是能进行数据库设计的强大的软件，是一款开发人员常用的[数据库建模](http://baike.baidu.com/view/280978.htm)工具。使用它可以分别从概念数据模型(Conceptual Data Model)和物理数据模型(Physical Data Model)两个层次对数据库进行设计。

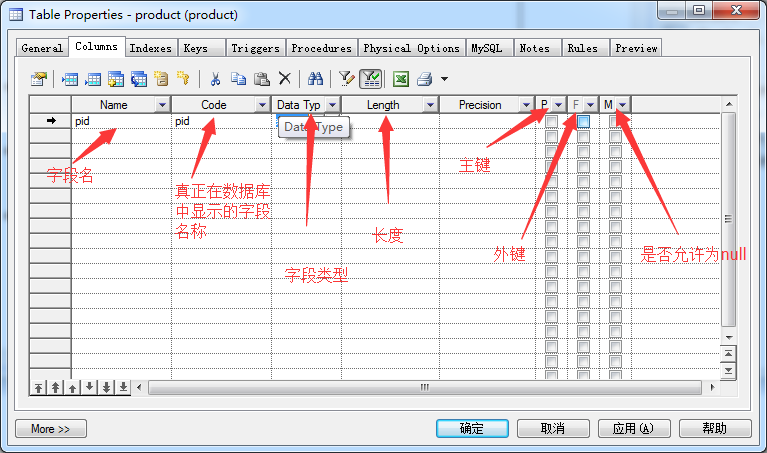
1. 具体实现：

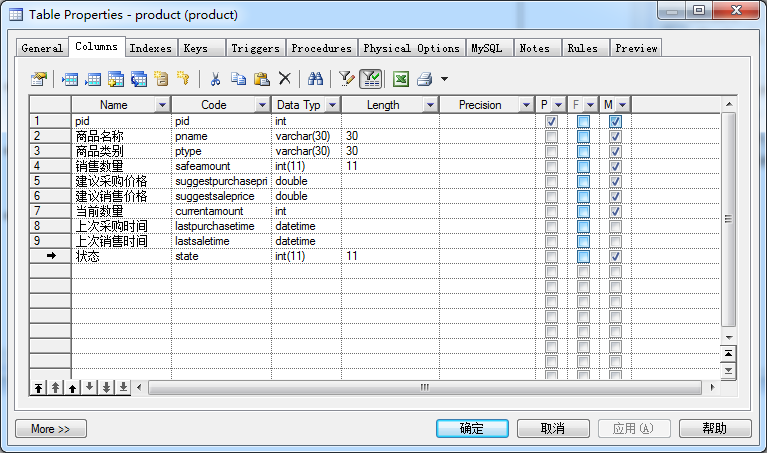
可以以商品和采购为例进行部分逻辑结构的设计，然后让学员去填充其他内容

1. **File—New Model—Physical Data Model—Physical Diagram—Model name** 设置为**pss**,DBMS属性设置为mysql5.0

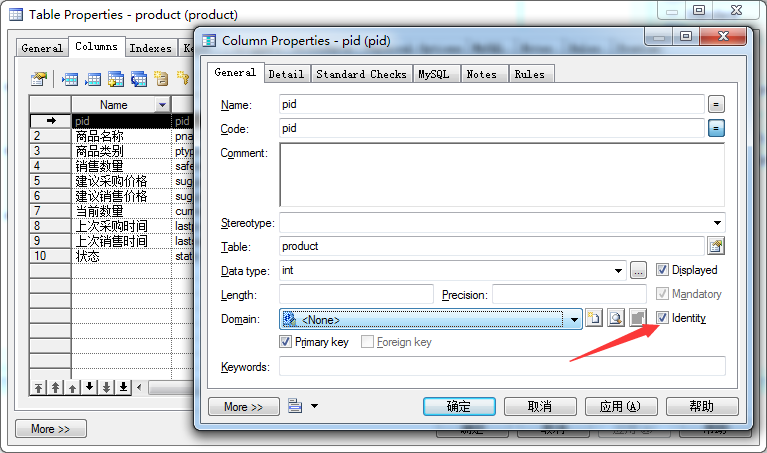


(2)插入一个table，命名为product

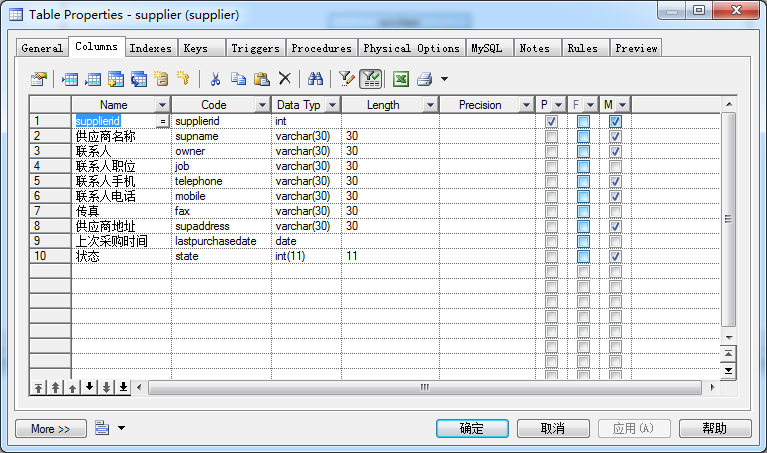




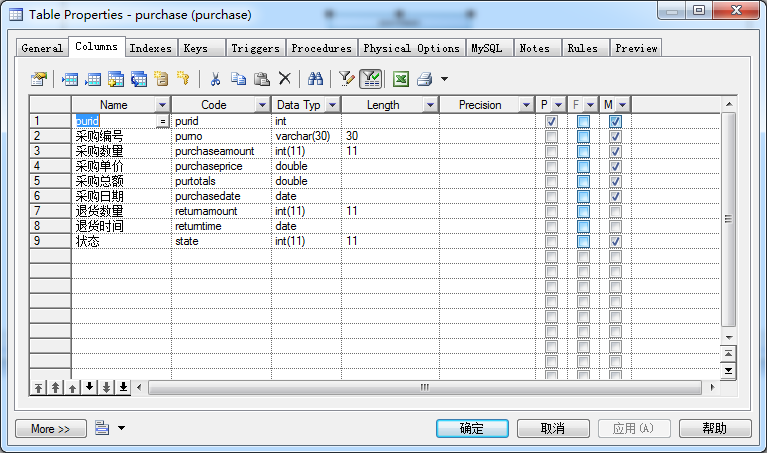
设置字段属性：



插入另一个table命名为supplier（供应商表）

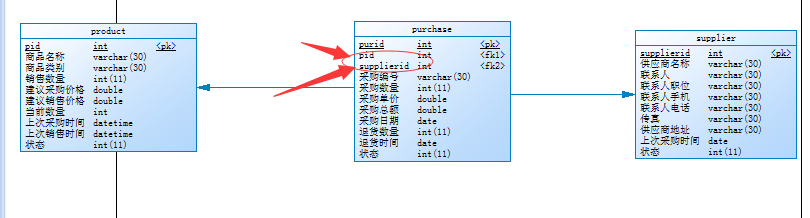


插入采购表（purchase）



确定三者关联关系，则需要添加关系线，外键表指向主键表

如图所示，外键表为自动增加两个外键字段：



## 进销存系统数据库设计

具体设计结构：

0数据表汇总

|  |  |
| --- | --- |
| **表名** | **说明** |
| **Product** | **产品表** |
| **Supplier** | **供应商资料表** |
| **Customer** | **客户信息表** |
| **Employee** | **员工表** |
| **Purchase** | **采购表** |
| **Sales** | **销售表** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **产品表**  Product表字段定义   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 表名 | Product（产品表） | | | | | 列名 | 数据类型（精度范围） | 空/非空 | 约束条件 | 说明 | | Pid | Int | 非空 | PK,自增 | 编号 | | ProductName | varchar(30) | 非空 |  | 产品名称 | | SafeStock | int | 非空 | Defalut:0 | 安全存量 | | Quantity | int | 非空 | Defalut:0 | 当前数量 | | SuggestBuyPrice | decimal(18,2) | 空 |  | 建议购买价 | | SuggestSalePrice | decimal(18,2) | 空 |  | 建议销售价 | | LastPurchaseDate | date | 空 |  | 最后进货日期 | | State | Int | 非空 | Default: 1 | 状态 | | 补充说明 |  | | | |   **供应商资料表**  Supplier表字段定义   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 表名 | Supplier（供应商资料表） | | | | | 列名 | 数据类型（精度范围） | 空/非空 | 约束条件 | 说明 | | SupplierID | Int | 非空 | PK,自增 | 编号 | | SupplierName | varchar(60) | 空 |  | 供应商全称 | | Owner | varchar(30) | 空 |  | 负责人 | | job | varchar(30) | 空 |  | 负责人称谓 | | Telephone | varchar(20) | 空 |  | 电话 | | MobilePhone | varchar(20) | 空 |  | 移动电话 | | Fax | varchar(20) | 空 |  | 传真 | | CompanyAddress | varchar(100) | 空 |  | 公司地址 | | FactoryAddress | varchar(100) | 空 |  | 工厂地址 | | LastPurchaseDate | datetime | 空 |  | 最后进货日期 | | State | Int | 非空 | Default :1 | 状态 | | 补充说明 |  | | | |     **客户信息表**  Customer表字段定义   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 表名 | Customer（客户信息表） | | | | | 列名 | 数据类型（精度范围） | 空/非空 | 约束条件 | 说明 | | CustomerID | Int | 非空 | PK 自增 | 编号 | | CustomerName | varchar(60) | 空 |  | 客户名称 | | TaoBaoID | varchar(30) | 空 |  | 客户淘宝号 | | Telephone | varchar(20) | 空 |  | 电话 | | MobilePhone | varchar(20) | 非空 |  | 移动电话 | | DeliveryAddress | varchar(100) | 空 |  | 客户送货地址 | | LastDeliveryDate | datetime | 空 |  | 最后送货日期 | | State | Int | 非空 | Default:1 | 状态 | | 补充说明 |  | | | |   **员工表**  Employee表字段定义   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 表名 | Employee（员工表） | | | | | 列名 | 数据类型（精度范围） | 空/非空 | 约束条件 | 说明 | | empID | Int | 非空 | PK,自增 | 编号 | | empName | varchar(10) | 非空 |  | 中文名称 | | empSex | varchar(60) | 非空 |  | 性别 | | Origin | Varchar(100) | 空 |  | 籍贯 | | Home | Varchar(100) | 空 |  | 现住址 | | phone | varchar(20) | 非空 |  | 电话 | | QQ | varchar(20) | 空 |  | QQ号 | | Wechat | varchar(50) | 空 |  | 微信号 | | State | Int | 非空 |  | 状态 | | 补充说明 |  | | | |   **采购表**  Pruchase表字段定义   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 表名 | Purchase（采购表） | | | | | 列名 | 数据类型（精度范围） | 空/非空 | 约束条件 | 说明 | | PurchaseID | varchar(8) | 非空 | FK | 采购单号 | | ProductID | varchar(8) | 非空 | FK | 产品编号（外键） | | PurchaseQuantity | Int | 非空 | Defalut:0 | 数量 | | PurchaseUnitPrice | decimal(18,2) | 非空 | Defalut:0 | 单价 | | totalMoney | decimal(18,2) | 非空 | Defalut:0 | 总金额 | | Supplierid | Int | 非空 |  | 供应商编号（外键） | | purchaseDate | Date | 非空 |  | 采购日期 | | returnAmount | Int | 空 |  | 退货数量 | | returnDate | Date | 空 |  | 退货日期 | | returnReason | Varchar(100) | 空 |  | 退货原因 | | State | Varchar | 非空 | Default 1 | 状态 | | 补充说明 |  | | | |   表 2-31 PruchaseDetail表字段定义  **销售表**  sales表字段定义   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 表名 | sales（销售表） | | | | | 列名 | 数据类型（精度范围） | 空/非空 | 约束条件 | 说明 | | salesID | Int | 非空 | FK,自动增长 | 销售单单号 | | ProductID | varchar(8) | 非空 | FK | 产品编号（外键） | | SalesQuantity | int | 非空 | Default:0 | 销售数量 | | SalesUnitPrice | decimal(18,0) | 非空 | Default:0 | 销售单价 | | totalMoney | decimal(18,0) | 非空 | Default:0 | 总金额 | | Customerid | Int | 非空 |  | 客户编号（外键） | | saleDate | Date | 非空 |  | 销售日期 | | returnAmount | Int | 空 |  | 退货数量 | | returnDate | Date | 空 |  | 退货日期 | | returnReason | Varchar(100) | 空 |  | 退货原因 | | 补充说明 |  | | | | |

## 进销存系统概要设计

根据前几节内容，由学员去完成系统的概要设计，此处略。

# 系统详细设计

## 章节目标

√ 掌握详细设计文档如何辨析

√ 掌握UML时序图如何绘制

√ 进销存系统接口设计

√ 进销存系统详细设计文档的编写

## 详细设计文档的编写

1. 主要内容：

本节主要是讲解详细设计文档的作用，及包含哪些模块。

1. 解析：

首先讲解详细设计文档当中包含的模块及各模块的内容:

详细设计的主要任务是设计每个模块的实现算法、所需的局部数据结构。详细设计的目标有两个：实现模块功能的算法要逻辑上正确和算法描述要简明易懂。

包含系统概述，详设概述和各模块流程图、输入输出规则、三层架构类设计说明、功能实现时序图。

1. 具体实现：

如用户登录模块：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实现流程图    输入输出规则  登录模块输入输出规则   |  |  | | --- | --- | | 模块名称 | 登录 | | 功能描述 | 用户登录系统以达到用户的系统权限 | | 用户角色 | 系统用户 | | 模块触发条件 | 直接选择登录 | | 输入规则 | 按照用户名和密码进行登录验证 | | 输出结果 | 登录成功后打开主窗口，用户可以对其权限范围内的信息进行维护。 | | 关键算法 | 1.查询数据库中用户名和密码 | | 数据库操作 | 查询用户表 |   视图层、业务层、数据持久层的说明   |  |  |  | | --- | --- | --- | | **视图** | **业务层（biz）** | **数据持久层（dao）** | | LoginFrame.java  MDIFrame.java | UserService.java  UserServiceImpl.java | UserDao.java  UserDaoImpl.java |   视图层页面设计   |  |  |  | | --- | --- | --- | | **界面** | **JSP** | **功能描述** | | 登录界面 | LoginFrame.java | 登录窗口 | | 主窗口 | MDIFrame.java | 主窗口，包含用户权限下的管理模块 |   业务层设计   |  |  |  | | --- | --- | --- | | **业务接口** | **本模块业务方法** | **方法说明** | | UserService.java | public Users CheckLogin(final String username,final String password); | 接收用户名和密码查询数据库返回User对象 | | UserServiceImpl.java | public Users CheckLogin(final String username,final String password); | 写具体的sql语句，调用数据持久层方法 |   数据持久层设计   |  |  |  | | --- | --- | --- | | **持久层接口** | **基类接口** | **接口说明** | | UserDao.java | BaseDao.java | 基类接口应用泛型和反射机制，对数据的操作进行统一处理，无需各持久层接口设计方法。 | | UserDaoImpl.java | UserDao.java | UserDaoImpl实现UserDao接口 | |

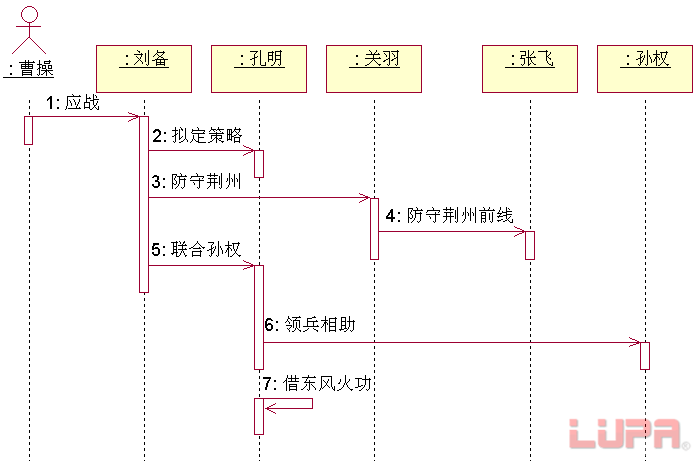
时序图设计将放在下节。

## 时序图设计

首先讲述时序图的概念及其作用

时序图（Sequence Diagram）是显示对象之间交互的图，这些对象是按时间顺序排列的。顺序图中显示的是参与交互的对象及其对象之间消息交互的顺序。时序图中包括的建模元素主要有：对象（Actor）、生命线（Lifeline）、控制焦点（Focus of control）、消息（Message）、自关联消息（Self-Message）等等。

那么时序图是怎么描述出对象之间的关系的呢，接下来就有一个例子说明：



整合成代码便是：

|  |
| --- |
| public class 关羽 {  Public void 防守荊州();  }  public class 张飞  {  public void 防守荆州前线();  }  public class 孙权{  public void 领兵相助();  }  publc class 刘备{  public void 应战();  }  publc class 孔明{  public void 拟定策略();  public void 联合孙权();  private void 借东风火攻();  } |

通过时序图就可以描述出各个对象所包含的行为（方法），及其之间的调用关系。

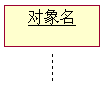
对象的左右顺序并不重要，但是为了画图画的清晰整洁起见，通常应遵循一下两个原则：

（1）把交互频繁的对象尽可能地靠拢。

（2）把初始化整个交互活动的对象（有时是一个参与者）放置在最左边。

**生命线lifeline**

表示对象的生存时间。生命线从对象创建开始到对象销毁时终止。

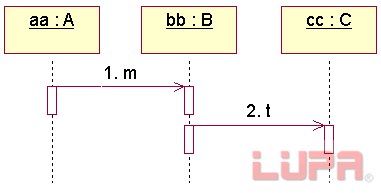
对象在生命线上的两种状态：休眠状态、激活状态。

**消息**

消息的概念：

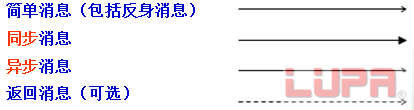
对象之间的交互是通过相互发消息来实现的。一个对象可以请求（要求）另一个对象做某件事件。

消息从源对象指向目标对象。消息一旦发送便将控制从源对象转移到目标对象。



 时序图中，消息的阅读顺序是严格自上而下的。

 消息的类型与符号



实例：同步异步

你叫我去吃饭，我听到了就和你去吃饭；如果没有听到，你就不停的叫，直到我告诉你听到了，才一起去吃饭。

你叫我去吃饭，然后自己去吃饭，我得到消息后可能立即走，也可能等到下班才去吃饭。

所以，要我请你吃饭就用同步的方法，要请我吃饭就用异步的方法，这样你可以省钱。

讲明白对象、生命线、消息之后，就可以以用户登录为案例进行时序图的绘制举例，然后由学生去填充进销存系统其它模块的内容。



## 通过UML类图的方式进行进销存系统接口设计

根据系统需求分析及概要设计文档，设计进销存系统UML类图

**1、类图的概念**

类是具有相似结构、行为和关系的一组对象的描述符。类是面向对象系统中最重要的构造块。类图显示了一组类、接口、协作以及他们之间的关系。在UML中问题域最终要被逐步转化，通过类来建模，通过编程语言构建这些类从而实现系统。类加上他们之间的关系就构成了类图，类图中还可以包含接口、包等元素，也可以包括对象、链等实例。

在UML类图中，类一般由三部分组成：

(1) 第一部分是**类名**：每个类都必须有一个名字，类名是一个字符串。

(2) 第二部分是类的属性(Attributes)：属性是指类的性质，即类的成员变量。一个类可以有任意多个属性，也可以没有属性

UML规定**属性**的表示方式为：

可见性 名称:类型 [ = 缺省值 ]

其中：

“可见性”表示该属性对于类外的元素而言是否可见，包括公有(public)、私有(private)和受保护(protected)三种，在类图中分别用符号+、-和#表示。

“名称”表示属性名，用一个字符串表示。

“类型”表示属性的数据类型，可以是基本数据类型，也可以是用户自定义类型。

“缺省值”是一个可选项，即属性的初始值。

(3) 第三部分是**类的操作(Operations)**：操作是类的任意一个实例对象都可以使用的行为，是类的成员方法。

UML规定操作的表示方式为：

可见性 名称(参数列表) [ : 返回类型]

其中：

“可见性”的定义与属性的可见性定义相同。

“名称”即方法名，用一个字符串表示。

“参数列表”表示方法的参数，其语法与属性的定义相似，参数个数是任意的，多个参数之间用逗号“，”隔开。

“返回类型”是一个可选项，表示方法的返回值类型，依赖于具体的编程语言，可以是基本数据类型，也可以是用户自定义类型，还可以是空类型(void)，如果是构造方法，则无返回类型。

在类图2中，操作method1的可见性为public(+)，带入了一个Object类型的参数par，返回值为空(void)；操作method2的可见性为protected(#)，无参数，返回值为String类型；操作method3的可见性为private(-)，包含两个参数，其中一个参数为int类型，另一个为int[]类型，返回值为int类型。

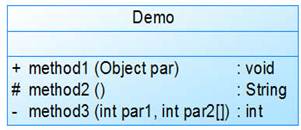


图2 类图操作说明示意图

由于在Java语言中允许出现内部类，因此可能会出现包含四个部分的类图，如图3所示：

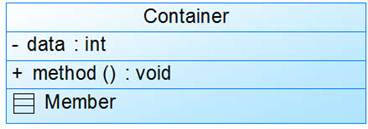


图3 包含内部类的类图

**2、类图关系描述**

在UML类图中，常见的有以下几种关系: 泛化（Generalization）,  实现（Realization），关联（Association)，聚合（Aggregation），组合(Composition)，依赖(Dependency)。

这几种关系在类图当中如何体现可参照知道文档进行讲解。



**例：StarUML完成商品及采购管理的类图**

****

****

此节由学生根据以上所学内容自行去设计。

## 进销存系统详细设计

此节由学生根据所学内容填充完善详细设计。

# 系统底层框架搭建

## 章节目标

√ 掌握系统三层架构的设计含义

√ 进销存系统的文件组织结构设计

√ 掌握类图的绘制方式，主要目标-能看懂

√ 进销存系统公共类及公共接口的设计及编写

## 三层架构说明

简述三层架构设计模式的意义，view层好比酒店的菜单，用户要什么菜需要根据菜单来点，业务逻辑层相当于服务员，服务员根据客户的请求向后台执行业务，数据访问层相当于厨师，业务需要执行的时候需要厨师提供最终的菜（数据）。

三层架构互相协作，同时各自独立完成自己的事情，降低层与层之间的依赖，有利于标准化，也有利于各层的复用，同时利于团队协作。

## 进销存系统文件组织结构设计

通常项目文件的组织结构如下：

Po:实体包，对应数据库中的表

Dao：数据访问包

Service：业务逻辑包

View：视图包，做显示和输入用

Util：通用工具包

学员根据此类设计进行进销存系统的文件组织结构设计。

## 进销存公共类和接口的设计及编写

往往系统设计过程分为以下过程：

静态界面的设计：前面已对其设计完毕。

公共类及接口设计（本节内容）：util包中的内容如：反射生成子窗体、窗体位置显示工具、导出工具类、日期选择器等。PO包中的内容:各个PO类的设计。BaseDao。

实现各模块业务：模块Dao、service及serviceImpl及最终的显示和输入。

举例实现一个采购业务，其他模块的Dao和service及实现类由学员独立去完成。

附BaseDao代码：

|  |
| --- |
| import java.sql.\*;  import java.util.List;  import org.apache.commons.dbutils.DbUtils;  import org.apache.commons.dbutils.QueryRunner;  import org.apache.commons.dbutils.handlers.BeanListHandler;  /\*\*  \* 负责数据库操作  \* @author yhy  \*/  public class BaseDao{  //获得连接对象  public Connection getConnection(){  ////////////  //////cdssadsa  Connection conn = null;  String jdbcDriver = "com.mysql.jdbc.Driver";  String jdbcURL = "jdbc:mysql://localhost:3306/pss";  String user = "root";  String password = "admin";  try {  DbUtils.loadDriver(jdbcDriver);  conn = DriverManager.getConnection(jdbcURL, user, password);  } catch (SQLException e) {  e.printStackTrace();  }  return conn;  }  /\*\*  \* 查找多个对象  \* @param sql  \* @param clazz  \* @return  \*/  public List query(String sql, Class clazz) {  List beans = null;  Connection conn = null;  try {  conn = getConnection();  //QueryRunner类是DBUtils中主要的操作数据的类  QueryRunner qRunner = new QueryRunner();  beans =  (List) qRunner.query(  conn,  sql,  new BeanListHandler(clazz));  } catch (SQLException e) {  e.printStackTrace();  } finally {  DbUtils.closeQuietly(conn);  }  return beans;  }    /\*\*  \* 查找对象  \* @param sql  \* @param clazz  \* @return  \*/  public Object get(String sql, Class clazz) {  List beans = null;  Object obj = null;  Connection conn = null;  try {  conn = getConnection();  QueryRunner qRunner = new QueryRunner();  beans =  (List) qRunner.query(  conn,  sql,  new BeanListHandler(clazz));  } catch (SQLException e) {  e.printStackTrace();  } finally {  DbUtils.closeQuietly(conn);  }  if(beans!=null && !beans.isEmpty()){ //注意这里  obj=beans.get(0);  }  return obj;  }    /\*\*  \* 执行更新的sql语句,插入,修改,删除  \* @param sql  \* @return  \*/  public boolean update(String sql,Object[] params) {  Connection conn = null;  boolean flag = false;  try {  conn = getConnection();  QueryRunner qRunner = new QueryRunner();  int i = qRunner.update(conn,sql,params);  if (i > 0) {  flag = true;  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  DbUtils.closeQuietly(conn);  }  return flag;  }  /\*\*  \* 更新数据库重载方法  \* @param conn 连接对象  \* @param sql sql语句  \* @param params 参数  \* @return  \*/  public boolean update(Connection conn,String sql,Object[] params) throws SQLException {  boolean flag = false;  conn = getConnection();  QueryRunner qRunner = new QueryRunner();  int i = qRunner.update(conn,sql,params);  if (i > 0) {  flag = true;  }  return flag;  }  /\*\*  \* 批量更新  \* @param sql  \* @param params  \* @return  \*/  public boolean batchUpdate(Connection conn,String sql,Object[][] params) throws SQLException{  QueryRunner qRunner = new QueryRunner();  int result = 0;  boolean flag = false;  result = qRunner.batch(conn, sql, params).length;  if(result > 0){  flag = true;  }  return flag;  }  } |

采购DAO，PurchaseDao.java:

|  |
| --- |
| public class PurchaseDao extends BaseDao{    } |

PurchaseService.java:

|  |
| --- |
| public interface PurchaseService {  //采购入库  public boolean purchasein(Object[][] purchases,Object[][] stocks);  // 查询采购单  public List<ViewPurchase> findByCondition(String start,String end,String conditions);  //采购退货  public boolean purchasereturn(Object[][] purchasereturn,Object[][] stocks);  //采购退货2  public boolean purchasereturn(int purid,int returnamont,Date returntime,int pid);  } |

PurchaseServiceImpl.java:

|  |
| --- |
| public class PurchaseServiceImpl implements PurchaseService{  ProductDao prodao = new ProductDao();//商品dao  PurchaseDao purdao = new PurchaseDao();//采购dao  @Override  public boolean purchasein(Object[][] purchases,Object[][]stocks) {  boolean result = true;  String sql1 = "insert into purchase(purno,pid,purchaseamount,purchaseprice,purtotals,supplierid,purchasedate) values(?,?,?,?,?,?,?)";  String sql2 = "update product set currentamount=currentamount+?,lastpurchasetime=? where pid=?";//库存增加  Connection conn = new BaseDao().getConnection();//获得连接    try {  conn.setAutoCommit(false);//设置不自动提交    purdao.batchUpdate(conn, sql1, purchases);//批量添加采购表  prodao.batchUpdate(conn, sql2, stocks);//更新商品库存  conn.commit(); //手动提交事务  } catch (Exception e) {  try {  result = false;  conn.rollback(); //回滚事务  } catch (SQLException ex) {  ex.printStackTrace();  }  e.printStackTrace();  }  return result;  }  @Override  public boolean purchasereturn(Object[][] purchasereturn,Object[][] stocks) {  boolean result = true;  String sql1 = "insert into purchase(purno,pid,purchaseamount,purchaseprice,purtotals,supplierid,purchasedate) values(?,?,?,?,?,?,?)";  String sql2 = "update product set currentamount=currentamount-? where pid=?";//库存减少  Connection conn = new BaseDao().getConnection();//获得连接    try {  conn.setAutoCommit(false);//设置不自动提交  purdao.batchUpdate(conn, sql1, purchasereturn);//批量添加采购表  prodao.batchUpdate(conn, sql2, stocks);//更新商品库存  conn.commit(); //提交事务  } catch (Exception e) {  try {  result = false;  conn.rollback(); //回滚事务  } catch (SQLException ex) {  ex.printStackTrace();  }  e.printStackTrace();  }  return result;  }  @Override  public List<ViewPurchase> findByCondition(String start, String end, String conditions) {  String sql = "select \* from view\_purchase where purchasedate between '"+start+"' and '"+end+"'";  if(conditions != null && conditions.length()>0){  sql +=" and concat(purno,pname,supname) like '%"+conditions+"%'";  }  return purdao.query(sql, ViewPurchase.class);  }  @Override  public boolean purchasereturn(int purid, int returnamount, Date returntime, int pid) {  boolean result = true;  //修改采购表的退货数量和退货时间  String sql1 = "update purchase set returnamount=?,returntime=?,purchaseamount=purchaseamount-? where purid=?";  Object[] params1 = {returnamount,returntime,returnamount,purid};  String sql2 = "update product set currentamount= currentamount-? where pid=?";  Object[] params2 = {returnamount,pid};  Connection conn = new BaseDao().getConnection();  try {    conn.setAutoCommit(false);//启动事务  purdao.update(conn, sql1, params1);  prodao.update(conn, sql2, params2);  conn.commit();  } catch (Exception e) {  try {  result = false;  conn.rollback();  } catch (Exception ex) {  ex.printStackTrace();  }  }  return result;  }    } |

# 系统重难点分析

## 章节目标

√ 采购业务实现

√ 如何实现导出excel

√ 利用JFreeChart生成图表

√ 业务事务的处理

## 采购业务的实现

上一节已经将底层的接口和实现类已完成，可以通过实现其中一个业务，其他的内容由学员去实现。

采购入库界面：



采购入库部分实现代码：

|  |
| --- |
| private void btnPurchaseInActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  //获取信息  DefaultTableModel dtm = (DefaultTableModel) this.tblPurchase.getModel();      Object[][] purchase = new Object[dtm.getRowCount()][7];//根据具体数值数创建采购数组  Object[][] stocks = new Object[dtm.getRowCount()][3];//商品库存数组    //遍历表格  for(int i=0;i<dtm.getRowCount();i++){  purchase[i][0] = NoUtil.createPurno();//采购单号  purchase[i][1] = this.tblPurchase.getValueAt(i, 0);//商品编号  stocks[i][2] = this.tblPurchase.getValueAt(i, 0); //第3个?  purchase[i][2] = this.tblPurchase.getValueAt(i, 4);//采购数量  stocks[i][0] = this.tblPurchase.getValueAt(i, 4);  purchase[i][3] = this.tblPurchase.getValueAt(i, 5);//采购价格  //总金额，价格 \* 数量  purchase[i][4] = new BigDecimal(purchase[i][3].toString()).multiply(new BigDecimal(purchase[i][2].toString()));    Supplier sup = (Supplier)this.tblPurchase.getValueAt(i, 6);    purchase[i][5] = sup.getSupplierid();  purchase[i][6] = new java.util.Date();  stocks[i][1] = new java.util.Date();  }  //调用批量添加  boolean result = purbiz.purchasein(purchase, stocks);    if(result == true){  JOptionPane.showMessageDialog(this, "采购入库成功");  //更新商品表  btnQueryActionPerformed(evt);//调用查询方法  }else{  JOptionPane.showMessageDialog(this, "采购入库失败");  }  //清空采购表  while(dtm.getRowCount()>0){  dtm.removeRow(0);  }  } |

## 导出excel及JfreeChart生成图表

主要目的：作技术统计使用

此节内容可通过指导文档，先由学生自己去根据文档实现导出和生成图表，第二天调查完成情况，根据完成情况自己决定是否需要现场演示。



文档会附赠生成图表的案例。

至此，进销存讲解内容结束，由学生去实现其他业务，完成整个进销存系统的实现。