息管理”菜单所包含的菜单项如图 3-7 所示。“奖惩信息管理”菜单所包含的菜单项如图 3-8 所示。

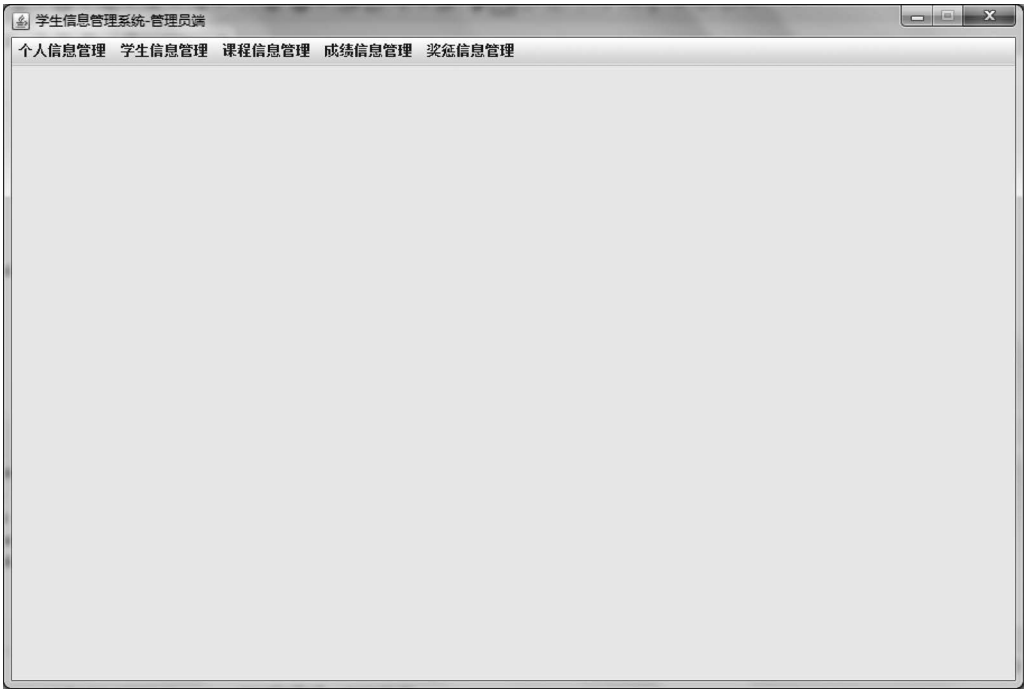



图 3-3　学生信息管理系统的主窗口-管理员端

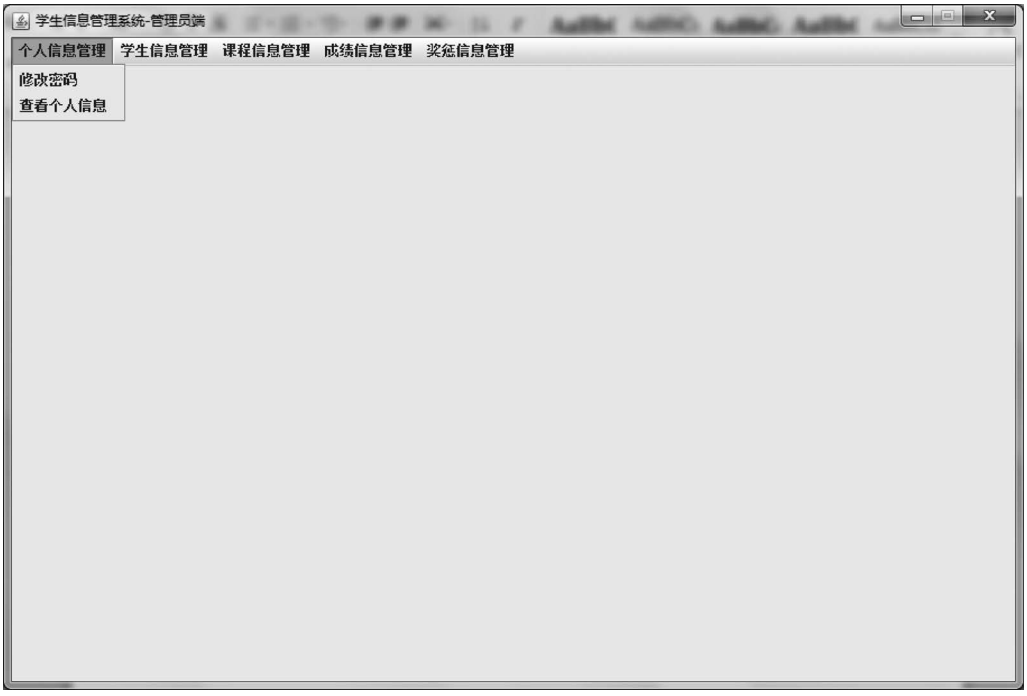



图 3-4　“个人信息管理”菜单所包含的菜单项

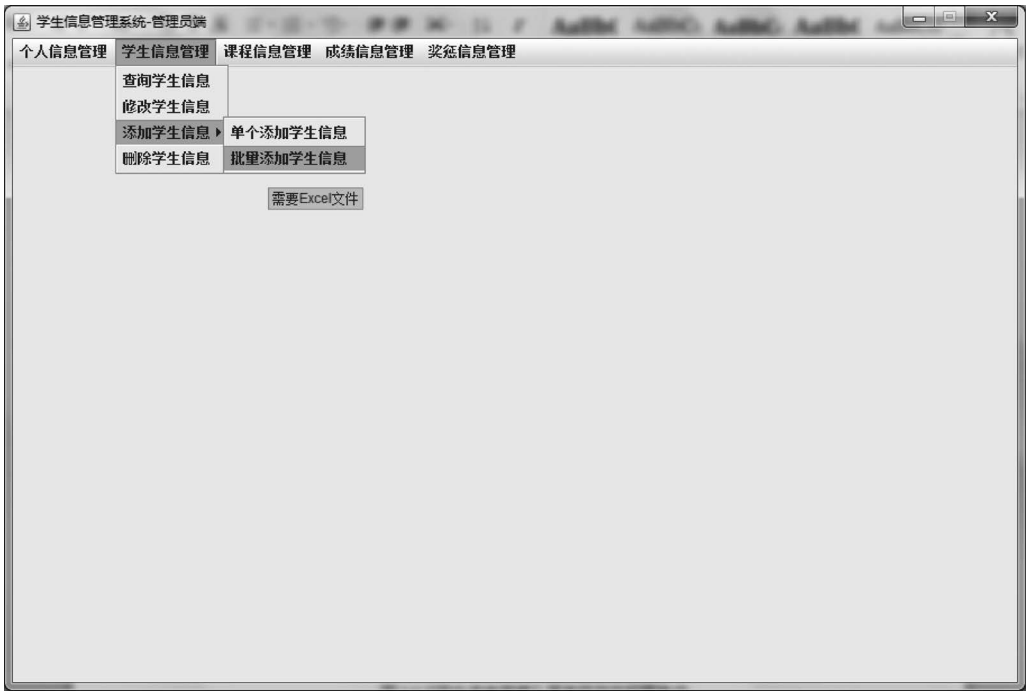



图 3-5 “学生信息管理”菜单所包含的菜单项

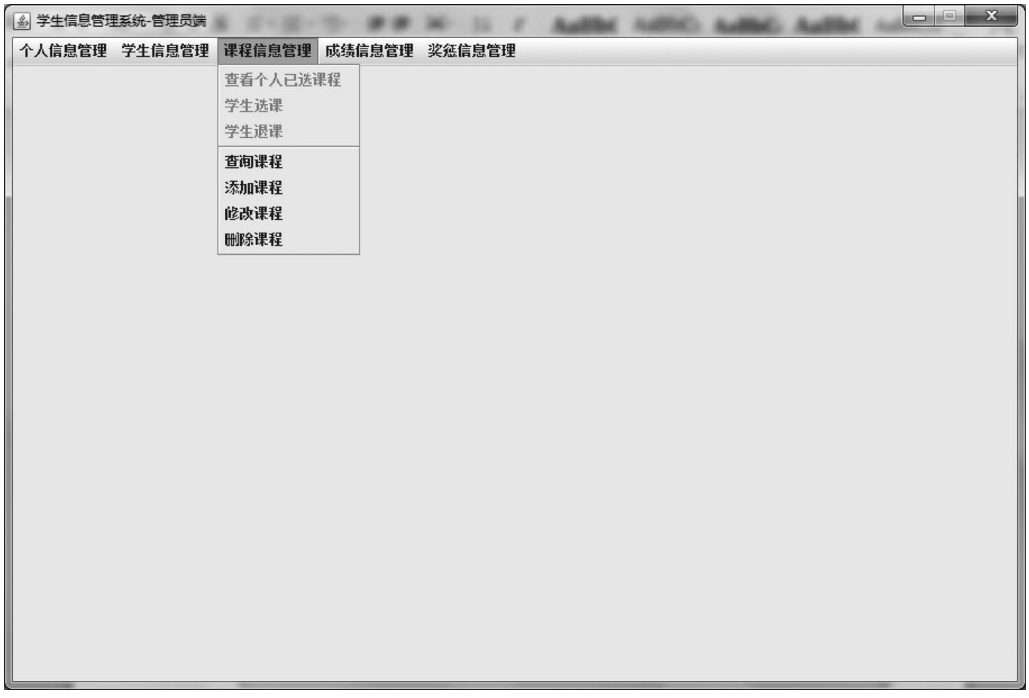



图 3-6 “课程信息管理”菜单所包含的菜单项


图 3-7 “成绩信息管理”菜单所包含的菜单项


图 3-8 “奖惩信息管理”菜单所包含的菜单项

## 3.4 本项目的设计方案

**1. 本项目的功能结构**

根据需求分析可知，本项目的用户分为 3 类，分别是管理员、教师和学生，不同用户实现的功能也不一样。具体的系统功能如图 3-9 所示。

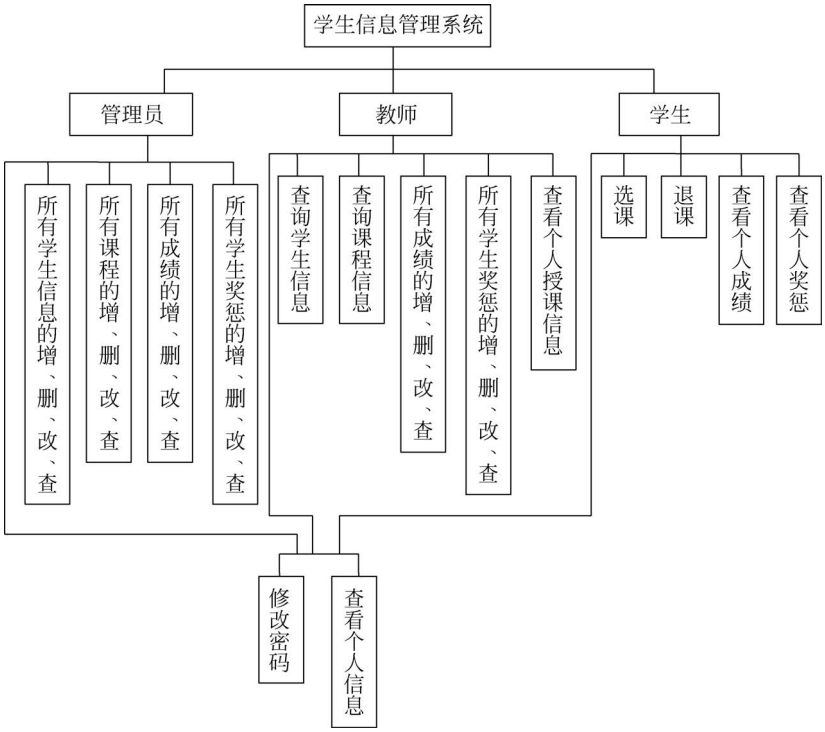



图 3-9 系统功能结构

**2. 本项目的设计步骤**

第一步：创建数据库以及相关数据表。

这里采用了目前流行的 MySQL 数据库。

(1) 创建数据库 student。

```
create database if not exists `student`;
```

(2) 创建数据表。

① 用户信息表 user。

用户信息表结构如图 3-10 所示。

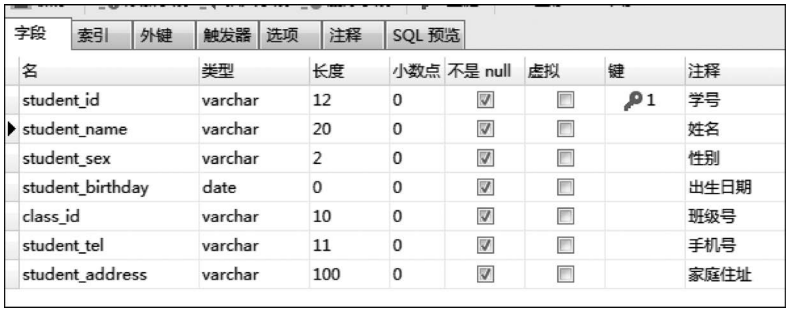
字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| user_id | varchar | 12 | 0 | ☑ | ☐ | 🔑1 | 用户账号 |
| password | varchar | 10 | 0 | ☑ | ☐ | | 密码 |
| level | varchar | 10 | 0 | ☑ | ☐ | | 用户类别 |

图 3-10 用户信息表结构

② 学生信息表 student。

学生信息表结构如图 3-11 所示。



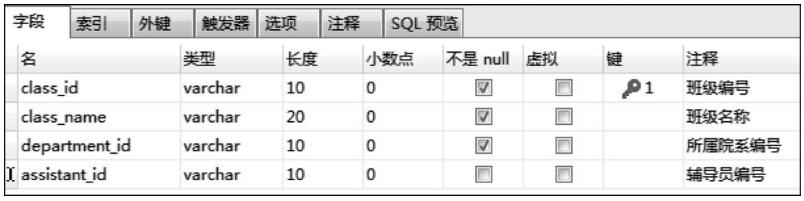
字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| student_id | varchar | 12 | 0 | ☑ | ☐ | 🔑1 | 学号 |
| student_name | varchar | 20 | 0 | ☑ | ☐ | | 姓名 |
| student_sex | varchar | 2 | 0 | ☑ | ☐ | | 性别 |
| student_birthday | date | 0 | 0 | ☑ | ☐ | | 出生日期 |
| class_id | varchar | 10 | 0 | ☑ | ☐ | | 班级号 |
| student_tel | varchar | 11 | 0 | ☑ | ☐ | | 手机号 |
| student_address | varchar | 100 | 0 | ☑ | ☐ | | 家庭住址 |

图 3-11　学生信息表结构

③ 班级信息表 sclass。

班级信息表结构如图 3-12 所示。



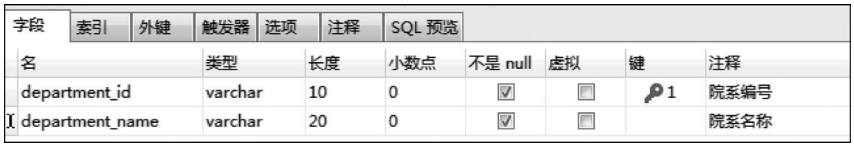
字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| class_id | varchar | 10 | 0 | ☑ | ☐ | 🔑1 | 班级编号 |
| class_name | varchar | 20 | 0 | ☑ | ☐ | | 班级名称 |
| department_id | varchar | 10 | 0 | ☑ | ☐ | | 所属院系编号 |
| assistant_id | varchar | 10 | 0 | ☐ | ☐ | | 辅导员编号 |

图 3-12　班级信息表结构

④ 院系(部门)信息表 department。

院系(部门)信息表结构如图 3-13 所示。



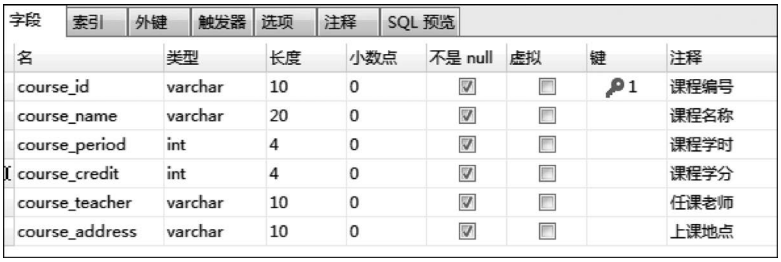
字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| department_id | varchar | 10 | 0 | ☑ | ☐ | 🔑1 | 院系编号 |
| department_name | varchar | 20 | 0 | ☑ | ☐ | | 院系名称 |

图 3-13　院系(部门)信息表结构

⑤ 课程信息表 course。

课程信息表结构如图 3-14 所示。



字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| course_id | varchar | 10 | 0 | ☑ | ☐ | 🔑1 | 课程编号 |
| course_name | varchar | 20 | 0 | ☑ | ☐ | | 课程名称 |
| course_period | int | 4 | 0 | ☑ | ☐ | | 课程学时 |
| course_credit | int | 4 | 0 | ☑ | ☐ | | 课程学分 |
| course_teacher | varchar | 10 | 0 | ☑ | ☐ | | 任课老师 |
| course_address | varchar | 10 | 0 | ☑ | ☐ | | 上课地点 |

图 3-14　课程信息表结构

⑥ 教师信息表 teacher。

教师信息表结构如图 3-15 所示。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| teacher_id | varchar | 12 | 0 | ☑ | ☐ | 1 | 教师编号 |
| teacher_name | varchar | 20 | 0 | ☑ | ☐ | | 姓名 |
| teacher_sex | varchar | 2 | 0 | ☑ | ☐ | | 性别 |
| teacher_birthday | date | 0 | 0 | ☑ | ☐ | | 出生日期 |
| department_id | varchar | 255 | 0 | ☑ | ☐ | | 所属部门编号 |
| teacher_position | varchar | 10 | 0 | ☑ | ☐ | | 职称 |
| teacher_tel | varchar | 11 | 0 | ☑ | ☐ | | 手机号 |
| teacher_address | varchar | 100 | 0 | ☑ | ☐ | | 家庭住址 |

图 3-15　教师信息表结构

⑦ 成绩信息表 score。

成绩信息表结构如图 3-16 所示。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| student_id | varchar | 12 | 0 | ☑ | ☐ | 1 | 学生编号 |
| course_id | varchar | 10 | 0 | ☑ | ☐ | 2 | 课程编号 |
| score | float | 0 | 0 | ☑ | ☐ | | 成绩 |

图 3-16　成绩信息表结构

⑧ 学生选课信息表 xclass。

学生选课信息表结构如图 3-17 所示。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| student_id | varchar | 12 | 0 | ☑ | ☐ | 1 | 学生编号 |
| course_id | varchar | 10 | 0 | ☑ | ☐ | 2 | 课程编号 |

图 3-17　学生选课信息表结构

⑨ 奖惩信息表 prize。

奖惩信息表结构如图 3-18 所示。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| prize_id | int | 10 | 0 | ☑ | ☐ | 1 | 序号（自动递增） |
| student_id | varchar | 12 | 0 | ☑ | ☐ | | 学生编号 |
| prize_date | date | 0 | 0 | ☑ | ☐ | | 奖惩时间 |
| prize_name | varchar | 20 | 0 | ☑ | ☐ | | 奖惩名称 |
| prize_level | varchar | 10 | 0 | ☑ | ☐ | | 奖惩级别 |

图 3-18　奖惩信息表结构

⑩ 辅导员信息表 assistant。

辅导员信息表结构如图 3-19 所示。

第二步：创建 Java 项目，实现系统的功能。

(1) 创建 Java 项目名为 SIMS。

(2) 创建包。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---|---|---|---|---|---|---|---|
| assistant_id | varchar | 10 | 0 | ☑ | ☐ | 1 | 辅导员编号 |
| assistant_name | varchar | 20 | 0 | ☑ | ☐ | | 辅导员姓名 |
| assistant_sex | varchar | 2 | 0 | ☑ | ☐ | | 辅导员性别 |
| assistant_birthday | date | 0 | 0 | ☑ | ☐ | | 辅导员出生日期 |
| department_id | varchar | 10 | 0 | ☑ | ☐ | | 所属院系 |
| assistant_tel | varchar | 11 | 0 | ☑ | ☐ | | 电话 |
| assistant_address | varchar | 50 | 0 | ☑ | ☐ | | 家庭住址 |

图 3-19　辅导员信息表结构

一般情况下，开发一个信息管理类的项目需要很多类，为了对这些类进行分层管理，按照业务逻辑习惯，一般需要在项目中创建如图 3-20 所示的 4 个包。

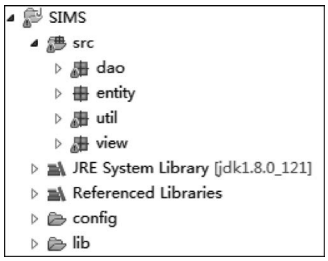



图 3-20　项目结构

其中，view 包存放图形界面程序。entity 包存放实体类。dao 包存放访问数据库的业务逻辑类。util 包存放辅助类。

（3）在各包中创建相应的类。

① view 包中的类。

view 包中所涉及的类及其功能描述如表 3-1 所示。

**表 3-1　view 包中所涉及的类及其功能描述**

| 类　　名 | 功　　能 |
|---|---|
| Login | 登录窗口，根据用户类别的不同进入相应的主窗口 |
| Mainframe | 系统主窗口，不同类别的用户登录后能完成各自不同的功能 |
| DisplayStuSelfInfo | 以学生身份登录后显示该学生的个人信息 |
| DisplayTeacherSelfInfo | 以教师或管理员身份登录后显示该教师或管理员的个人信息 |
| DisplaySelfScore | 以学生身份登录后显示该学生的个人成绩 |
| DisplaySelfCourse | 以学生身份登录后显示该学生的个人选课信息 |
| DisplaySelfPrize | 以学生身份登录后显示该学生的个人奖惩信息 |
| InsertStudent | 以管理员的身份登录后添加学生信息 |
| InsertCourse | 以管理员的身份登录后添加课程信息 |
| InsertScore | 以管理员或教师的身份登录后添加成绩信息 |
| InsertPrize | 以管理员或教师的身份登录后添加学生的奖惩信息 |
| ModifyStudent | 以管理员的身份登录后修改学生信息 |
| ModifyCourse | 以管理员的身份登录后修改课程信息 |
| ModifyScore | 以管理员或教师的身份登录后修改成绩信息 |
| ModifyPassword | 修改个人的登录密码 |
| SelectStudent | 以管理员或教师的身份登录后查询学生信息 |
| SelectCourse | 以管理员或教师的身份登录后查询课程信息 |
| SelectScore | 以管理员或教师的身份登录后修改学生成绩信息 |
| selectModifyDeletePrize | 以管理员或教师的身份登录后查询/修改/删除学生的奖惩信息 |
| SelectSelfCourse | 以学生的的身份登录后完成个人选课 |
| DeleteStudent | 以管理员的身份登录后删除指定学生的信息 |
| DeleteCourse | 以管理员的身份登录后删除指定课程的信息 |
| DeleteScore | 以管理员或教师的身份登录后删除指定学生的成绩信息 |

② entity 包中的类。

entity 包中所涉及的类及其功能描述如表 3-2 所示。

**表 3-2　entity 包中所涉及的类及其功能描述**

| 类　　名 | 功　　能 |
|---|---|
| Course | 课程信息实体类，对应数据表 course |
| Department | 部门信息实体类，对应数据表 depatment |
| Prize | 奖惩信息实体类，对应数据表 prize |
| SClass | 班级信息实体类，对应数据表 sclass |
| Sscore | 成绩信息实体类，对应数据表 sscore |
| Student | 学生信息实体类，对应数据表 student |
| Teacher | 教师信息实体类，对应数据表 teacher |
| User | 用户信息实体类，对应数据表 user |
| XClass | 学生选课信息实体类，对应数据表 xclass |

③ dao 包中的类。

dao 包中所涉及的类及其功能描述如表 3-3 所示。

**表 3-3　dao 包中所涉及的类及其功能描述**

| 类　　名 | 功　　能 |
|---|---|
| DBUtil | 封装了对数据库的连接操作 |
| CourseDao | 封装了对数据表 course 的增、删、改、查等操作 |
| DepartmentDao | 封装了对数据表 department 的增、删、改、查等操作 |
| PrizeDao | 封装了对数据表 prize 的增、删、改、查等操作 |
| SClassDao | 封装了对数据表 sclass 的增、删、改、查等操作 |
| SscoreDao | 封装了对数据表 sscore 的增、删、改、查等操作 |
| StudentDao | 封装了对数据表 student 的增、删、改、查等操作 |
| TeacherDao | 封装了对数据表 teacher 的增、删、改、查等操作 |
| UserDao | 封装了对数据表 user 的增、删、改、查等操作 |
| XClassDao | 封装了对数据表 xclass 的增、删、改、查等操作 |

④ util 包中的类。

util 包中所涉及的类及其功能描述如表 3-4 所示。

**表 3-4　util 包中所涉及的类及其功能描述**

| 类　　名 | 功　　能 |
|---|---|
| Config | 读取项目中的 config 文件夹中的 mysql. properties 文件中的访问数据库的相关信息 |
| DepartmentAndClassLinked | 根据所选的院系动态获取该院系的班级名称，实现院系和班级的级联效果 |
| Start | 启动项目 |

第三步：对本项目设置 MySQL 数据库的驱动类路径。

数据库驱动是由数据库提供商开发的，所以，这个包可以在数据库提供商那里免费下载。随着数据库的升级，这些驱动包也会随之升级，因此，存在不同的版本。如 MySQL 的数据库驱动就可以在 MySQL 的相关网站上下载。

本项目中采用的是 mysql-connector-Java-5. 0. 3-bin. jar 数据库驱动包。

在下载了数据库驱动包后，可以将驱动包放到本项目的 lib 文件夹下，然后将数据库驱动导入 Java 项目中。

在项目上右击，在弹出的快捷菜单中依次单击 Build Path→Configure Build Path，弹出如图 3-21 所示的 Properties for SIMS 对话框，然后在该对话框中选择 Libraries 选项卡，再单击 Add JARs 按钮，在弹出的如图 3-22 所示的 JAR Selection 对话框中，找到本项目中的 lib 文件夹中的数据库驱动包(mysql-connector-Java-5.0.3-bin.jar)，将其选中并打开，即可完成数据库驱动的导入。

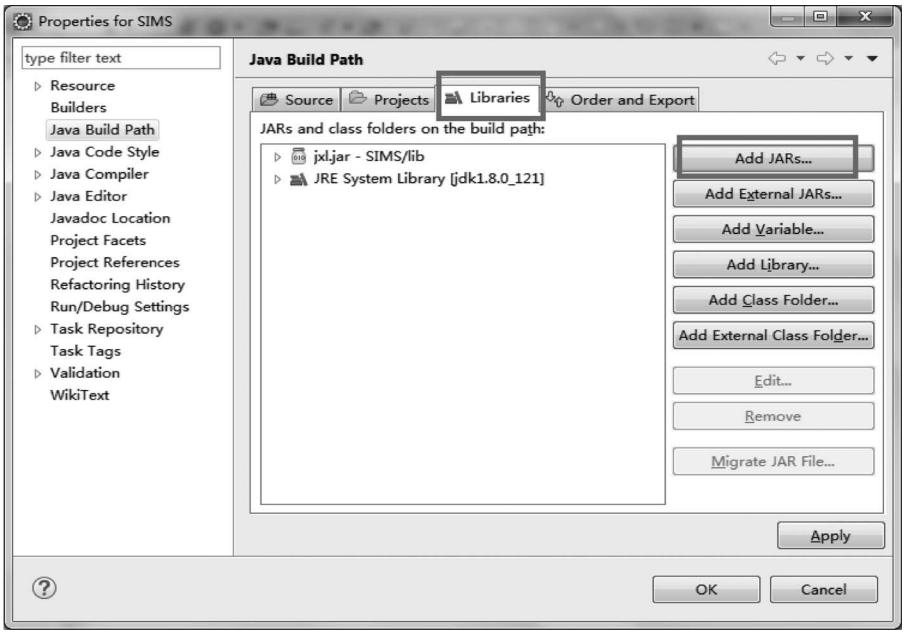



图 3-21　Properties for SIMS 对话框

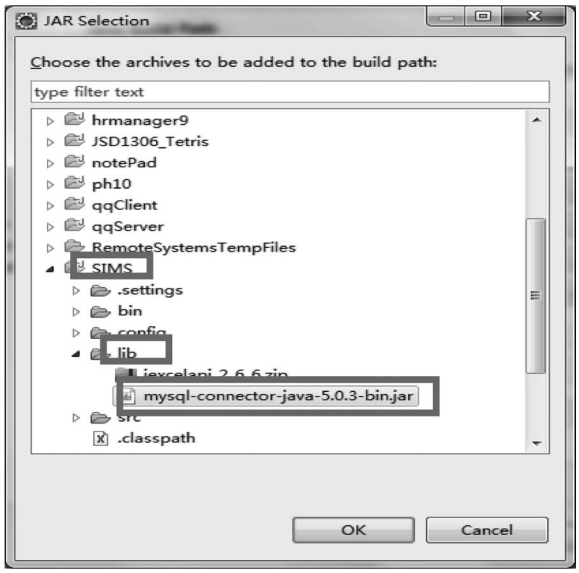



图 3-22　JAR Selection 对话框

# 3.5 本项目的实现过程



## 一、创建登录窗口、主窗口并实现登录功能

### 1. 在 view 包中创建登录窗口类 Login

代码如下：

```
package view;
import java.awt.Color;
import java.awt.Font;
import javax.swing.*;
import dao.UserDao;
import entity.User;
//登录窗口的设计
public class Login extends JFrame{
    //定义窗口中用到的组件
    private JPanel jp;
    private JButton btnLogin,btnCancel;
    private JLabel label,lblLogin,lblPassword,lblLevel;
    private JTextField txtLogin;
    private JPasswordField txtPassword;
    private JComboBox level;
    //构造方法:初始化登录窗口
    public Login(){
        setTitle("学生信息管理系统");                          //设置窗口的标题
        //创建组件、容器对象
        jp = new JPanel(null);                                 //将 jp 容器设置为空布局
        label = new JLabel("欢迎登录学生信息管理系统",JLabel.CENTER);
        Font f = new Font("宋体",Font.BOLD,20);                //创建一个字体对象
        label.setFont(f);                                      //设置标签的字体
        lblLogin = new JLabel("账 号:");                       //登录标签
        txtLogin = new JTextField();                           //登录文本框
        lblPassword = new JLabel("密 码:");                    //密码标签
        txtPassword = new JPasswordField();                    //密码输入框
        lblLevel = new JLabel("级 别");
        String str[] = {"学生","教师","管理员"};
        level = new JComboBox(str);                            //用户级别组合框
        btnLogin = new JButton("登录");                        //"登录"按钮
        btnCancel = new JButton("取消");                       //"取消"按钮
        //向窗口中添加面板 jp
        this.add(jp);
        //向面板中添加各个组件
        jp.add(label);
        jp.add(lblLogin);
        jp.add(txtLogin);
        jp.add(lblPassword);
        jp.add(txtPassword);
        jp.add(lblLevel);
        jp.add(level);
        jp.add(btnLogin);
        jp.add(btnCancel);
        //设置各个组件的坐标及大小
```

```
        label.setBounds(0,30,400,20);
        lblLogin.setBounds(73,70,60,15);
        txtLogin.setBounds(150, 70, 140, 20);
        lblPassword.setBounds(73,100,60,15);
        txtPassword.setBounds(150,100,140,21);
        lblLevel.setBounds(73,130,60,15);
        level.setBounds(150,130,140,21);
        btnLogin.setBounds(100,240,60,23);
        btnCancel.setBounds(210,240,60,23);
        //设置面板的背景色
        jp.setBackground(Color.PINK);
        this.setSize(400,350);                                  //设置窗口的大小
        this.setLocationRelativeTo(null);                       //窗口居中
        this.setResizable(false);                               //禁止改变框架大小
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);}
```

**2. 在 util 包中创建项目启动类 start**

代码如下：

```
package util;
import view.Login;
public class start {
    public static void main(String[] args) {
        Login f = new Login();
        f.setVisible(true);
    }
}
```

执行该类，就会得到如图 3-2 所示的登录窗口。目前该窗口只是一个静态窗口，不能实现“登录”和“取消”的动作处理。

**3. 在 entity 包中创建实体类 User**

该类是表 user 对应的实体类，其中的属性对应表 user 中的字段。通常情况下类名与表名一致，属性名和字段名(列名)一一对应。定义实体类的好处有：

① 对对象实体进行封装，体现面向对象程序设计思想。

② 属性可以对字段定义和状态进行判断和过滤。

③ 把相关信息用一个实体类封装后，方便用户对象在程序中作为参数传递。

代码如下：

```
package entity;
//类名尽量与对应的数据表名一致
public class User {
  //定义属性:习惯于属性名与对应数据表中的列名(字段名)一致
  private String userId;
  private String password;
  private String level;
  //定义各属性的 getXxx()/setXxx()方法
  public String getUserId() {
    return userId;
  }
  public void setUserId(String userId) {
    this.userId = userId;
  }
```

```
    public String getPassword() {
      return password;
    }
    public void setPassword(String password) {
      this.password = password;
    }
    public String getLevel() {
      return level;
    }
    public void setLevel(String level) {
      this.level = level;
    }
    //定义有参构造方法
    public User(String userId, String password, String level) {
      this.userId = userId;
      this.password = password;
      this.level = level;
    }
    //定义无参构造方法
    public User() {
    }
}
```

下面创建系统的主窗口。

**4. 在 view 包中创建主窗口类 MainFrame**

本窗口利用如下代码实现,目前它只是一个静态窗口,不能实现各菜单项的功能。

```
package view;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.sql.Date;
import java.util.ArrayList;
import javax.swing.*;
import dao.StudentDao;
import entity.Student;
import entity.User;
import jxl.*;
import jxl.read.biff.BiffException;
//系统窗口
public class MainFrame extends JFrame {
  //定义菜单栏、菜单、菜单项等属性
  private JMenuBar menuBar;
  private JMenu selfInfoMenu,studentMenu,courseMenu,scoreMenu,prizeMenu;
  private JMenu insertStudent;
  private JMenuItem modifyPassword,displaySelfInfo;
  private JMenuItem selectStudent,modifyStudent,deleteStudent,insertOneStudent,
        insertMoreStudent;
  private JMenuItem selectCourse,modifyCourse,insertCourse,deleteCourse,
        displaySelfCourse,selectSelfCourse,quitCourse;
  private JMenuItem displaySelfScore,selectScore,modifyScore,insertScore,deleteScore;
  private JMenuItem displaySelfPrize,selectModifyDeletePrize,insertPrize;
  //构造方法:初始化主窗口,并且通过参数接收当前登录用户的信息,为实现修改个
  //人密码、查看个人信息、根据用户的类别赋予不同的功能等
```

```
public MainFrame(final User user){
    menuBar = new JMenuBar();                                    //创建菜单栏对象
    this.setJMenuBar(menuBar);                                   //将菜单栏添加到窗口上面
    //创建各菜单对象
    selfInfoMenu = new JMenu("个人信息管理");
    studentMenu = new JMenu("学生信息管理");
    courseMenu = new JMenu("课程信息管理");
    scoreMenu = new JMenu("成绩信息管理");
    prizeMenu = new JMenu("奖惩信息管理");
    //将菜单添加到菜单栏中
    menuBar.add(selfInfoMenu);
    menuBar.add(studentMenu);
    menuBar.add(courseMenu);
    menuBar.add(scoreMenu);
    menuBar.add(prizeMenu);
    //创建菜单项
    displaySelfInfo = new JMenuItem("查看个人信息");
    modifyPassword = new JMenuItem("修改密码");
    selectStudent = new JMenuItem("查询学生信息");
    modifyStudent = new JMenuItem("修改学生信息");
    deleteStudent = new JMenuItem("删除学生信息");
    //创建"学生信息管理"菜单中的二级菜单"添加学生信息"
    insertStudent = new JMenu("添加学生信息");
    //创建"添加学生信息"二级菜单的菜单项
    insertOneStudent = new JMenuItem("单个添加学生信息");
    insertMoreStudent = new JMenuItem("批量添加学生信息");
    //当鼠标指针指向"批量添加学生信息"菜单项时即时给出提示信息:需要 Excel 文件
    insertMoreStudent.setToolTipText("需要 Excel 文件");
    //创建菜单项
    displaySelfCourse = new JMenuItem("查看个人已选课程");
    selectSelfCourse = new JMenuItem("学生选课");
    quitCourse = new JMenuItem("学生退课");
    selectCourse = new JMenuItem("查询课程");
    modifyCourse = new JMenuItem("修改课程");
    insertCourse = new JMenuItem("添加课程");
    deleteCourse = new JMenuItem("删除课程");
    displaySelfScore = new JMenuItem("查看个人成绩");
    selectScore = new JMenuItem("查询成绩");
    modifyScore = new JMenuItem("修改成绩 ");
    insertScore = new JMenuItem("录入成绩");
    deleteScore = new JMenuItem("删除成绩");
    displaySelfPrize = new JMenuItem("查看个人奖惩信息");
    selectModifyDeletePrize = new JMenuItem("查询/修改/删除奖惩信息");
    insertPrize = new JMenuItem("录入奖惩信息");
    //将菜单项添加到相应的菜单中
    selfInfoMenu.add(modifyPassword);
    selfInfoMenu.add(displaySelfInfo);
    studentMenu.add(selectStudent);
    studentMenu.add(modifyStudent);
    //将"添加学生信息"菜单添加到"学生信息管理"菜单中,构成级联菜单
    studentMenu.add(insertStudent);
    studentMenu.add(deleteStudent);
    insertStudent.add(insertOneStudent);
    insertStudent.add(insertMoreStudent);
```

```
courseMenu.add(displaySelfCourse);
courseMenu.add(selectSelfCourse);
courseMenu.add(quitCourse);
courseMenu.addSeparator();
courseMenu.add(selectCourse);
courseMenu.add(insertCourse);
courseMenu.add(modifyCourse);
courseMenu.add(deleteCourse);
scoreMenu.add(displaySelfScore);
scoreMenu.addSeparator();
scoreMenu.add(selectScore);
scoreMenu.add(modifyScore);
scoreMenu.add(insertScore);
scoreMenu.add(deleteScore);
prizeMenu.add(displaySelfPrize);
prizeMenu.addSeparator();
prizeMenu.add(selectModifyDeletePrize);
prizeMenu.add(insertPrize);
this.setSize(900, 600);
setLocationRelativeTo(null);                              //居中
setResizable(false);                                      //禁止改变框架大小
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//根据用户的不同,对主窗口设置不同的标题以及各自权限内不可用的菜单项
    if(user.getLevel().equals("学生")){
        this.setTitle("学生信息管理系统 - 学生端");
        selectStudent.setEnabled(false);
        modifyStudent.setEnabled(false);
        insertStudent.setEnabled(false);
        deleteStudent.setEnabled(false);
        modifyCourse.setEnabled(false);
        insertCourse.setEnabled(false);
        deleteCourse.setEnabled(false);
        selectScore.setEnabled(false);
        modifyScore.setEnabled(false);
        insertScore.setEnabled(false);
        deleteScore.setEnabled(false);
        selectModifyDeletePrize.setEnabled(false);
        insertPrize.setEnabled(false);
  }else if(user.getLevel().equals("教师")){
        this.setTitle("学生信息管理系统 - 教师端");
        modifyStudent.setEnabled(false);
        insertStudent.setEnabled(false);
        deleteStudent.setEnabled(false);
        modifyCourse.setEnabled(false);
        insertCourse.setEnabled(false);
        deleteCourse.setEnabled(false);
        displaySelfCourse.setEnabled(false);
        selectSelfCourse.setEnabled(false);
        quitCourse.setEnabled(false);
        displaySelfScore.setEnabled(false);
        displaySelfPrize.setEnabled(false);
  }else if(user.getLevel().equals("管理员")){
        this.setTitle("学生信息管理系统 - 管理员端");
        displaySelfCourse.setEnabled(false);
```

```
                selectSelfCourse.setEnabled(false);
                quitCourse.setEnabled(false);
                displaySelfScore.setEnabled(false);
                displaySelfPrize.setEnabled(false);
            }
        }
    }
```

**5. 利用 JDBC 技术访问表 user，实现登录功能**

(1) 在项目中创建 config 文件夹。

(2) 在 config 文件夹中创建数据库的属性文件：mysql.properties。

在该属性文件中以“键值对”的形式存放连接 MySQL 数据库的配置信息。这样做的好处是在不改变 Java 代码的情况下方便地更换数据库的相关信息。

文件内容如下：

```
driver = com.mysql.jdbc.Driver
url = jdbc:mysql://localhost:3306/student??useUnicode = true&characterEncoding = utf8
username = root
password = 123456
```

(3) 在 util 包中创建 config 类。

为了读取 mysql.properties 属性文件中的数据，此时需要编写一个 config 类，该类通过 java.util.Properties 类提供的 get()方法获取指定键所对应的值，为访问数据库时的“创建连接”步骤做准备。

config 类的代码如下：

```
package util;
import java.io.FileInputStream;
import java.util.Properties;
public class Config {
  private static Properties p = null;
  static {
      try{
          p = new Properties();
          //加载配置文件
          p.load(new FileInputStream("config\\mysql.properties"));
      }catch(Exception e){
          e.printStackTrace();
      }
  }
  //获取键对应的值
  public static String getValue(String key){
      return p.get(key).toString();
  }
}
```

(4) 在 dao 包中创建 DBUtil 类。

在访问数据库时，都要顺序执行“加载驱动、创建连接、创建 Statement 对象、执行 SQL 命令、处理结果、关闭创建的资源”等相同的步骤，无非是执行的 SQL 语句不同而已。因此，为了简化数据库访问操作，提高程序效率，就需要将访问数据库时的共同的基础代码进行封装，即编写一个访问数据库的工具类 DBUtil，该类可以提供访问数据库时所用到的连接、查

询、更新、关闭等操作的基本方法，这样其他类通过调用该类中的方法就可以进行数据库的访问了。

这里只给出了访问数据库时的“加载驱动、创建连接”的方法 getConnection()，其他方法如查询、更新、关闭等没有进一步给出，如果需要可自行定义。

DBUtil 类的代码如下：

```
package dao;
import java.sql.*;
import javax.swing.JOptionPane;
import util.Config;
public class DBUtil {
  public Connection getConnection(){
    //通过 Config 类获取 MySQL 数据库的配置信息
    String DRIVER = Config.getValue("driver");
    String URL = Config.getValue("url");
    String USERNAME = Config.getValue("username");
    String PASSWORD = Config.getValue("password");
    Connection con = null;
    try {
        //加载驱动
        Class.forName(DRIVER);
        //建立数据库连接
        con = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        } catch (Exception e) {
                //如果连接过程出现异常，则弹出异常信息对话框
                JOptionPane.showMessageDialog(null,"驱动错误或连接失败!");
            }
    return con;
  }
}
```

(5) 在 dao 包中创建访问 user 表的类：UserDao 类。

在创建 Java 项目时，希望把对数据库的访问封装起来，这些实现对数据库的访问的类一般被称为 Dao 类。

Dao 类的主要目的是封装数据库访问，使得数据访问和业务逻辑分离，以达到解耦的目的。这样可以提高代码的可重用性，减少重复代码，提高系统的可维护性。

一般情况下，数据库中的每个表都对应一个 Dao 类，完成对该表的增、删、改、查等操作。项目中的其他的类可以调用 Dao 类中的方法实现对数据的操作，一般来说，数据库中的每张数据表都要对应一个 Dao 类。这里对 user 表对应地创建一个 Dao 类：UserDao，封装对 user 表的各种增、删、改、查等操作。

UserDao 类的代码如下：

```
package dao;
import java.sql.*;
import javax.swing.JOptionPane;
import entity.User;
public class UserDao {
    //定义方法:根据账号、密码以及级别在 user 表中进行查询,如果查询到该
    //用户,则返回该用户的信息,否则,返回一个 null 值
    public User selectUserByIdAndPasswordAndLevel(String userId,String password,String level ){
```

```
        User user = null;
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
         //实例化数据库工具
        DBUtil dbUtil = new DBUtil();
        try{
            //打开数据库,获取连接对象
            con = dbUtil.getConnection();
            //创建 PreparedStatment 对象
            String sql = "select * from user where user_id = ? and password = ? and level = ?";
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1,userId);
            pstmt.setString(2, password);
            pstmt.setString(3, level);
            //执行查询命令,获取查询结果集
            rs = pstmt.executeQuery();
            //查看结果集中是否存在该用户,如果存在,则将该用户的信息封装成一
            //个 User 对象
            if(rs.next())
                user = new User(rs.getString(1),rs.getString(2),rs.getString(3));
        }catch(Exception e){
            JOptionPane.showMessageDialog(null,"查询 user 表失败");
            e.printStackTrace();
        }finally{
            //关闭创建的对象,释放资源
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
            }
        //返回用户信息
        return user;
    }
    //定义方法:完成修改密码的功能
    public boolean modifyPassword(String userId,String newpassword){
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
         //实例化数据库工具
        DBUtil dbUtil = new DBUtil();
        boolean flag = false;
        try{
            con = dbUtil.getConnection();
            String sql = "update user set password = ? where user_id = ?";
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, newpassword);
            pstmt.setString(2, userId);
```

```
            int n = pstmt.executeUpdate();
            if(n > 0)flag = true;
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 user 表失败!");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        return flag;
    }
    //定义方法:实现按照用户编号查找用户的功能
    public User selectUserById(String userId){
        User user = null;
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        DBUtil dbUtil = new DBUtil();
        try{
            con = dbUtil.getConnection();
            String sql = "select * from user where user_id = ?";
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, userId);
            rs = pstmt.executeQuery();
            if(rs.next())
                user = new User(rs.getString(1),rs.getString(2),rs.getString(3));
        }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 user 表失败!");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        return user;
    }
    //定义方法:实现添加新用户的功能
    public boolean insertUser(User user){
        boolean bool = false;
        Connection con  = null;
        PreparedStatement pstmt = null;
```

```
        ResultSet rs = null;
        DBUtil dbUtil = new DBUtil();
        try{
            con = dbUtil.getConnection();
            String sql = "insert into user values(?,?,?)";
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, user.getUserId());
            pstmt.setString(2, user.getPassword());
            pstmt.setString(3, user.getLevel());
            int n = pstmt.executeUpdate();
            if(n > 0)
                bool = true;
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 user 表失败!");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        return bool;
    }
}
```

(6) 修改 Login 类,在构造方法中采用匿名类的方式实现“登录”按钮的事件处理。

对“登录”按钮的事件处理过程:当单击“登录”按钮时,首先获取登录窗口中的账号、密码和级别信息,然后调用 UserDao 类中的 selectUserByIdAndPasswordAndLevel()方法查询数据库中的 user 表,最后根据该方法的返回值进行判断处理,如果返回值为 null,则提示用户“账号或密码或级别不正确”,如果返回值不为空,则打开系统的主窗口,同时隐藏登录窗口。

“登录”按钮事件处理的代码如下:

```
btnLogin.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e){
    //获取登录窗口中的账号、密码和级别信息
    String userId = txtLogin.getText();
    char[] passwordChars = txtPassword.getPassword();
    String password = new String(passwordChars);
    int index = level.getSelectedIndex();
    String userlevel = null;
    if(index == 0)
        userlevel = "学生";
    else if(index == 1)
        userlevel = "教师";
    else if(index == 2)
        userlevel = "管理员";
```

```
        UserDao ud = new UserDao();
        User user = ud.selectUserByIdAndPasswordAndLevel( userId,password, userlevel );
        if(user == null){
            JOptionPane.showMessageDialog(Login.this.btnLogin,"账号或密码或级别错误,
                                                    请重新输入");
            //清空账号和密码框中的内容
            txtLogin.setText("");
            txtPassword.setText("");
            //账号输入框得到焦点
            Login.this.txtLogin.requestFocus();
            return;
        }else{
            //创建主窗口对象
            MainFrame mf = new MainFrame(user);
            //主窗口显示
            mf.setVisible(true);
            //登录窗口隐藏
            Login.this.setVisible(false);
        }
      }
    });
```

除了“登录”按钮外，登录窗口中还有一个“取消”按钮。下面对“取消”按钮进行事件处理。

(7) 修改 Login 类，在构造方法中采用匿名类的方式实现“取消”按钮的事件处理。

对“取消”按钮的事件处理过程：清空账号和密码框的内容，同时账号输入框获得焦点。代码如下：

```
//"取消"按钮注册事件
btnCancel.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        //清空编号和密码内容
        txtLogin.setText("");
        txtPassword.setText("");
        //用户编号输入框得到焦点
        Login.this.txtLogin.requestFocus();
    }
});
```

当在图 3-2 所示的窗口中输入账号、密码以及选择用户级别(如管理员)后，单击“登录”按钮时，则利用 JDBC 技术访问 student 数据库中的 user 表，如果连接数据库出错，则弹出如图 3-23 所示的对话框；如果该用户存在，则跳转到如图 3-3 所示的主窗口，如果该用户不存在，则弹出如图 3-24 所示的“消息”对话框。如果单击如图 3-2 所示登录窗口中的“取消”按钮，则清空账号和密码框中的内容。


图 3-23　连接数据库失败的“消息”对话框


图 3-24　登录失败的“消息”对话框

## 二、“修改密码”菜单项功能实现

### 1. 实现思路

当单击“修改密码”菜单项时，首先应该调出如图 3-25 所示的“修改密码”窗口。在该窗口中输入当前用户的旧密码、新密码、确认密码等信息。


图 3-25 “修改密码”窗口

当单击该窗口中的“修改”按钮时，首先检查该窗口中的 3 个密码框是否都不为空，如果存在空的密码框，则利用如图 3-26 所示的“消息”对话框进行提示，要求用户重新输入旧密码或新密码或确认密码等信息；如果 3 个密码框都不为空，则接着判断当前用户输入的旧密码与 user 数据表中的密码是否一致，如果不一致，则利用如图 3-27 所示“消息”对话框进行提示，要求用户重新输入旧密码；如果与旧密码一致，最后还要判断用户输入的新密码和确认密码是否一致，如果二者不一致，则弹出如图 3-28 所示的“消息”对话框，要求用户重新输入新密码、确认密码等信息，如果新密码和确认密码一致，则调用 UserDao 类中的 modifyPassword()方法修改 user 数据表中当前用户的密码并返回一个 boolean 值。如果返回值为 true，则弹出如图 3-29 所示的“密码修改成功”消息提示框，否则弹出“密码修改失败”消息提示框。

当单击“取消”按钮时，清空 3 个密码框中的内容。


图 3-26 密码框为空时的消息提示对话框


图 3-27　旧密码不正确时的消息提示对话框


图 3-28　新密码和确认密码不一致时的消息提示对话框


图 3-29　密码修改成功时的消息提示对话框

**2. 代码实现**

(1) 定义修改密码的窗口类 ModifyPassword。

代码如下：

```
package view;
import java.awt.event.*;
import javax.swing.*;
import dao.UserDao;
import entity.User;
public class ModifyPassword extends JFrame{
    //定义组件
    private JPanel jp;
    private JLabel lblOldPassword,lblNewPassword,lblConfirmPassword;
    private JPasswordField txtOldPassword,txtNewPassword,txtConfirmPassword;
    private JButton btnModify,btnCancel;
    //定义构造方法:完成窗口的初始化,通过形参接收当前登录用户的信息,为修改
    //密码做好准备
    public ModifyPassword(final User user){
      super("修改密码");
      jp=new JPanel(null);
      lblOldPassword=new JLabel("请输入旧密码");
      lblNewPassword=new JLabel("请输入新密码");
      lblConfirmPassword=new JLabel("请确认新密码");
      txtOldPassword=new JPasswordField();
      txtNewPassword=new JPasswordField();
      txtConfirmPassword=new JPasswordField();
      btnModify=new JButton("修改");
      btnCancel=new JButton("取消");
      //设置各组件的绝对位置和大小
      lblOldPassword.setBounds(60, 40, 100, 15);
      lblNewPassword.setBounds(60, 90, 100, 15);
      lblConfirmPassword.setBounds(60, 140, 100, 15);
      txtOldPassword.setBounds(160, 36, 160, 21);
      txtNewPassword.setBounds(160, 86, 160, 21);
      txtConfirmPassword.setBounds(160, 136, 160, 21);
      btnModify.setBounds(140, 210, 60, 23);
      btnCancel.setBounds(210, 210, 60, 23);
```

```
    //将中间容器(面板)jp 添加到窗口中
    this.add(jp);
    //将各组件添加到中间容器 jp 中
    jp.add(lblOldPassword);
    jp.add(lblNewPassword);
    jp.add(lblConfirmPassword);
    jp.add(txtOldPassword);
    jp.add(txtNewPassword);
    jp.add(txtConfirmPassword);
    jp.add(btnModify);
    jp.add(btnCancel);
    this.setSize(420,350);
    setLocationRelativeTo(null);                    //窗口居中
    setResizable(false);                            //窗口大小不可改变
  }
}
```

(2) 修改 MainFrame 类的构造方法,添加如下代码,实现“修改密码”菜单项的事件处理。

```
modifyPassword.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
          ModifyPassword f = new ModifyPassword(user);
          f.setVisible(true);
          }
});
```

(3) 修改 ModifyPassword 类的构造方法,添加如下代码,实现“修改”按钮的事件处理。

```
btnModify.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ex){
        //获取输入的旧密码、新密码以及确认密码
        String oldPass = new String(txtOldPassword.getPassword());
        String pass1 = new String(txtNewPassword.getPassword());
        String pass2 = new String(txtConfirmPassword.getPassword());
          //判断旧密码框是否为空,如果为空,则提示"旧密码框不能为空,请输入!"
          if(oldPass.equals("")){
          JOptionPane.showMessageDialog(btnModify, "旧密码框不能为空,请输入!");
          txtOldPassword.requestFocus();
          return;
          }
    //判断新密码框是否为空,如果为空,则提示"新密码框不能为空,请输入!"
          if(pass1.equals("")){
          JOptionPane.showMessageDialog(btnModify, "新密码框不能为空,请输入!");
          txtNewPassword.requestFocus();
          return;
    }
    //判断确认密码框是否为空,如果为空,则提示"确认密码框不能为空,请输入!"
    if(pass2.equals("")){
          JOptionPane.showMessageDialog(btnModify, "确认密码框不能为空,请输入!");
          txtConfirmPassword.requestFocus();
          return;
    }
    //判断当前输入的旧密码是否正确
    if(!oldPass.equals(user.getPassword())){
        JOptionPane.showMessageDialog(btnModify,"旧密码不正确,请重新输入!");
        txtOldPassword.setText("");
```

```
            txtOldPassword.requestFocus();
            return;
        }
        //判断新密码与确认密码是否一致
        if(!pass1.equals(pass2)){
            JOptionPane.showMessageDialog(btnModify,"新密码与确认密码不相同,请重新输入!");
            txtNewPassword.setText("");
            txtConfirmPassword.setText("");
            txtNewPassword.requestFocus();
            return;
        }
        //访问 user 表,完成修改当前用户的密码的功能
        }
    });
```

这时如果 3 个密码框都符合要求,接下来就要访问 user 表,实现修改当前用户密码的功能。

(4) 修改 UserDao 类,添加修改密码的方法 modifyPassword( )。

代码如下:

```
public boolean modifyPassword(String userId, String newpassword){
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    //实例化数据库工具
    DBUtil dbUtil = new DBUtil();
    boolean flag = false;
    //打开数据库,获取连接对象
    try{
      con = dbUtil.getConnection();
      pstmt = con.prepareStatement("update user set password = ? where user_id = ?");
      //创建 PreparedStatement 对象
      pstmt.setString(1,newpassword);
      pstmt.setString(2,userId);
      int n = pstmt.executeUpdate();
      if(n > 0) flag = true;
  }catch(Exception e){
      JOptionPane.showMessageDialog(null,"访问 user 表失败!");
  }finally{
      try{
          if(rs!= null)
              rs.close();
          if(pstmt!= null)
              pstmt.close();
          if(con!= null)
          con.close();
      }catch(Exception e){
          e.printStackTrace();
      }
  }
return flag;
}
```

(5) 修改 ModifyPassword 类的构造方法,在“修改”按钮事件处理的代码段中的注释“//访问 user 表,完成修改当前用户的密码的功能”处调用 modifyPassword()方法,实现修

改密码的功能。

添加代码如下：

```
try{
    UserDao ud = new UserDao();
    boolean flag = ud.modifyPassword(user.getUserId(),pass1);
    if(flag)
        OptionPane.showMessageDialog(btnModify, "密码修改成功");
    else JOptionPane.showMessageDialog(btnModify, "密码修改失败");
} catch (Exception e) {
    e.printStackTrace();
}
```

至此，修改密码的功能就成功实现了。

另外，在修改密码的对话框中还有一个“取消”按钮，下面快速实现“取消”按钮的功能。

(6) 修改 ModifyPassword 类的构造方法，添加如下代码，实现“取消”按钮的事件处理。

```
btnCancel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        txtOldPassword.setText("");
        txtNewPassword.setText("");
        txtConfirmPassword.setText("");
    }
});
```

运行该系统，单击“修改密码”菜单项，发现该菜单项能正确完成当前用户修改密码的功能。

### 三、“查看个人信息”菜单项功能实现

#### 1. 实现思路

由于本系统具有 3 类用户，不同身份的用户他们的个人信息是不同的，那么在实现该菜单项的功能时就要针对不同的用户展示不同的查看结果。当学生身份的用户单击“查看个人信息”菜单项时，应该调出如图 3-30 所示的窗口，在该窗口中显示出当前学生的个人信息，并且这些信息不允许用户修改。当教师或管理员（管理员也是一名教师，其信息也存储在 teacher 表中）身份的用户单击“查看个人信息”菜单项时，应该调出如图 3-31 所示的窗口，在该窗口中显示出当前教师的个人信息，并且这些信息不允许用户修改。


图 3-30　学生个人信息


图 3-31　教师(管理员)个人信息

**2. 代码实现**

下面先创建显示学生信息的窗口。

(1) 定义显示学生信息的窗口。

代码如下：

```
public class DisplayStuSelfInfo extends JFrame {
  private JPanel jp;
  private JLabel lblSId,lblSName,lblSex,lblDepartment,lblClass,lblBirthday,
                 lblTel,lblAddress;
  private JTextField txtSId,txtSName,txtSex,txtDepartment,txtClassName,txtBirthday,
                 txtTel,txtAddress;
  public DisplayStuSelfInfo(User user){
      super("个人信息");
      jp = new JPanel(null);
      lblSId = new JLabel("学号:");
      lblSName = new JLabel("姓名:");
      lblSex = new JLabel("性别:");
      lblDepartment = new JLabel("院系:");
      lblClass = new JLabel("班级:");
      lblBirthday = new JLabel("出生日期:");
      lblTel = new JLabel("联系电话:");
      lblAddress = new JLabel("家庭住址:");
      txtSId = new JTextField(10);
      txtSName = new JTextField(10);
      txtSex = new JTextField(10);
      txtDepartment = new JTextField(10);
      txtClassName = new JTextField(10);
      txtBirthday = new JTextField(10);
      txtTel = new JTextField(12);
      txtAddress = new JTextField(20);
      this.add(jp);
      jp.add(lblSId);
      jp.add(txtSId);
      jp.add(lblSName);
      jp.add(txtSName);
```

```
        jp.add(lblSex);
        jp.add(txtSex);
        jp.add(lblBirthday);
        jp.add(txtBirthday);
        jp.add(lblDepartment);
        jp.add(txtDepartment);
        jp.add(lblClass);
        jp.add(txtClassName);
        jp.add(lblTel);
        jp.add(txtTel);
        jp.add(lblAddress);
        jp.add(txtAddress);
        lblSId.setBounds(50,20,100,15);
        lblSName.setBounds(50, 50, 100, 15);
        lblSex.setBounds(50, 80, 100, 15);
        lblBirthday.setBounds(50, 110, 100, 15);
        lblDepartment.setBounds(50, 140, 100, 15);
        lblClass.setBounds(50, 170, 100, 15);
        lblTel.setBounds(50, 200, 100, 15);
        lblAddress.setBounds(50, 230, 100, 15);
        txtSId.setBounds(150, 20, 180, 21);
        txtSName.setBounds(150, 50, 180, 21);
        txtSex.setBounds(150, 80, 180, 21);
        txtBirthday.setBounds(150, 110, 180, 21);
        txtDepartment.setBounds(150, 140, 180, 21);
        txtClassName.setBounds(150, 170, 180, 21);
        txtTel.setBounds(150, 200, 180, 21);
        txtAddress.setBounds(150, 230, 180, 21);
        this.setSize(420, 350);
        setLocationRelativeTo(null);              //Frame 居中
        setResizable(false);                      //禁止改变框架大小
    }
    }
```

此时上述窗口中没有显示当前用户的信息。要想得到当前学生用户的信息，需要访问数据表 student。与 user 表的处理方法一样，这里也要对 student 表定义相应的实体类和访问业务逻辑类。

(2) 在 entity 包中定义实体类 Student。

```
public class Student {
  //定义与 student 表中的字段相对应的属性变量
  private String studentId;
  private String studentName;
  private String studentSex;
  private Date studentBirthday;
  private String classId;
  private String studentTel;
  private String studentAddress;
  //定义各属性的 getXxx()、setXxx()方法
  public String getStudentId() {
      return studentId;
  }
  public void setStudentId(String studentId) {
      this.studentId = studentId;
```

```
    }
    public String getStudentName() {
        return studentName;
    }
    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }
    public String getStudentSex() {
        return studentSex;
    }
    public void setStudentSex(String studentSex) {
        this.studentSex = studentSex;
    }
    public Date getStudentBirthday() {
        return studentBirthday;
    }
    public void setStudentBirthday(Date studentBirthday) {
        this.studentBirthday = studentBirthday;
    }
    public String getClassId() {
        return classId;
    }
    public void setClassId(String classId) {
        this.classId = classId;
    }
    public String getStudentTel() {
        return studentTel;
    }
    public void setStudentTel(String studentTel) {
        this.studentTel = studentTel;
    }
    public String getStudentAddress() {
        return studentAddress;
    }
    public void setStudentAddress(String studentAddress) {
        this.studentAddress = studentAddress;
    }
    //定义构造方法
    public Student(String studentId, String studentName, String studentSex,
           Date studentBirthday, String classId, String studentTel,
           String studentAddress) {
           this.studentId = studentId;
           this.studentName = studentName;
           this.studentSex = studentSex;
           this.studentBirthday = studentBirthday;
           this.classId = classId;
           this.studentTel = studentTel;
           this.studentAddress = studentAddress;
      }
      public Student() {
           }
}
```

（3）在 dao 包中定义访问 student 表的类 StudentDao，在该类中定义方法 selectBySId()，实现通过学号查询学生信息的功能。

```
public class StudentDao {
    //通过学生学号查询得到学生的信息
    public Student selectBySId(String sId){
      Student student = null;
      Connection con = null;
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      //实例化数据库工具
      DBUtil dbUtil = new DBUtil();
      try{
      //打开数据库,获取连接对象
      con = dbUtil.getConnection();
      //创建 PreparedStatment 对象
      pstmt = con.prepareStatement("select * from student where student_id = ? ");
      pstmt.setString(1, sId);
      rs = pstmt.executeQuery();                    //获取查询结果集
      //如果访问结果集中有数据,则将这些数据实例化为一个 student 对象并返回
      if(rs.next())
          student = new Student(rs.getString(1),rs.getString(2),rs.getString(3),rs.getDate(4),
                    rs.getString(5),rs.getString(6),rs.getString(7));
      }catch(Exception e){
              JOptionPane.showMessageDialog(null, "访问 user 表失败!");
      }finally{
            try{
              if(rs!= null)
                  rs.close();
              if(pstmt!= null)
                  pstmt.close();
              if(con!= null)
                  con.close();
          }catch( Exception e){
              e.printStackTrace( );
          }
      }
    return student ;
} }
```

这时通过该方法就可以得到当前学生用户的信息，得到的信息包括学号、姓名、性别、出生日期、班级编号、联系电话和家庭住址。但是在需求中要求显示的班级信息不是班级编号，而是班级名称，下面就访问 sclass 表，定义一个方法实现通过班级编号获取班级名称的功能。

（4）在 entity 包中定义实体类 SClass。

```
public class SClass {
  //定义与 sclass 表中的字段相对应的属性变量
  private String classId;
  private String className;
  private String departmentId;
  private String assitantId;
  //定义各属性的 getXxx()、setXxx()方法
  public String getClassId() {
```

```
    return classId;
  }
  public void setClassId(String classId) {
    this.classId = classId;
  }
  public String getClassName() {
    return className;
  }
  public void setClassName(String className) {
    this.className = className;
  }
  public String getDepartmentId() {
    return departmentId;
  }
  public void setDepartmentId(String departmentId) {
    this.departmentId = departmentId;
  }
  public String getAssitantId() {
    return assitantId;
  }
  public void setAssitantId(String assitantId) {
    this.assitantId = assitantId;
  }
  //定义构造方法
  public SClass(String classId, String className, String departmentId, String assitantId) {
    this.classId = classId;
    this.className = className;
    this.departmentId = departmentId;
    this.assitantId = assitantId;
  }
  public SClass() {

  }
}
```

(5) 在 dao 包中定义访问 sclass 表的类 SClassDao,在该类中定义方法 selectByClassId(),实现通过班级编号获取班级信息的功能。

```
public class SClassDao {
//通过班级编号获取班级信息
public SClass selectByClassId(String classId){
    SClass c = null;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    //实例化数据库工具
    DBUtil dbUtil = new DBUtil();
    try{
        //打开数据库,获取连接对象
        con = dbUtil.getConnection();
        //创建 PreparedStatment 对象
        pstmt = con.prepareStatement("select * from sclass where class_id = ?");
        pstmt.setString(1, classId);
        rs = pstmt.executeQuery();              //获取查询结果集
```

```
            //如果访问结果集中有数据,则用这些数据实例化 c 对象并返回
            if(rs.next())
                c = new SClass(rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4));
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 sclass 表失败");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
                }catch(Exception e){
                    e.printStackTrace();
                }
        }
        return c;
        }
}
```

在这里通过上述方法,就可以查询到指定班级编号的班级信息了。信息包括班级编号、班级名称、院系编号以及辅导员编号。

回过头来再查看一下需求,发现需求中还需要显示出该学生用户所在的院系名称,接下来还要进一步访问 department 表,实现通过院系编号获取到院系名称的功能。

(6) 在 entity 包定义实体类 Department 类。

```
public class Department {
    private String departmentName;
    private String departmentId;
    public String getDepartmentName() {
      return departmentName;
   }
      public void setDepartmentName(String departmentName) {
       this.departmentName = departmentName;
      }
    public String getDepartmentId() {
      return departmentId;
    }
    public void setDepartmentId(String departmentId) {
      this.departmentId = departmentId;
    }
    public Department(String departmentName, String departmentId) {
      this.departmentName = departmentName;
      this.departmentId = departmentId;
  }
  public Department() {
  }
}
```

(7) 在 dao 包中定义访问 department 表的类 DepartmentDao,在该类中定义方法 selectByDepartmentId(),实现通过院系编号获取院系名称的功能。

```
public class DepartmentDao {
  //通过院系编号获取院系名称
```

```
    public String selectByDepartmentId(String departmentId){
      String departmentName = null;
      Connection con = null;
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      DBUtil dbUtil = new DBUtil();
      try{
          con = dbUtil.getConnection();
          pstmt = con.prepareStatement("select * from department where department_id = ?");
          pstmt.setString(1, departmentId);
          rs = pstmt.executeQuery();
          if(rs.next())
            departmentName = rs.getString(2);
          }catch(Exception e){
              JOptionPane.showMessageDialog(null, "访问 department 表失败");
          }finally{
              try{
                  if(rs!= null)
                      rs.close();
                  if(pstmt!= null)
                      pstmt.close();
                  if(con!= null)
                      con.close();
                  }catch(Exception e){
                      e.printStackTrace();
                  }
          }
          return departmentName;
      }
  }
```

至此，确定好了通过学号获取学生信息、通过班级编号获取班级信息以及通过院系编号获取院系名称的功能。接下来就调用这些方法来获取学生的完整信息，并显示在相应的文本框中。

（8）修改 DisplayStuSelfInfo 类，在构造方法中添加如下代码，实现获取学生的完整信息并显示在相应的文本框中的功能。

```
//访问 student、sclass 和 department 表，得到当前学生的信息、班级信息以及院系名称
StudentDao sd = new StudentDao();
Student s = sd.selectBySId(user.getUserId());
SClassDao cd = new SClassDao();
SClass c = cd.selectByClassId(s.getClassId());
DepartmentDao dd = new DepartmentDao();
String departmentName = dd.selectByDepartmentId(c.getDepartmentId());
//将学生信息在对应文本框中显示出来
txtSId.setText(s.getStudentId());
txtSName.setText(s.getStudentName());
txtSex.setText(s.getStudentSex());
txtClassName.setText(c.getClassName());
txtDepartment.setText(departmentName);
txtTel.setText(s.getStudentTel());
txtAddress.setText(s.getStudentAddress());
Date d = s.getStudentBirthday();
```

```
//将 Date 类型的生日转换为 String 类型的字符串,并显示在相应的文本框中
txtBirthday.setText(String.valueOf(d));
```

(9) 将文本框内容设置为不可编辑状态。

```
txtSId.setEditable(false);
txtSName.setEditable(false);
txtSex.setEditable(false);
txtBirthday.setEditable(false);
txtDepartment.setEditable(false);
txtClassName.setEditable(false);
txtAddress.setEditable(false);
txtTel.setEditable(false);
```

此时,学生用户查看个人信息的功能就开发完成了。

下面,仿照学生用户查看个人信息的实现过程,快速实现教师或管理员查看个人信息的功能。这里,为了管理方便,把管理员的信息也存储在教师表中。

(10) 定义显示教师或管理员信息的窗口。

```
public class DisplayTeacherSelfInfo extends JFrame {
private JPanel jp;
private JLabel lblSId,lblSName,lblSex,lblDepartment,lblBirthday,lblPosition,
lblTel,lblAddress;
private JTextField txtSId,txtSName,txtSex,txtDepartment,txtBirthday,txtPosition,
txtTel,txtAddress;
public DisplayTeacherSelfInfo(User user){
    super("个人信息");
    jp = new JPanel(null);
    lblSId = new JLabel("编号:");
    lblSName = new JLabel("姓名:");
    lblSex = new JLabel("性别:");
    lblDepartment = new JLabel("院系:");
    lblBirthday = new JLabel("出生日期:");
    lblPosition = new JLabel("职称:");
    lblTel = new JLabel("联系电话:");
    lblAddress = new JLabel("家庭住址:");
    txtSId = new JTextField(10);
    txtSName = new JTextField(10);
    txtSex = new JTextField(10);
    txtDepartment = new JTextField(10);
    txtBirthday = new JTextField(10);
    txtPosition = new JTextField(10);
    txtTel = new JTextField(12);
    txtAddress = new JTextField(20);
    this.add(jp);
    jp.add(lblSId);
    jp.add(txtSId);
    jp.add(lblSName);
    jp.add(txtSName);
    jp.add(lblSex);
    jp.add(txtSex);
    jp.add(lblBirthday);
    jp.add(txtBirthday);
    jp.add(lblDepartment);
```

```
        jp.add(txtDepartment);
        jp.add(lblPosition);
        jp.add(txtPosition);
        jp.add(lblTel);
        jp.add(txtTel);
        jp.add(lblAddress);
        jp.add(txtAddress);
        lblSId.setBounds(50,20,100,15);
        lblSName.setBounds(50, 50, 100, 15);
        lblSex.setBounds(50, 80, 100, 15);
        lblBirthday.setBounds(50, 110, 100, 15);
        lblDepartment.setBounds(50, 140, 100, 15);
        lblPosition.setBounds(50, 170, 100, 15);
        lblTel.setBounds(50, 200, 100, 15);
        lblAddress.setBounds(50, 230, 100, 15);
        txtSId.setBounds(150, 20, 180, 21);
        txtSName.setBounds(150, 50, 180, 21);
        txtSex.setBounds(150, 80, 180, 21);
        txtBirthday.setBounds(150, 110, 180, 21);
        txtDepartment.setBounds(150, 140, 180, 21);
        txtPosition.setBounds(150, 170, 180, 21);
        txtTel.setBounds(150, 200, 180, 21);
        txtAddress.setBounds(150, 230, 180, 21);
        //将文本框内容设置为不可编辑状态
        txtSId.setEditable(false);
        txtSName.setEditable(false);
        txtSex.setEditable(false);
        txtBirthday.setEditable(false);
        txtDepartment.setEditable(false);
        txtPosition.setEditable(false);
        txtAddress.setEditable(false);
        txtTel.setEditable(false);
        this.setSize(420, 350);
        this.setLocationRelativeTo(null);
        setResizable(false);
        }
}
```

(11) 在 entity 包中定义实体类 Teacher 类。

```
public class Teacher {
private String teacherId;
private String teacherName;
private String teacherSex;
private Date teacherBirthday;
private String departmentId;
private String position;
private String teacherTel;
private String teacherAddress;
public String getTeacherId() {
    return teacherId;
  }
  public void setTeacherId(String teacherId) {
    this.teacherId = teacherId;
  }
```

```
public String getTeacherName() {
  return teacherName;
}
public void setTeacherName(String teacherName) {
  this.teacherName = teacherName;
}
public String getTeacherSex() {
  return teacherSex;
}
public void setTeacherSex(String teacherSex) {
  this.teacherSex = teacherSex;
}
public Date getTeacherBirthday() {
  return teacherBirthday;
}
public void setTeacherBirthday(Date teacherBirthday) {
  this.teacherBirthday = teacherBirthday;
}
public String getDepartmentId() {
  return departmentId;
}
public void setDepartmentId(String departmentId) {
  this.departmentId = departmentId;
}
public String getPosition() {
  return position;
}
public void setPosition(String position) {
  this.position = position;
}
public String getTeacherTel() {
  return teacherTel;
}
public void setTeacherTel(String teacherTel) {
  this.teacherTel = teacherTel;
}
public String getTeacherAddress() {
  return teacherAddress;
}
public void setTeacherAddress(String teacherAddress) {
  this.teacherAddress = teacherAddress;
}
public Teacher(String teacherId, String teacherName, String teacherSex, Date
               teacherBirthday, String departmentId,String position,String teacherTel,
               String teacherAddress) {
  super();
  this.teacherId = teacherId;
  this.teacherName = teacherName;
  this.teacherSex = teacherSex;
  this.teacherBirthday = teacherBirthday;
  this.departmentId = departmentId;
  this.position = position;
  this.teacherTel = teacherTel;
  this.teacherAddress = teacherAddress;
```

```
    }
    public Teacher() {
    }
}
```



（12）在 dao 包中定义访问 teacher 表的类 TeacherDao，在该类中定义方法 selectByTId()，实现通过教师编号得到教师信息的功能。

```
public class TeacherDao {
//通过教师编号得到教师的信息
public Teacher selectByTId(String tId){
  Teacher teacher = null;
  Connection con = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;
  DBUtil dbUtil = new DBUtil();
  try{
    con = dbUtil.getConnection();
    pstmt = con.prepareStatement("select * from teacher where teacher_id = ?");
    pstmt.setString(1, tId);
    rs = pstmt.executeQuery();
    if(rs.next())
      teacher = new Teacher(rs.getString(1),rs.getString(2),rs.getString(3),rs.getDate(4),
                  rs.getString(5),rs.getString(6),rs.getString(7),rs.getString(8));
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 teacher 表失败!");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
        }
        return teacher;
}
```

（13）修改 DisplayTeacherSelfInfo 类，在构造方法中添加如下代码，获取教师的完整信息，并显示在相应的文本框中。

```
TeacherDao td = new TeacherDao();
Teacher t = td.selectByTId(user.getUserId());
DepartmentDao dd = new DepartmentDao();
String departmentName = dd.selectByDepartmentId(t.getDepartmentId());
txtSId.setText(t.getTeacherId());
txtSName.setText(t.getTeacherName());
txtSex.setText(t.getTeacherSex());
txtPosition.setText(t.getPosition());
txtDepartment.setText(departmentName);
txtTel.setText(t.getTeacherTel());
txtAddress.setText(t.getTeacherAddress());
```

```
Date d = t.getTeacherBirthday();
txtBirthday.setText(String.valueOf(d));
```

（14）修改 MainFrame 类的构造方法，添加如下代码，实现不同身份的用户“查看个人信息”菜单项的事件处理。

```
displaySelfInfo.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent arg0) {
      if(user.getLevel().equals("学生")){
          DisplayStuSelfInfo d = new DisplayStuSelfInfo(user);
          d.setVisible(true);
      }else{
          DisplayTeacherSelfInfo t = new DisplayTeacherSelfInfo(user);
          t.setVisible(true);
      }
    }
});
```

## 四、“查询学生信息”菜单项功能实现

### 1. 实现思路

当教师或管理员单击“查询学生信息”菜单项时，首先应该调出如图 3-32 所示的“查询学生信息”窗口，在该窗口中默认显示所有学生的信息。当选择“按学号查询学生”单选按钮时，窗口变为如图 3-33 所示，在该窗口中输入学号并单击“查询”按钮时，会完成按学号查询的功能并将查询结果显示在窗口中；当选择“按班级查询学生”单选按钮时，窗口变为如图 3-34 所示，在该窗口中选择院系和班级并单击“查询”按钮时，会完成按班级查询的功能并将查询结果显示在窗口中；当选择“按院系查询学生”单选按钮时，窗口变为如图 3-35 所示，在该窗口中选择院系并单击“查询”按钮时，会完成按院系查询的功能并将查询结果显示在窗口中。


图 3-32 “查询学生信息”窗口


图 3-33　按学号查询学生信息

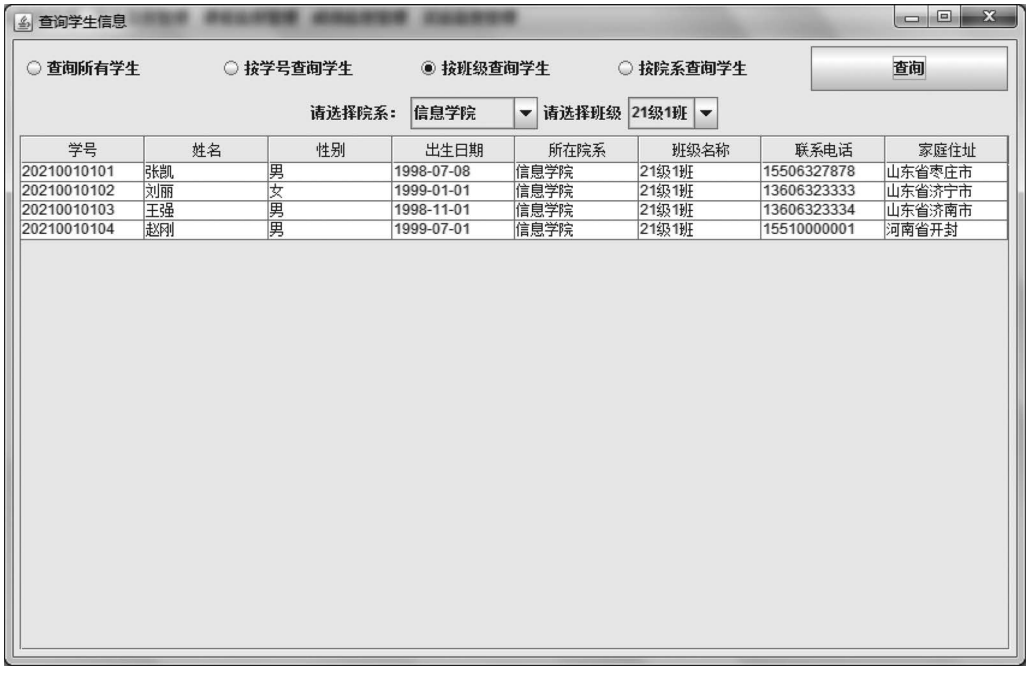
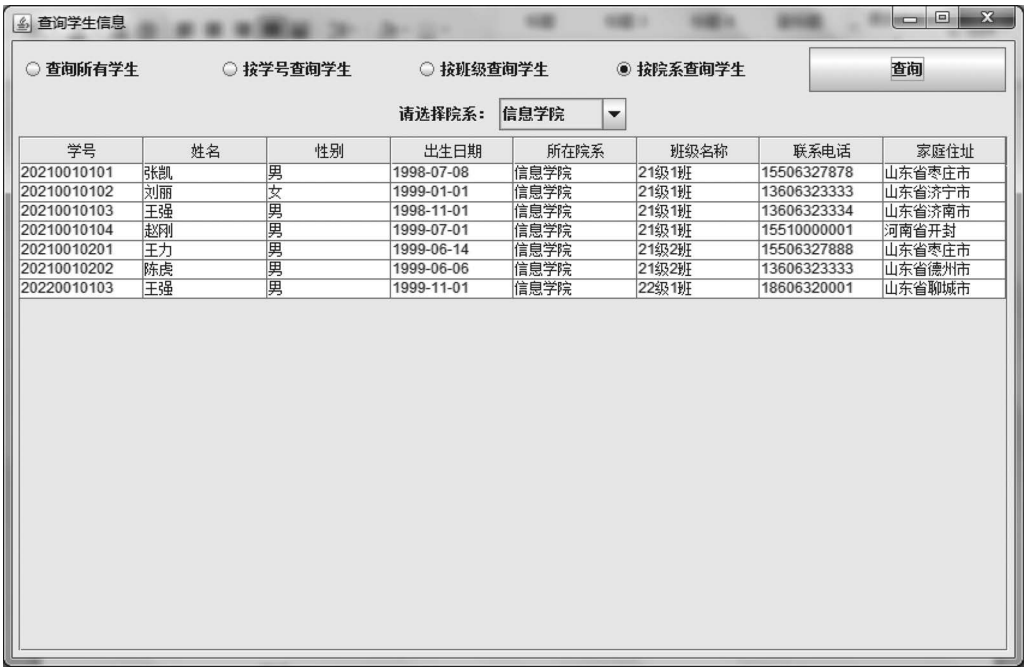



图 3-34　按班级查询学生信息




| 学号 | 姓名 | 性别 | 出生日期 | 所在院系 | 班级名称 | 联系电话 | 家庭住址 |
|---|---|---|---|---|---|---|---|
| 20210010101 | 张凯 | 男 | 1998-07-08 | 信息学院 | 21级1班 | 15506327878 | 山东省枣庄市 |
| 20210010102 | 刘丽 | 女 | 1999-01-01 | 信息学院 | 21级1班 | 13606323333 | 山东省济宁市 |
| 20210010103 | 王强 | 男 | 1998-11-01 | 信息学院 | 21级1班 | 13606323334 | 山东省济南市 |
| 20210010104 | 赵刚 | 男 | 1999-07-01 | 信息学院 | 21级1班 | 15510000001 | 河南省开封 |
| 20210010201 | 王力 | 男 | 1999-06-14 | 信息学院 | 21级2班 | 15506327888 | 山东省枣庄市 |
| 20210010202 | 陈虎 | 男 | 1999-06-06 | 信息学院 | 21级2班 | 13606323333 | 山东省德州市 |
| 20220010103 | 王强 | 男 | 1999-11-01 | 信息学院 | 22级1班 | 18606320001 | 山东省聊城市 |

图 3-35　按院系查询学生信息

**2. 代码实现**

通过上述分析不难发现，要想完成上述查询功能需要访问 student 表、sclass 表以及 department 表。下面一一加以实现。

(1) 实现查询所有学生信息的功能。

因为不需要查询条件，所以这里只需要修改 StudentDao 类，添加 getAllStudents()方法，实现查询所有学生信息的功能即可。

代码如下：

```
public ArrayList < Student > getAllStudents(){
    ArrayList < Student > students = new ArrayList < Student >();
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    //实例化数据库工具
    DBUtil dbUtil = new DBUtil();
    Student student = null;
    try{
        //打开数据库,获取连接对象
        con = dbUtil.getConnection();
        //创建 prepareStatement 对象
        pstmt = con.prepareStatement("select * from student");
        rs = pstmt.executeQuery();                    //获取查询结果
        //将查询结果中的每一行封装成一个 student 对象,并添加到 students 集合中
        while(rs.next()){
            student = new Student(rs.getString(1),rs.getString(2),rs.getString(3),
                rs.getDate(4),rs.getString(5),rs.getString(6),rs.getString(7));
            students.add(student);
```

```
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 student 表失败!");
        }finally{
            try{
                    if(rs!= null)
                        rs.close();
                    if(pstmt!= null)
                        pstmt.close();
                    if(con!= null)
                        con.close();
                }catch( Exception e){
                    e.printStackTrace( );
                    }
        }
        return students;
    }
```

(2) 实现按学号查询学生信息的功能。

分析：这里先要获取学生编号，然后通过学号查询学生信息即可。前面已经在StudentDao类中定义过按学号查询学生信息的方法selectBySId()，在此就不多做解释了。

(3) 实现按班级查询学生信息的功能。

分析：在这里采用院系与班级联动的形式选择班级。

① 通过院系名称获取院系编号。

```
public String selectByDepartmentName(String departmentName){
    String departmetId = null;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from department where department_name
                                    = ?");
        pstmt.setString(1, departmentName);
        rs = pstmt.executeQuery();
        if(rs.next())
            departmetId = rs.getString(1);
    }catch(Exception e){
         JOptionPane.showMessageDialog(null, "访问 department 表失败");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
    }
```

```
    return departmetId;
}
```

② 实现通过班级的名称以及所在院系获得班级编号的功能。

```
public String selectByClassNameAndDepartment(String className,String department){
    String classId = null;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DepartmentDao dd = new DepartmentDao();
    String departmentId = dd.selectByDepartmentName(department);
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from sclass where class_name = ? and department_
id = ?");
        pstmt.setString(1, className);
        pstmt.setString(2, departmentId);
        rs = pstmt.executeQuery();
        if(rs.next())
            classId = rs.getString(1);
    }catch(Exception e){
        JOptionPane.showMessageDialog(null,"访问 sclass 表失败!");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
        }catch(Exception e){
            e.printStackTrace();
        }
  }
    return classId;
}
```

③ 修改 StudentDao 类，添加 getStudentsByClassId()方法，实现按照班级编号查询学生信息的功能。

```
public ArrayList < Student > getStudentsByClassId(String classId){
    ArrayList < Student > students = new ArrayList < Student >();
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    Student student = null;
    try{
     con = dbUtil.getConnection();
     pstmt = con.prepareStatement("select * from student where class_id = ?");
     pstmt.setString(1, classId);
     rs = pstmt.executeQuery();
     //将查询结果封装到 students 集合中
```

```
        while(rs.next()){
            student = new Student(rs.getString(1),rs.getString(2),rs.getString(3),rs.getDate(4),
                        rs.getString(5),rs.getString(6),rs.getString(7));
            students.add(student);
        }
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 student 表失败!");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
                }catch( Exception e){
                    e.printStackTrace( );
                }
    }
    return students;
}
```

当选择按照班级查询学生信息时，需要在“院系”组合框和“班级”组合框中选择院系和班级名称。为了与数据库中的信息一致，这里应该先查询 department 表，获取所有的院系名称并添加到“院系”组合框中，然后当选择其中的一个院系时，再根据该院系的编号查询 sclass 表，获取该院系的所有班级名称并动态地添加到“班级”组合框中，实现院系和班级之间的联动效果。

(4) 实现院系和班级之间的联动效果。

① 修改 DepartmentDao 类，添加获取所有院系名称的方法 getAllDepartmentName()。

```
public ArrayList < String > getAllDepartmentName(){
        ArrayList < String > allDepartmentNames = new ArrayList < String >();
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        DBUtil dbUtil = new DBUtil();
        try{
            con = dbUtil.getConnection();
            pstmt = con.prepareStatement("select department_name from department");
            rs = pstmt.executeQuery();
            while(rs.next())
                allDepartmentNames.add(rs.getString(1));
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 department 表失败");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
```

```
                }catch(Exception e){
                    e.printStackTrace();
                    }
        }
        return allDepartmentNames;
    }
```

② 修改 SClassDao 类，添加根据院系编号得到该院系所有班级名称的方法 selectClassNamesByDepartmentId()。

```
public ArrayList<String> selectClassNamesByDepartmentId(String departmentId){
    ArrayList<String> classNames = new ArrayList<String>();
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
      con = dbUtil.getConnection();
      pstmt = con.prepareStatement("select class_name from sclass where department_id = ?");
      pstmt.setString(1, departmentId);
      rs = pstmt.executeQuery();
      //将查询结果封装到 classNames 集合中
      while(rs.next()){
         classNames.add(rs.getString(1));
      }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 sclass 表失败");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
    }
    return classNames;
}
```

③ 定义监听类，实现对“院系”组合框选项改变时的监听。

因为在以后的功能中还要用到院系和班级之间的联动，所以这里将监听类定义在 util 包中以供其他模块使用。

```
public class DepartmentAndClassLinked implements ItemListener{
  JComboBox cmbDepartment,cmbClass;
  //构造方法
  public DepartmentAndClassLinked(JComboBox cmbDepartment,JComboBox cmbClass){
      this.cmbDepartment = cmbDepartment;
      this.cmbClass = cmbClass;
  }
  public void itemStateChanged(ItemEvent e) {
```

```
        DepartmentDao dd = new DepartmentDao();
        SClassDao sd = new SClassDao();
        String departmentId = null;
        String sclassId = null;
        //获取用户选中的院系名称
        String departmentName = (String)cmbDepartment.getSelectedItem();
        //查询 department 表,根据院系名称获取院系编号
        departmentId = dd.selectByDepartmentName(departmentName);
        //查询 sclass 表,根据院系编号获取该院系的所有班级名称
        ArrayList<String> allClassNames = sd.selectClassNamesByDepartmentId(departmentId);
        //清空"班级"组合框中的选项
        cmbClass.removeAllItems();
        //将查询到的班级名称添加到"班级"组合框中
        for (String className : allClassNames) {
        cmbClass.addItem(className);
      }
    }
    }
```

(5) 实现按院系查询的功能。

分析：该选项要求能够实现按照院系名称查询学生信息的功能。我们知道，student 表中没有院系名称，只有班级编号，而 sclass 表中有班级编号和院系编号，但是没有院系名称，department 表中有院系名称和院系编号。所以，要完成该功能，需要对 student、sclass、department 3 个数据表进行联合查询：首先要通过院系名称查询 department 表获取院系编号，然后通过院系编号查询 sclass 表获取该院系所有班级的编号，最后通过班级编号查询 student 表获取学生的信息。

上面已经在 DepartmentDao 类中定义过 selectByDepartmentName()方法，实现了通过院系名称获取院系编号的功能，下面接着实现通过院系编号查询 sclass 表获取该院系所有班级的编号的功能。

① 修改 SClassDao 类，添加 selectByDepartmentId()方法，实现根据院系编号得到该院系所有班级编号的集合的功能。

```
//定义方法:根据院系编号得到该院系所有班级编号的集合的功能
public ArrayList<String> selectByDepartmentId(String departmentId){
    ArrayList<String> classIds = new ArrayList<String>();
    String classId = null;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from sclass where department_id = ?");
        pstmt.setString(1, departmentId);
        rs = pstmt.executeQuery();
        //将查询结果封装到 classIds 集合中
        while(rs.next()){
            classId = rs.getString(1);
            classIds.add(classId);
        }
```

```
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 sclass 表失败");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch(Exception e){
                e.printStackTrace();
                }
    }
    return classIds;
}
```

② 修改 StudentDao 类，添加方法 getStudentsByDepartment()，实现按照院系名称查询学生信息的功能。

```
//按照院系名称查询学生信息的功能
 public ArrayList < Student > getStudentsByDepartment(String departmentName){
   ArrayList < Student > students = new ArrayList < Student >();
   SClassDao cd = new SClassDao();
   DepartmentDao dd = new DepartmentDao();
   //通过院系名称获取院系编号
   String departmentId = dd.selectByDepartmentName(departmentName);
   //通过院系编号获取该院系的所有班级编号
   ArrayList < String > classIds = cd.selectByDepartmentId(departmentId);
   //通过班级编号获取该院系的所有学生信息
  if(classIds!= null){
    for(String classId:classIds){
       ArrayList < Student > s = getStudentsByClassId(classId);
       students.addAll(s);
    }
   }
   return students;
 }
```

前期的功能模块实现后，下面开始创建显示查询到的学生信息的窗口，实现按不同的要求查询学生信息并显示到查询窗口中的功能。

(6) 定义显示查询学生信息的窗口。

代码如下：

```
public class SelectStudent extends JFrame {
  //定义所需的组件变量
  private JPanel jp1,jp11,jp12,jp2;
  //单选按钮
  private JRadioButton selectById,selectByClass,selectByDepartment,selectAll;
  //按钮组
  private ButtonGroup bg;
  private JLabel lblSId,lblClass,lblDepartment;
  private JTextField txtSId;
  //"班级""院系"组合框
```

```
private JComboBox cmbClassName,cmbDepartmentName;
private DefaultTableModel model;                    //表格模式
private JTable table;                               //表格
private JScrollPane sp;                             //滚动面板
private JButton ok;
ArrayList < Student > students = null;              //存放查询到的学生的集合
StudentDao sd = new StudentDao();
SClassDao cd = new SClassDao();
DepartmentDao dd = new DepartmentDao();
Object content[ ][ ] = null;
String title[ ] = {"学号","姓名","性别","出生日期","所在院系","班级名称","联系电话",
            "家庭住址"};
//构造方法
public SelectStudent(){
    setTitle("查询学生信息");
    jp1 = new JPanel();
    jp1.setLayout(new GridLayout(2,1));
    jp11 = new JPanel(new GridLayout(1,5));
    jp12 = new JPanel();
    jp2 = new JPanel();
    jp2.setBorder(new EmptyBorder(5,5,5,5));
    jp2.setLayout(new BorderLayout(0,0));
    bg = new ButtonGroup();
    selectAll = new JRadioButton("查询所有学生",true);
    selectById = new JRadioButton("按学号查询学生");
    selectByClass = new JRadioButton("按班级查询学生");
    selectByDepartment = new JRadioButton("按院系查询学生");
    bg.add(selectAll);
    bg.add(selectById);
    bg.add(selectByClass);
    bg.add(selectByDepartment);
    ok = new JButton("查询");
    jp11.add(selectAll);
    jp11.add(selectById);
    jp11.add(selectByClass);
    jp11.add(selectByDepartment);
    jp11.add(ok);
    jp1.add(jp11);
    lblSId = new JLabel("请输入学号:");
    txtSId = new JTextField(20);
    lblDepartment = new JLabel("请选择院系:");
    lblClass = new JLabel("请选择班级");
    //访问 department 表,获取所有院系名称
    DepartmentDao dd = new DepartmentDao();
    ArrayList < String > allDepartmentNames = dd.getAllDepartmentName();
    //创建"院系"组合框对象
    cmbDepartmentName = new JComboBox();
    //初始化"院系"组合框:将查询到的院系名称添加到"院系"组合框中
    for (String name : allDepartmentNames) {
        cmbDepartmentName.addItem(name);
    }
    //创建"班级"组合框对象
    cmbClassName = new JComboBox();
    //初始化"班级"组合框:根据"院系"组合框中的第一个院系名称查询 sclass 表,获取该院系的
```

```
        //所有班级名称并添加到"班级"组合框中
        //获取用户选中的院系名称
        String departmentName = (String)cmbDepartmentName.getItemAt(0);
        //查询 department 表,根据院系名称获取院系编号
        String departmentId = dd.selectByDepartmentName(departmentName);
        //查询 sclass 表,根据院系编号获取该院系的所有班级名称
        ArrayList < String > allClassNames =
                  cd.selectClassNamesByDepartmentId(departmentId);
        //将查询到的班级名称添加到"班级"组合框中
        for (String className : allClassNames) {
          cmbClassName.addItem(className);
        }
        //为"院系"组合框添加事件处理
        cmbDepartmentName.addItemListener(new DepartmentAndClassLinked(
                    cmbDepartmentName,cmbClassName));
        jp12.add(lblSId);
        jp12.add(txtSId);
        jp12.add(lblDepartment);
        jp12.add(cmbDepartmentName);
        jp12.add(lblClass);
        jp12.add(cmbClassName);
        jp1.add(jp12);
        //窗口初始化时默认以下控件不显示
        lblSId.setVisible(false);
        txtSId.setVisible(false);
        lblClass.setVisible(false);
        cmbClassName.setVisible(false);
        lblDepartment.setVisible(false);
        cmbDepartmentName.setVisible(false);
        //创建表格的默认模式对象
        model = new DefaultTableModel();
        //利用模式创建表格
        //创建表格的默认模式对象
        model = new DefaultTableModel();
        //利用模式创建表格
        table = new JTable(model);
        //设置表格选择模式为单一选择
        table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        students = sd.getAllStudents();              //查询 student 表,获取所有学生的信息
        setTableContent(students);                   //将所有的学生信息在表格中显示出来
        sp = new JScrollPane(table);                 //将表格放到滚动面板中
        jp2.add(sp, BorderLayout.CENTER);
        jp2.add(jp1, BorderLayout.NORTH);
        this.add(jp2);
        this.setSize(850,550);
        this.setLocationRelativeTo(null);
}
//自定义方法:将相应的学生信息在表格中显示出来
public void setTableContent(List < Student > student){
    int rows = students.size();
    content = new Object[rows][8];
    int i = 0;
    for(Student s:students){
        content[i][0] = s.getStudentId();
```

```
            content[i][1] = s.getStudentName();
            content[i][2] = s.getStudentSex();
            content[i][3] = s.getStudentBirthday();
            SClass c = cd.selectByClassId(s.getClassId());
            String d = dd.selectByDepartmentId(c.getDepartmentId());
            content[i][4] = d;
            content[i][5] = c.getClassName();
            content[i][6] = s.getStudentTel();
            content[i][7] = s.getStudentAddress();
            i++;
        }
        model.setDataVector(content, title);
      }
    }
```

(7) 查询学生信息窗口中的单选按钮事件处理。

接下来修改 SelectStudent 窗口类的构造方法,实现对各个单选按钮进行事件处理的功能。

① “查询所有学生”单选按钮的事件处理。

当选中该按钮时,lblSId、txtSId、lblDepartment、department、lblClass、className 控件都不显示。

代码如下:

```
selectAll.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        lblSId.setVisible(false);
        txtSId.setVisible(false);
        lblDepartment.setVisible(false);
        cmbDepartmentName.setVisible(false);
        lblClass.setVisible(false);
        cmbClassName.setVisible(false);
    }
});
```

② “按学号查询学生”单选按钮的事件处理。

当选中该按钮时,lblSId、txtSId 显示,而 lblDepartment、department、lblClass、className 控件都不显示。同时“学号”文本框(txtSId)获得焦点。

```
selectById.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //TODO Auto-generated method stub
        lblSId.setVisible(true);
        txtSId.setVisible(true);
        lblDepartment.setVisible(false);
        cmbDepartmentName.setVisible(false);
        lblClass.setVisible(false);
        cmbClassName.setVisible(false);
        txtSId.requestFocus();                    //"学号"文本框获得焦点
    }
});
```

③ “按班级查询学生”单选按钮的事件处理。

当选中该按钮时,lblSId、txtSId 不显示,而 lblDepartment、department、lblClass、

className 显示。

```
selectByClass.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //TODO Auto-generated method stub
        lblSId.setVisible(false);
        txtSId.setVisible(false);
        lblDepartment.setVisible(true);
        cmbDepartmentName.setVisible(true);
        lblClass.setVisible(true);
        cmbClassName.setVisible(true);
    }
});
```

④ “按院系查询学生”单选按钮的事件处理。

当选中该按钮时，lblDepartment、department 显示，而 lblSId、txtSId、lblClass、className 控件都不显示。

```
selectByDepartment.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        lblDepartment.setVisible(true);
        cmbDepartmentName.setVisible(true);
        lblSId.setVisible(false);
        txtSId.setVisible(false);
        lblClass.setVisible(false);
        cmbClassName.setVisible(false);
    }
});
```

(8) “查询”按钮的事件处理。

在 SelectStudent 类的构造方法中添加如下代码，完成“查询”按钮的事件处理。当单击“查询”按钮后，程序首先要判断哪个单选按钮被选中，然后根据选项调用相应的查询方法完成查询并将查询结果显示在表格中。

代码如下：

```
ok.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    //根据选中的单选按钮调用相应的查询方法查询出各自范围内的学生信息
        if(selectAll.isSelected()){
            students = sd.getAllStudents();
        }else if(selectById.isSelected()){
            Student s = sd.selectBySId(txtSId.getText());
            students.clear();
            if(s!= null){
                students.add(s);
            }
        }else if(selectByClass.isSelected()){
            String strDep = (String)department.getSelectedItem();
            String strClass = (String)className.getSelectedItem();
            String classId = cd.selectByClassNameAndDepartment(strClass,strDep);
            students = sd.getStudentsByClassId(classId);
    }else if(selectByDepartment.isSelected()){
```

```
            students = sd.getStudentsByDepartment((String)department.getSelectedItem());
              }
        //调用自定义方法 setTableContent(),将相应的学生信息在表格中显示出来
        setTableContent(students);
        }
    });
```

修改 MainFrame 窗口，在构造方法中添加如下代码，实现菜单项“查询学生信息”的事件处理。

```
selectStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        SelectStudent f = new SelectStudent();
        f.setVisible(true);
    }
});
```

**五、“添加学生信息”菜单项功能实现**

**1. 实现思路**

在添加学生信息时，提供了两种添加方式：一种是单个添加；另一种是批量添加。

当选择“单个添加学生信息”菜单项时，首先应该调出如图 3-36 所示的窗口，在该窗口中输入要添加学生的信息，然后单击“添加”按钮，实现添加一个学生信息的功能。

当选择“批量添加学生信息”菜单项时，首先应该调出如图 3-37 所示的“打开”对话框，然后在该对话框中选择一个包含多个学生信息的 Excel 文件(扩展名为.xls)，接着单击“打开”按钮，就可以实现批量添加学生信息的功能。


图 3-36　单个添加学生信息

**2. 代码实现**

下面先来实现“单个添加学生信息”菜单项的功能。

(1)“单个添加学生信息”菜单项的功能实现。

当在如图 3-36 所示的窗口中输入要添加学生的信息，然后单击“添加”按钮时，程序首先按照学号查询 student 表，如果该学号存在，则弹出如图 3-38 所示的提示对话框，提示“该


图 3-37　从 Excel 表中批量添加学生信息

学生已经存在,请重新输入学号”;如果 student 表中没有该学生,则向 student 表中添加一条新记录,如果添加成功,则弹出如图 3-39 所示的消息对话框,表示添加成功。

当单击“取消”按钮时,清空文本框中的内容,单选按钮初始化为“男”,出生日期初始化为“1990-01-01”。


图 3-38　学生已经存在的消息对话框


图 3-39　添加学生成功的消息对话框

首先创建如图 3-36 所示的窗口。

① 定义添加单个学生信息的窗口。

```
public class InsertStudent extends JFrame{
  //定义组件
  private JPanel jp;
  private JLabel lblSId,lblSName,lblSex,lblDepartment,lblClass,lblBirthday,lblYear,
      lblMonth,lblDay,lblTel,lblAddress;
  private JTextField txtSId,txtSName,txtTel,txtAddress;
  private ButtonGroup bg;
  private JRadioButton male,female;
  private JComboBox cmbDepartmentName,cmbClassName,year,day,month;
  private JButton btnAdd,cancel;
  //定义构造方法:初始化窗口
  public InsertStudent(){
    super("添加学生信息");
    //创建各个组件
    jp = new JPanel(null);
    lblSId = new JLabel("学号:");
    lblSName = new JLabel("姓名:");
    lblSex = new JLabel("性别:");
```

```
        lblBirthday = new JLabel("出生日期:");
        lblYear = new JLabel("年");
        lblMonth = new JLabel("月");
        lblDay = new JLabel("日");
        lblDepartment = new JLabel("院系:");
        lblClass = new JLabel("班级:");
        lblTel = new JLabel("联系电话:");
        lblAddress = new JLabel("家庭住址:");
        txtSId = new JTextField(10);
        txtSName = new JTextField(10);
        bg = new ButtonGroup();
        male = new JRadioButton("男",true);
        female = new JRadioButton("女");
        bg.add(male);
        bg.add(female);
        String s1[] = {"1990","1991","1992","1993","1994","1995","1996","1997","1998",
                   "1999","2000","2001","2002","2003","2004","2005","2006","2007"};
        year = new JComboBox(s1);
      String s2[] = {"01","02","03","04","05","06","07","08","09","10","11","12"};
      month = new JComboBox(s2);
      String s3[] = {"01","02","03","04","05","06","07","08","09","10","11","12","13",
                   "14","15","16","17","18","19","20","21","22","23","24","25","26","27",
                   "28","29","30","31"};
      day = new JComboBox(s3);
      //访问 department 表,获取所有院系名称
      DepartmentDao dd = new DepartmentDao();
      ArrayList < String > allDepartmentNames = dd.getAllDepartmentName();
      //创建"院系"组合框对象
      cmbDepartmentName = new JComboBox();
      //初始化"院系"组合框:将查询到的院系名称添加到"院系"组合框中
      for (String name : allDepartmentNames) {
        cmbDepartmentName.addItem(name);
    }
      //创建"班级"组合框对象
      cmbClassName = new JComboBox();
      //初始化"班级"组合框:根据"院系"组合框中的第一个院系名称查询 sclass 表,获取该院系的所有
      //班级名称并添加到"班级"组合框中
      //获取用户选中的院系名称
      String departmentName = (String)cmbDepartmentName.getItemAt(0);
      //查询 department 表,根据院系名称获取院系编号
      String departmentId = dd.selectByDepartmentName(departmentName);
      //查询 sclass 表,根据院系编号获取该院系的所有班级名称
      SClassDao cd = new SClassDao();
      ArrayList < String > allClassNames = cd.selectClassNamesByDepartmentId(departmentId);
      //将查询到的班级名称添加到"班级"组合框中
      for (String className : allClassNames) {
          cmbClassName.addItem(className);
      }
      //为"院系"组合框添加事件处理,实现院系和班级联动的动态效果
      cmbDepartmentName.addItemListener(new DepartmentAndClassLinked(cmbDepartmentName,
cmbClassName));
      txtTel = new JTextField(12);
      txtAddress = new JTextField(20);
      btnAdd = new JButton("添加");
```

```
 cancel = new JButton("取消");
//将中间面板添加到窗口中,将各个组件添加到面板中
this.add(jp);
jp.add(lblSId);
jp.add(txtSId);
jp.add(lblSName);
jp.add(txtSName);
jp.add(lblSex);
jp.add(male);
jp.add(female);
jp.add(lblBirthday);
jp.add(year);
jp.add(lblYear);
jp.add(month);
jp.add(lblMonth);
jp.add(day);
jp.add(lblDay);
jp.add(lblDepartment);
jp.add(cmbDepartmentName);
jp.add(lblClass);
jp.add(cmbClassName);
jp.add(lblTel);
jp.add(txtTel);
jp.add(lblAddress);
jp.add(txtAddress);
jp.add(btnAdd);
jp.add(cancel);
//设置各个组件的位置和大小
lblSId.setBounds(50, 20, 100, 15);
lblSName.setBounds(50, 50, 100, 15);
lblSex.setBounds(50, 80, 100, 15);
lblBirthday.setBounds(50, 110, 100, 15);
lblDepartment.setBounds(50, 140, 100, 15);
lblClass.setBounds(50, 170, 100, 15);
lblTel.setBounds(50, 200, 100, 15);
lblAddress.setBounds(50, 230, 100, 15);
txtSId.setBounds(150, 20, 210, 21);
txtSName.setBounds(150, 50, 210, 21);
male.setBounds(180, 80, 90, 21);
female.setBounds(270, 80, 90, 21);
year.setBounds(150, 110, 60, 21);
lblYear.setBounds(210, 110, 30, 21);
month.setBounds(240, 110, 40, 21);
lblMonth.setBounds(280, 110, 30, 21);
day.setBounds(310, 110, 40, 21);
lblDay.setBounds(350, 110, 30, 21);
cmbDepartmentName.setBounds(150, 140, 210, 21);
cmbClassName.setBounds(150, 170, 210, 21);
txtTel.setBounds(150, 200, 210, 21);
txtAddress.setBounds(150, 230, 210, 21);
btnAdd.setBounds(100, 270, 100, 21);
cancel.setBounds(230, 270, 100, 21);
//将光标置于"学号"文本框
txtSId.requestFocus();
```

```
 this.setSize(420, 350);
 this.setLocationRelativeTo(null);              //Frame 居中
 this.setResizable(false);                      //禁止改变框架大小
}}
```



接下来，实现该窗口中的“添加”按钮的事件处理：访问 student 表，完成学生信息的添加。

② 修改 StudentDao 类，添加 insertStudent()方法，实现添加新学生信息的功能。

由于添加学生信息时有两种方式：单个添加和批量添加，为了使该方法具有通用性，因此规定了该方法的参数为集合类型，返回为 int 类型，表示添加成功的个数。

代码如下：

```
public int insertStudent(ArrayList < Student > stus){
    int count = 0;
    Connection con = null;
    PreparedStatement pstmt = null;
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("insert into student values(?,?,?,?,?,?,?)");
        for (Student s:stus) {
          if(selectBySId(s.getStudentId())!= null) continue;
            pstmt.setString(1, s.getStudentId());
            pstmt.setString(2, s.getStudentName());
            pstmt.setString(3, s.getStudentSex());
            pstmt.setDate(4, s.getStudentBirthday());
            pstmt.setString(5, s.getClassId());
            pstmt.setString(6, s.getStudentTel());
            pstmt.setString(7, s.getStudentAddress());
            int n = pstmt.executeUpdate();
            if(n == 1){
             count++;
             }
        }
    }catch(Exception e){
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "向 student 表中添加信息失败!");
    }finally{
        try{
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
        }catch( Exception e){
            e.printStackTrace( );
        }
    }
    return count;
}
```

③ 修改 InsertStudent 类的构造方法，实现“添加”按钮的事件处理。

```
btnAdd.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        //获取输入的学号、姓名、家庭住址，进行非空判断
```

```
        String sId = txtSId.getText();
        String sName = txtSName.getText();
        String sAddress = txtAddress.getText();
        if("".equals(sId)||"".equals(sName)||"".equals(sAddress)){
            JOptionPane.showMessageDialog(InsertStudent.this.btnAdd, "学号、姓名、家庭住址
不允许为空!");
        }else{
            //String sId1 = txtSId.getText();
            StudentDao sd = new StudentDao();
            Student student = sd.selectBySId(sId);
            if(student!= null){
                JOptionPane.showMessageDialog(InsertStudent.this.btnAdd, "该学生已存在,请
重新输入学号.");
            }else{
                student = new Student();
                student.setStudentId(sId);
                student.setStudentName(txtSName.getText());
                student.setStudentAddress(txtAddress.getText());
                student.setStudentTel(txtTel.getText());
                if(male.isSelected())
                    student.setStudentSex("男");
                else
                    student.setStudentSex("女");
                String strYear = (String)year.getSelectedItem();
                String strMonth = (String)month.getSelectedItem();
                String strDay = (String)day.getSelectedItem();
                Date birthday = Date.valueOf(strYear + "-" + strMonth + "-" + strDay);
                student.setStudentBirthday(birthday);
                //给 student 对象的班级编号属性赋值
                SClassDao cd = new SClassDao();
                String strClass = (String)cmbClassName.getSelectedItem();
                String strDep = (String)cmbDepartmentName.getSelectedItem();
                String classId = cd.selectByClassNameAndDepartment(strClass, strDep);
                student.setClassId(classId);
                ArrayList<Student> stus = new ArrayList<Student>();
                stus.add(student);
                int n = sd.insertStudent(stus);
                if(n == 1)
                 JOptionPane.showMessageDialog(InsertStudent.this.btnAdd, "添加学生成功!");
                else
                 JOptionPane.showMessageDialog(InsertStudent.this.btnAdd, "添加学生失败!");
                }
            }
        }
    });
```

④ 修改 InsertStudent 类的构造方法,添加“取消”按钮的事件处理。

```
cancel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        txtSId.setText("");
        txtSName.setText("");
        txtAddress.setText("");
        txtTel.setText("");
        male.setSelected(true);
```

```
            year.setSelectedIndex(0);
            month.setSelectedIndex(0);
            day.setSelectedIndex(0);
            txtSId.requestFocus();
        }
    });
```

接着，修改 MainFrame 类，在其构造方法中添加如下代码，实现单击菜单项“添加单个学生信息”的事件处理。

```
insertOneStudent.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        InsertStudent f = new InsertStudent();
        f.setVisible(true);
}
```

（2）“批量添加学生信息”菜单项的实现。

代码如下：

```
insertMoreStudent.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        ArrayList < Student > allStu = new ArrayList < Student >();
        //创建"文件"对话框对象
        JFileChooser jfc = new JFileChooser();
        //设置"文件"对话框的默认路径为 C 盘根目录
        jfc.setCurrentDirectory(new File("C:\\"));
        //显示文件"打开"对话框，并返回一个整数值
        int val = jfc.showOpenDialog(MainFrame.this);
        //如果单击"文件"对话框中的"打开"按钮，则打开选中的 Excel 文件，并将文件中
        //的内容添加到数据表中
        if(val == JFileChooser.APPROVE_OPTION){
        //获得文件路径和文件名并创建对应的文件对象
        String fileName = jfc.getSelectedFile().getName();
        File f = new File( jfc.getCurrentDirectory().toString() +
                        jfc.getSelectedFile().getName());
        if(fileName == null||!(fileName.endsWith(".xls"))){
            JOptionPane.showMessageDialog(jfc, "请选择一个.xls 文件");
        }else{
          //读取 Excel 文件流
          InputStream is = null;
          try {
            is = new FileInputStream(f.getAbsolutePath());
          } catch (FileNotFoundException e1) {
              e1.printStackTrace();
        }
          //使用 jxl 包中的 Workbook 类、Sheet 类获取 Excel 文件中的数据并保存到集合中
          Workbook rwb = null;
          try {
            rwb = Workbook.getWorkbook(is);
            //获取当前 Excel 文件中共有几个表，然后对每个表分别进行处理
            Sheet[] sheets = rwb.getSheets();
            for (int n = 0; n < sheets.length; n++) {
              Sheet sheet = sheets[n];
              //有多少行
              int rows = sheet.getRows();
```

```
                //行循环:处理每行的数据.第 0 行是表头,不需要处理
                for (int i = 1; i < rows; i++) {
                    //将当前行中的数据封装为一个 Student 对象
                    String studentId = sheet.getCell(0,i).getContents();
                    String studentName = sheet.getCell(1,i).getContents();
                    String studentSex = sheet.getCell(2,i).getContents();
                    String classId = sheet.getCell(4,i).getContents();
                    String studentTel = sheet.getCell(5,i).getContents();
                    String studentAddress = sheet.getCell(6,i).getContents();
                    Student s = new Student(studentId,studentName,studentSex,new
                        Date(1994,11,14),classId,studentTel,studentAddress);
                    //将 student 对象添加到集合 allStu 中
                    allStu.add(s);
                }
            }
            //访问数据库,将上述 allStu 集合中的学生信息添加到 student 表中
            StudentDao sd = new StudentDao();
            int count = sd.insertStudent(allStu);
            JOptionPane.showMessageDialog(null, "成功添加了" + count + "个学生信息");
            } catch (BiffException e1) {
                // TODO Auto - generated catch block
                e1.printStackTrace();
            } catch (IOException e1) {
                // TODO Auto - generated catch block
                e1.printStackTrace();
}     }         }     }     });
```

## 六、“修改学生信息”菜单项功能实现

### 1. 实现思路

当单击“修改学生信息”菜单项时,首先应该调出如图 3-40 所示的窗口,当在该窗口中输入学号并按 Enter 键后,触发学号文本框的 ActionEvent 事件,如果该学号不存在,则弹出如图 3-41 所示的消息提示对话框,提示该学号不存在,要求重新输入学号;如果该学号存在,则获取该学生的信息并显示到图 3-40 的相应位置。这时,用户可以按照需求修改该生的相关信息。修改完毕后单击“修改”按钮,如果修改成功则弹出如图 3-42 所示的消息提示对话框,否则弹出如图 3-43 所示的消息提示对话框。当单击“取消”按钮时,清空文本框中的内容,单选按钮初始化为“男”,出生日期初始化为“1990-01-01”。


图 3-40 “修改学生信息”窗口


图 3-41　学生不存在消息提示对话框


图 3-42　修改成功消息提示对话框


图 3-43　修改失败消息提示对话框

**2. 代码实现**

(1) 定义修改学生信息的窗口类 ModifyStudent。

该窗口的代码与 InsertStudent 类相似。

```
public class ModifyStudent extends JFrame{
  private JPanel jp;
  private JLabel lblSId,lblSName,lblSex,lblDepartment,lblClass,lblBirthday,lblYear,
              lblMonth,lblDay,lblTel,lblAddress;
  private JTextField txtSId,txtSName,txtTel,txtAddress;
  private ButtonGroup bg;
  private JRadioButton male,female;
  private JComboBox cmbDepartmentName,cmbClassName,year,month,day;
  private JButton btnAdd,cancel;
  private JLabel alert;
  public ModifyStudent(){
    super("修改学生信息");
    jp = new JPanel(null);
    alert = new JLabel("请先输入要修改的学生的学号,输完后按 Enter 键!");
    alert.setForeground(Color.red);
    lblSId = new JLabel("学号:");
    lblSName = new JLabel("姓名:");
    lblSex = new JLabel("性别:");
    lblBirthday = new JLabel("出生日期:");
    lblYear = new JLabel("年");
    lblMonth = new JLabel("月");
    lblDay = new JLabel("日");
    lblDepartment = new JLabel("院系:");
    lblClass = new JLabel("班级:");
    lblTel = new JLabel("联系电话:");
    lblAddress = new JLabel("家庭住址:");
    txtSId = new JTextField(10);
    txtSName = new JTextField(10);
    bg = new ButtonGroup();
    male = new JRadioButton("男",true);
    female = new JRadioButton("女");
    bg.add(male);
    bg.add(female);
```

```
        String s1[] = {"1990","1991","1992","1993","1994","1995","1996","1997","1998",
                "1999","2000","2001","2002","2003","2004","2005","2006","2007"};
        year = new JComboBox(s1);
        String s2[] = {"01","02","03","04","05","06","07","08","09","10","11","12"};
        month = new JComboBox(s2);
        String s3[] = {"01","02","03","04","05","06","07","08","09","10","11","12",
                "13","14","15","16","17","18","19","20","21","22","23","24","25","26",
                "27","28","29","30","31"};
        day = new JComboBox(s3);
        //访问 department 表,获取所有院系名称
        DepartmentDao dd = new DepartmentDao();
        ArrayList < String > allDepartmentNames = dd.getAllDepartmentName();
        //创建"院系"组合框对象
        cmbDepartmentName = new JComboBox();
        //初始化"院系"组合框:将查询到的院系名称添加到"院系"组合框中
        for (String name : allDepartmentNames) {
            cmbDepartmentName.addItem(name);
        }
        //创建"班级"组合框对象
        cmbClassName = new JComboBox();
        //初始化"班级"组合框:根据"院系"组合框中的第一个院系名称查询 sclass 表,获取该院系的
        //所有班级名称并添加到"班级"组合框中
        //获取用户选中的院系名称
        String departmentName = (String)cmbDepartmentName.getItemAt(0);
        //查询 department 表,根据院系名称获取院系编号
        String departmentId = dd.selectByDepartmentName(departmentName);
        //查询 sclass 表,根据院系编号获取该院系的所有班级名称
        SClassDao cd = new SClassDao();
        ArrayList < String > allClassNames = cd.selectClassNamesByDepartmentId(departmentId);
        //将查询到的班级名称添加到"班级"组合框中
        for (String className : allClassNames) {
            cmbClassName.addItem(className);
        }
        //为"院系"组合框添加事件处理,实现院系和班级联动的动态效果
        cmbDepartmentName.addItemListener(new DepartmentAndClassLinked(cmbDepartmentName,
cmbClassName));
        txtTel = new JTextField(12);
        txtAddress = new JTextField(20);
        btnAdd = new JButton("修改");
        cancel = new JButton("取消");
        this.add(jp);
        jp.add(alert);
        jp.add(lblSId);
        jp.add(txtSId);
        jp.add(lblSName);
        jp.add(txtSName);
        jp.add(lblSex);
        jp.add(male);
        jp.add(female);
        jp.add(lblBirthday);
        jp.add(year);
        jp.add(lblYear);
        jp.add(month);
        jp.add(lblMonth);
```

```
        jp.add(day);
        jp.add(lblDay);
        jp.add(lblDepartment);
        jp.add(cmbDepartmentName);
        jp.add(lblClass);
        jp.add(cmbClassName);
        jp.add(lblTel);
        jp.add(txtTel);
        jp.add(lblAddress);
        jp.add(txtAddress);
        jp.add(btnAdd);
        jp.add(cancel);
        alert.setBounds(60, 0, 300, 15);
        lblSId.setBounds(50, 20, 100, 15);
        lblSName.setBounds(50, 50, 100, 15);
        lblSex.setBounds(50, 80, 100, 15);
        lblBirthday.setBounds(50, 110, 100, 15);
        lblDepartment.setBounds(50, 140, 100, 15);
        lblClass.setBounds(50, 170, 100, 15);
        lblTel.setBounds(50, 200, 100, 15);
        lblAddress.setBounds(50, 230, 100, 15);
        txtSId.setBounds(150, 20, 210, 21);
        txtSName.setBounds(150, 50, 210, 21);
        male.setBounds(180, 80, 90, 21);
        female.setBounds(270, 80, 90, 21);
        year.setBounds(150, 110, 60, 21);
        lblYear.setBounds(210, 110, 30, 21);
        month.setBounds(240, 110, 40, 21);
        lblMonth.setBounds(280, 110, 30, 21);
        day.setBounds(310, 110, 40, 21);
        lblDay.setBounds(350, 110, 30, 21);
        cmbDepartmentName.setBounds(150, 140, 210, 21);
        cmbClassName.setBounds(150, 170, 210, 21);
        txtTel.setBounds(150, 200, 210, 21);
        txtAddress.setBounds(150, 230, 210, 21);
        btnAdd.setBounds(100, 270, 100, 21);
        cancel.setBounds(230, 270, 100, 21);
        txtSId.requestFocus();                          //将光标置于学号文本框
        this.setSize(420,350);
        setLocationRelativeTo(null);                    //居中
        setResizable(false);
        }
}
```

(2) 实现“输入学号后按 Enter 键”的事件处理。

```
txtSId.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    String sId = txtSId.getText();
    if(sId.equals("")){
        JOptionPane.showMessageDialog(ModifyStudent.this.btnModify, "学号不能为空!");
        txtSId.requestFocus();
    }else{
        StudentDao sd = new StudentDao();
        Student student = sd.selectBySId(sId);
```

```
        if(student == null){
          JOptionPane.showMessageDialog(ModifyStudent.this.btnModify, "学号:" + sId + "不存
在,请重新输入!");
          txtSId.setText("");
          txtSId.requestFocus();
        }
        else{
            //获取 student 表中当前学生的姓名、电话、家庭住址并显示到相应文本框中
            txtSName.setText(student.getStudentName());
            txtTel.setText(student.getStudentTel());
            txtAddress.setText(student.getStudentAddress());
            //获取 student 表中当前学生的性别信息,利用单选按钮显示
            if("男".equals(student.getStudentSex()))
                male.setSelected(true);
            else
                female.setSelected(true);
            //获取 student 表中当前学生的出生日期,然后分解成年、月、日显示到相应的组合框中
            Date birthday = student.getStudentBirthday();
            String strbirth = birthday.toString();
            String y = strbirth.substring(0,4);
            String m = strbirth.substring(5,7);
            String d = strbirth.substring(8,10);
            year.setSelectedItem(y);
            month.setSelectedItem(m);
            day.setSelectedItem(d);
            //利用当前学生的班级编号获取班级名称和所在院系并显示到相应文本框中
            SClassDao cd = new SClassDao();
            SClass c = cd.selectByClassId(student.getClassId());
            if(c!= null){
            DepartmentDao dd = new DepartmentDao();
            String departmentName = dd.selectByDepartmentId(c.getDepartmentId());
                        cmbDepartmentName.setSelectedItem(departmentName);
            cmbClassName.setSelectedItem(c.getClassName());
        }       }    }    }    });
```

这时当输入学号并按 Enter 键后,当前学生的信息就显示到相应位置上,用户可以在此基础上修改学生信息。修改完毕后,用户应当单击“修改”按钮实现数据表中数据的修改过程,这需要执行 update 命令来完成。

(3) 修改 StudentDao 类,添加 modifyStudent()方法,实现修改指定学号的学生信息的功能。

```
public boolean modifyStudent(Student s){
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    //实例化数据库工具
    DBUtil dbUtil = new DBUtil();
    boolean flag = false;
    //打开数据库,获取连接对象
    try{
      con = dbUtil.getConnection();
      pstmt = con.prepareStatement("update student set student_name = ?, student_sex = ?,
student_birthday = ?,class_id = ?,student_tel = ?,student_address = ? where student_id = ?");
      //创建 PreparedStatment 对象
```

```
        pstmt.setString(1,s.getStudentName());
        pstmt.setString(2,s.getStudentSex());
        pstmt.setDate(3,s.getStudentBirthday());
        pstmt.setString(4,s.getClassId());
        pstmt.setString(5,s.getStudentTel());
        pstmt.setString(6,s.getStudentAddress());
        pstmt.setString(7,s.getStudentId());
        int n = pstmt.executeUpdate();
        if(n > 0) flag = true;
    }catch(Exception e){
        JOptionPane.showMessageDialog(null,"访问 student 表失败!");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        return flag;
    }
```

(4) 实现“修改”按钮的事件处理。

```
        btnModify.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                Student student = new Student();
                //给 student 对象的属性赋值
                student.setStudentId(txtSId.getText());
                student.setStudentName(txtSName.getText());
                student.setStudentAddress(txtAddress.getText());
                student.setStudentTel(txtTel.getText());
                if(male.isSelected())
                  student.setStudentSex("男");
                else
                  student.setStudentSex("女");
                String strYear = (String)year.getSelectedItem();
                String strMonth = (String)month.getSelectedItem();
                String strDay = (String)day.getSelectedItem();
                Date birthday = Date.valueOf(strYear + " - " + strMonth + " - " + strDay);
                student.setStudentBirthday(birthday);
                SClassDao cd = new SClassDao();
                String strClass = (String)cmbClassName.getSelectedItem();
                String strDep = (String)cmbDepartmentName.getSelectedItem();
                String classId = cd.selectByClassNameAndDepartment(strClass, strDep);
                //给 student 对象的班级编号属性赋值
                if(classId == null){
                    JOptionPane.showMessageDialog(ModifyStudent.this.btnAdd, "您所选择的班级
或院系有误,请重新选择!");
                    return;
                }else{
```

```
            student.setClassId(classId);
        }
         StudentDao sd = new StudentDao();
         boolean flag = sd.modifyStudent(student);
         if(flag)
             JOptionPane.showMessageDialog(ModifyStudent.this.btnAdd, "修改学生信息成功!");
         else
             JOptionPane.showMessageDialog(ModifyStudent.this.btnAdd, "修改学生信息失败!");
         }});
```

(5)“取消”按钮的事件处理。

```
cancel.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            txtSId.setText("");
            txtSName.setText("");
            txtAddress.setText("");
            txtTel.setText("");
            male.setSelected(true);
            year.setSelectedIndex(0);
            month.setSelectedIndex(0);
            day.setSelectedIndex(0);
            txtSId.requestFocus();
        } });
```

(6) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“修改学生信息”的事件处理。

```
modifyStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ModifyStudent f = new ModifyStudent();
        f.setVisible(true);
    }});
```

### 七、“删除学生信息”菜单项功能实现

#### 1. 实现思路

当单击“删除学生信息”菜单项时,首先应该调出如图 3-44 所示的窗口,当在该窗口中的文本框中输入学号并单击“删除”按钮时,如果该学号不存在,则弹出如图 3-45 所示的消息提示对话框;如果存在,则弹出如图 3-46 所示的确认对话框,单击“是”按钮则删除该学生的信息,单击“否”按钮则不会删除。当单击图 3-44 所示的窗口中的“取消”按钮时会清空学号文本框中的内容。

注意,在删除学生信息的同时,应该相应的删除用户表中的相关信息。


图 3-44 “删除学生信息”窗口


图 3-45 学生不存在消息提示对话框


图 3-46　确认提示框

**2. 代码实现**

(1) 在 view 包中定义“删除学生信息”窗口。

```
public class DeleteStudent extends JFrame{
private JLabel lblSId;
private JTextField txtSId;
private JButton del,cancel;
private JPanel jp;
public DeleteStudent(){
    super("删除学生信息");
    jp = new JPanel(null);
    lblSId = new JLabel("请输入要删除的学生的学号:");
    txtSId = new JTextField(12);
    del = new JButton("删除");
    cancel = new JButton("取消");
    this.add(jp);
    jp.add(lblSId);
    jp.add(txtSId);
    jp.add(del);
    jp.add(cancel);
    lblSId.setBounds(60, 30, 180, 20);
    txtSId.setBounds(60, 60, 180, 20);
    del.setBounds(80, 90, 60, 23);
    cancel.setBounds(160, 90, 60, 23);
    this.setSize(300, 200);
    setLocationRelativeTo(null);                    //Frame 居中
    setResizable(false);                            //禁止改变框架大小
}
}
```

(2) 修改 MainFrame 窗口，在构造方法中添加如下代码，实现菜单项“删除学生信息”的事件处理。

```
deleteStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DeleteStudent f = new DeleteStudent();
        f.setVisible(true);
    }
});
```

接下来实现“删除学生信息”窗口中的“删除”按钮和“取消”按钮的事件处理。

(3) 修改 StudentDao 类，添加 deleteStudent()方法，实现在 student 表中删除指定学号的学生信息的功能。

```
public boolean deleteBySId(String sId) {
    boolean bool = false;
```

```
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    //实例化数据库工具
    DBUtil dbUtil = new DBUtil();
    //打开数据库,获取连接对象
    try{
      con = dbUtil.getConnection();
      //查询学生信息
      pstmt = con.prepareStatement("delete FROM student where student_id = ?");
      pstmt.setString(1,sId);
      int n = pstmt.executeUpdate();            //获取影响的行数
      if(n > 0)
        bool = true;
      }catch(Exception e){
        JOptionPane.showMessageDialog(null,"访问 student 表失败!");
      }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
      return bool;
  }
```

(4) 修改 DeleteStudent 类的构造方法,添加如下代码实现“删除”按钮的事件处理。

```
del.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      String sId = txtSId.getText();
      int r = JOptionPane.showConfirmDialog(del,"您确定要删除学号为:" + sId + "的学生信息
吗?","删除",JOptionPane.YES_NO_OPTION);
      if(r == JOptionPane.YES_OPTION){
        StudentDao sd = new StudentDao();
        boolean flag = sd.deleteBySId(sId);
        if(flag)
            JOptionPane.showMessageDialog(null,"删除成功!");
        else
            JOptionPane.showMessageDialog(null,"您要删除的学生不存在!");
}    }    });
```

(5) 修改 DeleteStudent 类的构造方法,添加如下代码实现“取消”按钮的事件处理。

```
cancel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      txtSId.setText("");
      txtSId.requestFocus();
}    });
```

至此,“删除学生信息”的菜单项就实现了。

## 八、“查看个人已选课程”菜单项功能实现

### 1. 实现思路

当单击“查看个人已选课程”菜单项时，首先应该调出如图 3-47 所示的窗口，通过本窗口可以了解到目前自己的选课情况。

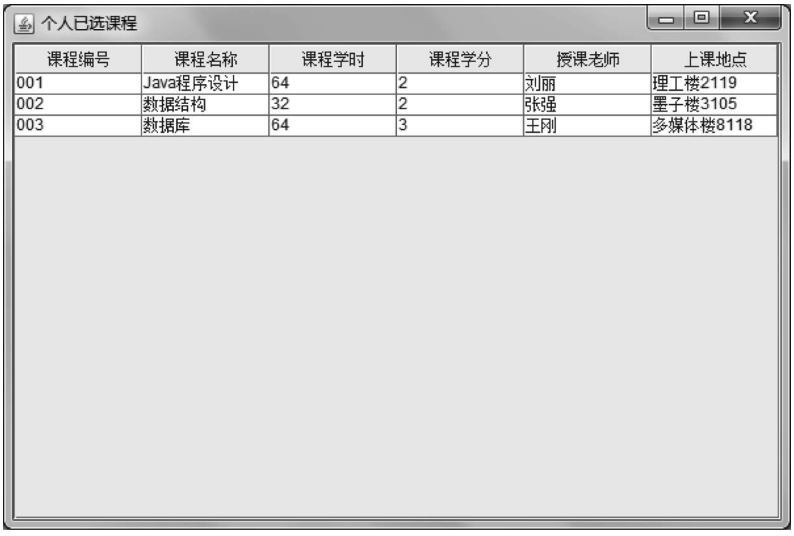



图 3-47 “个人已选课程”窗口

### 2. 代码实现

(1) 在 entity 包中定义实体类 Course。

```
public class Course {
  private String courseId;
  private String courseName;
  private int coursePeriod;
  private int courseCredit;
  private String courseTeacher;
  private String courseAddress;
  public String getCourseId() {
       return courseId;
  }
  public void setCourseId(String courseId) {
       this.courseId = courseId;
  }
  public String getCourseName() {
       return courseName;
  }
  public void setCourseName(String courseName) {
         this.courseName = courseName;
  }
  public int getCoursePeriod() {
       return coursePeriod;
  }
  public void setCoursePeriod(int coursePeriod) {
         this.coursePeriod = coursePeriod;
  }
```

```
    public int getCourseCredit() {
        return courseCredit;
    }
    public void setCourseCredit(int courseCredit) {
        this.courseCredit = courseCredit;
    }
    public String getCourseTeacher() {
        return courseTeacher;
    }
    public void setCourseTeacher(String courseTeacher) {
        this.courseTeacher = courseTeacher;
    }
    public String getCourseAddress() {
        return courseAddress;
    }
    public void setCourseAddress(String courseAddress) {
        this.courseAddress = courseAddress;
    }
public Course(String courseId, String courseName, int coursePeriod, int courseCredit, String
        courseTeacher,String courseAddress) {
        this.courseId = courseId;
        this.courseName = courseName;
        this.coursePeriod = coursePeriod;
        this.courseCredit = courseCredit;
        this.courseTeacher = courseTeacher;
        this.courseAddress = courseAddress;
    }
    public Course() {

    }
}
```

(2) 在 Dao 包中定义 XClassDao 类,添加 selectByCId()方法,通过用户 id 查找所选课程的 id 集合。

```
public ArrayList < String > selectByCId(String id){
    ArrayList < String > n = new ArrayList < String >();
    int i = 0;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from xclass where student_id = ? ");
        pstmt.setString(1, id);
        rs = pstmt.executeQuery();
        while(rs.next())
        {
              n.add(rs.getString(2));
        }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 xclass 表失败!");
        e.printStackTrace();
```

```
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
        }
      return n;
    }
```

（3）在 Dao 包中定义 CourseDao 类，添加 selectByCourseIdx()方法，通过课程 id 获取课程对象。

```
public Course selectByCourseIdx(String cid){
 Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    Course course = null;
    int i = 0;
    try{
      con = dbUtil.getConnection();
      pstmt = con.prepareStatement("select * from course where course_id = ? ");
      pstmt.setString(1, cid);
      rs = pstmt.executeQuery();
      while(rs.next()){
      course = new Course(rs.getString(1),rs.getString(2),rs.getInt(3),rs.getInt(4),
            rs.getString(5),rs.getString(6));
     }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 course 表失败!");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
        }catch( Exception e){
            e.printStackTrace( );
            }
    }
 return course;
 }
```

（4）在 view 包中定义查看个人已选课程的窗口。

```
public class DisplaySelfCourse extends JFrame{
  private JTable table;
```

```
  private DefaultTableModel model;
  private JScrollPane jsp;
  private JPanel jp;
  ArrayList < Course > sc = new ArrayList < Course >();
  int rows = 0;
  int i = 0;
  String title[ ] = {"课程编号","课程名称","课程学时","课程学分","授课老师","上课地点"};
  Object content[ ][ ] = null;
  public DisplaySelfCourse(User user){
    jp = new JPanel(new BorderLayout());
    model = new DefaultTableModel();
    table = new JTable(model);
    jsp = new JScrollPane(table);
    this.add(jp);
    jp.add(jsp,BorderLayout.CENTER);
    CourseDao sd = new CourseDao();
    XClassDao xc = new XClassDao();
    ArrayList < String > Cid = xc.selectByCId(user.getUserId());
    while(i < Cid.size()){
      Course cc =  sd.selectByCourseIdx(Cid.get(i));
      sc.add(cc);
      i++;
    }
   rows = sc.size();
   setTableContent(rows);
   this.setTitle("个人已选课程");
   this.setSize(600, 400);
   this.setLocationRelativeTo(null);
 }
private void setTableContent(int rows2) {
    content = new Object[rows][6];
    int i = 0;
    for(Course s1:sc){
        content[i][0] = s1.getCourseId();
        content[i][1] = s1.getCourseName();
        content[i][2] = s1.getCoursePeriod();
        content[i][3] = s1.getCourseCredit();
        content[i][4] = s1.getCourseTeacher();
        content[i][5] = s1.getCourseAddress();
        i++;
    }
    model.setDataVector(content, title);
}
}
```

（5）修改 MainFrame 窗口，在构造方法中添加如下代码，实现菜单项“查看个人已选课程”的事件处理。

```
displaySelfCourse.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent arg0) {
        DisplaySelfCourse dsc = new DisplaySelfCourse(user);
        dsc.setVisible(true);
    } });
```

## 九、“学生选课”菜单项功能实现

### 1. 实现思路

当单击“学生选课”菜单项时，首先应该调出如图 3-48 所示的窗口，通过本窗口可以完成选课。如果已经选过，则弹出如图 3-49 所示的“该课程已选”消息提示对话框，否则弹出如图 3-50 所示的“选课成功”消息提示对话框。


图 3-48 “学生选课”窗口


图 3-49 “该课程已选”消息提示对话框


图 3-50 “选课成功”消息提示对话框

### 2. 代码实现

(1) 在 view 包中定义“学生选课”窗口。

```
public class SelectSelfCourse extends JFrame {
  private JTable table;
  private DefaultTableModel model;
  private JScrollPane jsp;
  private JPanel jpm,jp1;
  private JButton xuanze;
  ArrayList < Course > sc = null;
  int rows = 0;
  Object content[ ][ ] = null;
  String title[ ] = {"课程编号","课程名称","课程学时","课程学分","授课老师","上课地点"};
  public SelectSelfCourse(final User user){
     model = new DefaultTableModel();
     table = new JTable(model);
     jsp = new JScrollPane(table);
     table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

```
        jp1 = new JPanel();
        jpm = new JPanel(new BorderLayout());
        xuanze = new JButton("选择");
        jp1.add(xuanze);
        jpm.add(jsp,BorderLayout.CENTER);
        jpm.add(jp1,BorderLayout.SOUTH);
        CourseDao sd = new CourseDao();
        sc = sd.selectAllCourse();
        rows = sc.size();
        setTableContent(rows);
        this.add(jpm);
        this.setTitle("学生选课");
        this.setSize(600,400);
        this.setLocationRelativeTo(null);
    }
    private void setTableContent(int rows) {
         content = new Object[rows][6];
         int i = 0;
         for(Course s1:sc){
             content[i][0] = s1.getCourseId();
             content[i][1] = s1.getCourseName();
             content[i][2] = s1.getCoursePeriod();
             content[i][3] = s1.getCourseCredit();
             content[i][4] = s1.getCourseTeacher();
             content[i][5] = s1.getCourseAddress();
             i++;
         }
         model.setDataVector(content, title);
}    }
```

(2) 修改 XClassDao 类,添加 selectBySId()方法,通过用户 id 查找已选课程 id 集合。

```
public ArrayList<String> selectBySId(String id){
    ArrayList<String> string = new ArrayList<String>();
    int i = 0;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from xclass where student_id = ?");
        pstmt.setString(1, id);
        rs = pstmt.executeQuery();
        int n = 0;
        while(rs.next()){
            string.add(rs.getString(2));
        }
    }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 xclass 表失败!");
            e.printStackTrace();
    }finally{
            try{
                if(rs!= null)
                    rs.close();
```

```
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
        }
     return string;
    }
```

(3) 修改 XClassDao 类,添加 insertclass()方法,完成选课。

```
public boolean insertclass( String v,String xclass){
 boolean bool = false;
 Connection con  = null;
 PreparedStatement pstmt = null;
 ResultSet rs = null;
 DBUtil dbUtil = new DBUtil();
 try{
     con = dbUtil.getConnection();
     pstmt = con.prepareStatement("insert into xclass values(?,?)");
     pstmt.setString(1, v);
     pstmt.setString(2, xclass);
     int n = pstmt.executeUpdate();
     if(n > 0)
         bool = true;
 }catch(Exception e){
     e.printStackTrace();
     JOptionPane.showMessageDialog(null, "访问 xclass 表失败!");
 }finally{
     try{
         if(rs!= null)
             rs.close();
         if(pstmt!= null)
             pstmt.close();
         if(con!= null)
             con.close();
     }catch(Exception e){
         e.printStackTrace();
     }
  }
  return bool;
}
```

(4) “选择”按钮的事件处理。

```
xuanze.addActionListener( new ActionListener(){
 public void actionPerformed(ActionEvent arg0) {
     XClassDao s = new XClassDao();
     int index[ ] = table.getSelectedRows();
     String v =  table.getValueAt(index[0], 0).toString();
     ArrayList < String > as = s.selectBySId(user.getUserId());
     if(as!= null){
       if(as.contains(v)){
         JOptionPane.showMessageDialog(null, "该课程已选","提示",JOptionPane.INFORMATION_
MESSAGE);
```

```
        } else{
             boolean bool = s.insertclass(user.getUserId(),v);
             if(bool == true){
             JOptionPane.showMessageDialog(null, "选课成功","提示",JOptionPane.INFORMATION_
MESSAGE);
}else JOptionPane.showMessageDialog(null, "选课失败","提示",
             JOptionPane.WARNING_MESSAGE);
}    }  }    });
```

(5) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“学生选课”的事件处理。

```
selectSelfCourse.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent arg0) {
        SelectSelfCourse a = new SelectSelfCourse(user);
        a.setVisible(true);
}} );
```

## 十、“学生退课”菜单项功能实现

### 1. 实现思路

当单击“学生退课”菜单项时,首先应该调出如图 3-51 所示的窗口,当选择某一行并单击“退课”按钮时,则弹出如图 3-52 所示的询问对话框,如果单击“是”按钮,则退课成功,这时单击“刷新”按钮则会更新窗口。


图 3-51 “学生退课”窗口


图 3-52 询问对话框

**2. 代码实现**

(1) 修改 XClassDao 类,添加 deleteCourse()方法,完成退课。

```
public boolean deleteCourse(JTable table){
 int index[ ] = table.getSelectedRows();
 if(index.length == 0){
     JOptionPane.showMessageDialog(null, "请选择要退回的课程");
 }else{
     try{
        int k = JOptionPane.showConfirmDialog(null, "您确定不选该课程吗?");
        if(k == JOptionPane.YES_OPTION){
           Connection con = null;
           PreparedStatement pstmt = null;
           ResultSet rs = null;
           DBUtil dbUtil = new DBUtil();
           try{
             con = dbUtil.getConnection();
             String sno = table.getValueAt(index[0],0).toString();
             String cno = table.getValueAt(index[0], 1).toString();
             pstmt = con.prepareStatement("delete from xclass where course_id = ? and student_
id = ? ");
             pstmt.setString(1, cno);
             pstmt.setString(2, sno);
             int count = pstmt.executeUpdate();
             if(count == 1)
            return true;
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
           }catch( Exception e){
                e.printStackTrace( );
        }} }
     }catch(Exception e){
        e.printStackTrace();
 }    }
 return false;
}
```

(2) 在 view 包中定义学生退课窗口。

```
public class QuitCourse extends JFrame implements ActionListener{
  private JButton b_shuaxin,tuike;
  private DefaultTableModel model = new DefaultTableModel();
  private JPanel p_main,p_button;
  private JTable table = new JTable(model);
  private JScrollPane sp_table;
  String sid = null;
  public QuitCourse(User user){
    sid = user.getUserId();
    sp_table = new JScrollPane(table);
    table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

```
        tuike = new JButton("退课");
        b_shuaxin = new JButton("刷新");
        tuike.addActionListener(this);
        b_shuaxin.addActionListener(this);
        p_button = new JPanel();
        p_button.add(b_shuaxin);
        p_button.add(tuike);
        p_main = new JPanel(new BorderLayout());
        p_main.add(sp_table,BorderLayout.CENTER);
        p_main.add(p_button,BorderLayout.SOUTH);
        sp_table.setBounds(10,200,430,250);
        tuike.setBounds(150,300,80,80);
        b_shuaxin.setBounds(300,300,80,80);
        this.chaKan(sid);
        this.setContentPane(p_main);
        this.setTitle("学生退课");
        this.setSize(600, 400);
        this.setLocationRelativeTo(null);
    }
    public void actionPerformed(ActionEvent e){
        Object obj = e.getSource();
        if(obj == b_shuaxin){
            chaKan(sid);
        }else if(obj == tuike){
            XClassDao c = new XClassDao();
            boolean flag = c.deleteCourse(table);
            if(flag)
                JOptionPane.showMessageDialog(null, "退课成功");
            else
                JOptionPane.showMessageDialog(null, "退课失败");
            }     }
    public void chaKan(String sid){
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        DBUtil dbUtil = new DBUtil();
        try{
            con = dbUtil.getConnection();
            pstmt = con.prepareStatement("select * from xclass where student_id = ? ");
            pstmt.setString(1, sid);
            rs = pstmt.executeQuery();
            ResultSetMetaData rsmd = rs.getMetaData();
            int colCount = rsmd.getColumnCount();
            Vector < String > title = new Vector < String >();
            title.add("学号");
            title.add("课程编号");
            Vector < Vector < String >> data = new Vector < Vector < String >>();
            int rowCount = 0;
            while(rs.next()){
                rowCount++;
                Vector < String > rowdata = new Vector < String >();
                for(int i = 1;i < = colCount;i++){
                    rowdata.add(rs.getString(i));
                }
                data.add(rowdata);}
            if(rowCount = = 0){
                model.setDataVector(null, title);
            }else{
```

```
                model.setDataVector(data, title);
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
        }
    }    }
```

(3) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“学生退课”的事件处理。

```
quitCourse.addActionListener( new ActionListener(){
    public void actionPerformed(ActionEvent arg0) {
        QuitCourse dsc = new QuitCourse(user);
        dsc.setVisible(true);
}});
```

## 十一、“查询课程”菜单项功能实现

### 1. 实现思路

当单击“查询课程”菜单项时,调出如图 3-53 所示的查询课程的窗口,默认查询所有课程;也可以通过单选按钮实现如图 3-54 所示的按课程名称查询、如图 3-55 所示的按课程学分查询以及如图 3-56 所示的按授课老师查询。


图 3-53　查询所有课程


图 3-54　按课程名称查询


图 3-55　按课程学分查询


图 3-56　按授课老师查询

**2. 代码实现**

(1) 修改 CourseDao 类,添加 selectAllCourse()方法,实现查询所有课程。

```
//查看所有课程信息
  public ArrayList < Course > selectAllCourse(){
      ArrayList < Course > courses = new ArrayList < Course >();
      Connection con = null;
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      DBUtil dbUtil = new DBUtil();
      Course course = null;
      try{
          con = dbUtil.getConnection();
          pstmt = con.prepareStatement("select * from course");
          rs = pstmt.executeQuery();
          while(rs.next()){
              course = new Course(rs.getString(1),rs.getString(2),rs.getInt(3),rs.getInt(4),
                  rs.getString(5),rs.getString(6));
              courses.add(course);
          }
      }catch(Exception e){
          JOptionPane.showMessageDialog(null, "访问 course 表失败!");
      }finally{
          try{
              if(rs!= null)
                  rs.close();
              if(pstmt!= null)
                  pstmt.close();
```

```
            if(con!= null)
                con.close();
            }catch( Exception e){
                e.printStackTrace( );
            }
     }
     return courses;
 }
```

(2) 修改 CourseDao 类,添加 selectByName()方法,实现按课程名称查询课程。

```
public ArrayList < Course > selectByName(String name){
    ArrayList < Course > courses = new ArrayList < Course >();
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    Course course = null;
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from course where course_name = ?");
        pstmt.setString(1,name );
        rs = pstmt.executeQuery();
        while(rs.next()){
            course = new Course(rs.getString(1),rs.getString(2),rs.getInt(3),rs.getInt(4),
                    rs.getString(5),rs.getString(6));
            courses.add(course);
        }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 course 表失败!");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch( Exception e){
                e.printStackTrace( );
        }
    }
    return courses;
}
```

(3) 修改 CourseDao 类,添加 selectByCredit()方法,实现按课程学分查询课程。

```
public ArrayList < Course > selectByCredit(String credit){
 int i = Integer.parseInt(credit);
 ArrayList < Course > courses = new ArrayList < Course >();
 Connection con = null;
 PreparedStatement pstmt = null;
 ResultSet rs = null;
```

```
DBUtil dbUtil = new DBUtil();
Course course = null;
try{
    con = dbUtil.getConnection();
    pstmt = con.prepareStatement("select * from course where course_credit = ?");
    pstmt.setInt(1, i);
    rs = pstmt.executeQuery();
    while(rs.next()){
        course = new
Course(rs.getString(1),rs.getString(2),rs.getInt(3),rs.getInt(4),rs.getString(5),rs.
getString(6));
        courses.add(course);
    }
}catch(Exception e){
    JOptionPane.showMessageDialog(null, "访问 course 表失败!");
}finally{
    try{
        if(rs!= null)
            rs.close();
        if(pstmt!= null)
            pstmt.close();
        if(con!= null)
            con.close();
        }catch( Exception e){
          e.printStackTrace( );
   }
}
return courses;
}
```

(4) 修改 CourseDao 类,添加 selectByTeacher()方法,实现按授课老师查询课程。

```
public ArrayList<Course> selectByTeacher(String teacher){
   ArrayList<Course> courses = new ArrayList<Course>();
   Connection con = null;
   PreparedStatement pstmt = null;
   ResultSet rs = null;
   DBUtil dbUtil = new DBUtil();
   Course course = null;
   try{
       con = dbUtil.getConnection();
       pstmt = con.prepareStatement("select * from course where course_teacher = ?");
       pstmt.setString(1, teacher);
       rs = pstmt.executeQuery();
       while(rs.next()){
           course = new Course(rs.getString(1),rs.getString(2),rs.getInt(3),rs.getInt(4),
                   rs.getString(5),rs.getString(6));
           courses.add(course);
       }
   }catch(Exception e){
       JOptionPane.showMessageDialog(null, "访问 course 表失败!");
   }finally{
       try{
           if(rs!= null)
               rs.close();
```

```
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
            }catch( Exception e){
                e.printStackTrace( );
      }
    }
    return courses;
  }
```

(5) 在 view 包中定义“查询课程信息”窗口。

```
public class SelectCourse extends JFrame implements ActionListener{
  private JPanel jp1,jp11,jp12,jp2;
    private JRadioButton selectByName,selectByCredit,selectByTeacher,selectAll;
    private ButtonGroup bg;
    private JLabel lblName,lblCredit,lblTeacher;
    private JTextField txtName,txtTeacher,txtCredit;
    private DefaultTableModel model;
    private JTable table;
    private JScrollPane sp;
    private JButton ok;
    int rows = 0;
    ArrayList < Course > students = null;
    CourseDao sd = new CourseDao();
    Object content[][] = null;
    String title[] = {"课程编号","课程名称","课程学时","课程学分","授课老师","上课地点"};
     public SelectCourse(){
         setTitle("查询课程");
         jp1 = new JPanel();
         jp1.setLayout(new GridLayout(2,1));
         jp11 = new JPanel(new GridLayout(1,5));
         jp12 = new JPanel();
         jp2 = new JPanel();
         jp2.setBorder(new EmptyBorder(5,5,5,5));
         jp2.setLayout(new BorderLayout(0,0));
         bg = new ButtonGroup();
         selectAll = new JRadioButton("查询所有课程",true);
         selectByName = new JRadioButton("按课程名称查询");
         selectByCredit = new JRadioButton("按课程学分查询");
         selectByTeacher = new JRadioButton("按授课老师查询");
         bg.add(selectAll);
         bg.add(selectByName);
         bg.add(selectByCredit);
         bg.add(selectByTeacher);
         ok = new JButton("查询");
         ok.addActionListener(this);
         jp11.add(selectAll);
         jp11.add(selectByName);
         jp11.add(selectByCredit);
         jp11.add(selectByTeacher);
         jp11.add(ok);
         jp1.add(jp11);
         lblName = new JLabel("请输入课程名称:");
```

```
txtName = new JTextField(20);
lblTeacher = new JLabel("请输入授课老师:");
lblCredit = new JLabel("请输入课程学分:");
txtTeacher = new JTextField(20);
txtCredit = new JTextField(20);
jp12.add(lblName);
jp12.add(txtName);
jp12.add(lblTeacher);
jp12.add(txtTeacher);
jp12.add(lblCredit);
jp12.add(txtCredit);
jp1.add(jp12);
lblName.setVisible(false);
txtName.setVisible(false);
lblCredit.setVisible(false);
txtCredit.setVisible(false);
lblTeacher.setVisible(false);
txtTeacher.setVisible(false);
model = new DefaultTableModel();
table = new JTable(model);
table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
 students = sd.selectAllCourse();
rows = students.size();
setTableContent(rows);
sp = new JScrollPane(table);
jp2.add(sp, BorderLayout.CENTER);
jp2.add(jp1, BorderLayout.NORTH);
this.add(jp2);
this.setSize(700, 500);
this.setLocationRelativeTo(null);
selectAll.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e) {
      lblName.setVisible(false);
      txtName.setVisible(false);
      lblTeacher.setVisible(false);
      txtTeacher.setVisible(false);
      lblCredit.setVisible(false);
      txtCredit.setVisible(false);
      }
  });
  selectByName.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
          lblName.setVisible(true);
          txtName.setVisible(true);
          lblTeacher.setVisible(false);
          txtTeacher.setVisible(false);
          lblCredit.setVisible(false);
          txtCredit.setVisible(false);
          txtName.requestFocus();
      }   });
  selectByCredit.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
      lblName.setVisible(false);
      txtName.setVisible(false);
```

```
        lblTeacher.setVisible(false);
        txtTeacher.setVisible(false);
        lblCredit.setVisible(true);
        txtCredit.setVisible(true);
    }  });
    selectByTeacher.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e) {
             lblTeacher.setVisible(true);
             txtTeacher.setVisible(true);
             lblName.setVisible(false);
             txtName.setVisible(false);
             lblCredit.setVisible(false);
             txtCredit.setVisible(false);
          }  });     }
       public void setTableContent(int rows){
          content = new Object[rows][6];
          int i = 0;
          for(Course s:students){
              content[i][0] = s.getCourseId();
              content[i][1] = s.getCourseName();
              content[i][2] = s.getCoursePeriod();
              content[i][3] = s.getCourseCredit();
              content[i][4] = s.getCourseTeacher();
              content[i][5] = s.getCourseAddress();
              i++;
          }
          model.setDataVector(content, title);
      }
      //"查询"按钮的事件处理
      public void actionPerformed(ActionEvent e) {
          if(selectAll.isSelected()){
               students = sd.selectAllCourse();
              rows = students.size();
              setTableContent(rows);
          }else if(selectByName.isSelected()){
               students = sd.selectByName(txtName.getText());
               rows = students.size();
          }else if(selectByCredit.isSelected()){
               String strClass = txtCredit.getText();
               students = sd.selectByCredit(strClass);
               rows = students.size();
               }else if(selectByTeacher.isSelected()){
                    students = sd.selectByTeacher(txtTeacher.getText());
                    rows = students.size();
               }
           setTableContent(rows);
       } }
```

(6) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“查询课程”的事件处理。

```
selectCourse.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent arg0) {
            SelectCourse sc = new SelectCourse( );
```

```
            sc.setVisible(true);
    } });
```

## 十二、“添加课程”菜单项功能实现



### 1. 实现思路

当单击“添加课程”菜单项时，首先应该调出如图 3-57 所示的“添加课程”窗口。如果所填信息正确，则调出如图 3-58 所示的“添加成功”消息提示对话框；如果课程编号重复或课程学时、课程学分不是整数，则调出如图 3-59 所示的“添加失败”消息提示对话框。


图 3-57 “添加课程”窗口


图 3-58 “添加成功”消息提示对话框


图 3-59 “添加失败”消息提示对话框

### 2. 代码实现

(1) 修改 CourseDao 类，添加 insertCourse()方法，添加课程信息。

```
public boolean insertCourse(String cid,String cname,String tp,String tc,String tt,String ta){
        boolean bool = false;
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rsResultSet = null;
        DBUtil dbUtil = new DBUtil();
        try{
            con = dbUtil.getConnection();
            pstmt = con.prepareStatement("insert into course values(?,?,?,?,?,?)");
```

```
            pstmt.setString(1, cid);
            pstmt.setString(2, cname);
            pstmt.setInt(3, Integer.parseInt(tp));
            pstmt.setInt(4, Integer.parseInt(tc));
            pstmt.setString(5, tt);
            pstmt.setString(6,ta);
            int n = pstmt.executeUpdate();
            if(n > 0)
                bool = true;
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
        }
        return bool;
    }
```

（2）在 view 包中定义“添加课程”窗口。

```
public class InsertCourse extends JFrame {
  private JPanel jp;
  private JLabel lblCId,lblName,lblPeriod,lblCredit,lblTeacher,lblAddress;
  private JTextField txtCId,txtName,txtPeriod,txtCredit,txtTeacher,txtAddress;
  private JButton btnAdd,cancel;
  public InsertCourse(){
      super("添加课程");
      jp = new JPanel(null);
      lblCId = new JLabel("课程编号:");
      lblName = new JLabel("课程名称:");
      lblPeriod = new JLabel("课程学时:");
      lblCredit = new JLabel("课程学分:");
      lblTeacher = new JLabel("授课老师:");
      lblAddress = new JLabel("上课地点:");
      txtCId = new JTextField(10);
      txtName = new JTextField(10);
      txtPeriod = new JTextField(12);
      txtCredit = new JTextField(12);
      txtTeacher = new JTextField(12);
      txtAddress = new JTextField(10);
      btnAdd = new JButton("添加");
      cancel = new JButton("取消");
      this.add(jp);
      jp.add(lblCId);
      jp.add(txtCId);
      jp.add(lblName);
      jp.add(txtName);
      jp.add(lblPeriod);
      jp.add(txtPeriod);
```

```
jp.add(lblCredit);
jp.add(txtCredit);
jp.add(lblTeacher);
jp.add(txtTeacher);
jp.add(lblAddress);
jp.add(txtAddress);
jp.add(btnAdd);
jp.add(cancel);
lblCId.setBounds(50, 20, 100, 15);
lblName.setBounds(50, 70, 100, 15);
lblPeriod.setBounds(50, 120, 100, 15);
lblCredit.setBounds(50,170,100,15);
lblTeacher.setBounds(50,220,100,15);
lblAddress.setBounds(50,270,100,15);
txtCId.setBounds(150, 20, 210, 21);
txtName.setBounds(150, 70, 210, 21);
txtPeriod.setBounds(150, 120, 210, 21);
txtCredit.setBounds(150,170,210,21);
txtTeacher.setBounds(150,220,210,21);
txtAddress.setBounds(150,270,210,21);
btnAdd.setBounds(70, 330, 100, 21);
cancel.setBounds(240, 330, 100, 21);
txtCId.requestFocus();                    //将光标置于学号文本框
this.setSize(400,400);
setLocationRelativeTo(null);              //居中
setResizable(false);
btnAdd.addActionListener(new ActionListener(){
 public void actionPerformed(ActionEvent e) {
   if("".equals(txtCId.getText())){
    JOptionPane.showMessageDialog(InsertCourse.this.btnAdd, "课程编号不为空");
    txtCId.requestFocus();
   }else if("".equals(txtName.getText())){
    JOptionPane.showMessageDialog(InsertCourse.this.btnAdd, "课程名称不为空");
    txtName.requestFocus();
   }else if("".equals(txtPeriod.getText())){
    JOptionPane.showMessageDialog(InsertCourse.this.btnAdd, "课程学时不为空");
    txtPeriod.requestFocus();
   }else if("".equals(txtCredit.getText())){
    JOptionPane.showMessageDialog(InsertCourse.this.btnAdd, "课程学分不为空");
    txtCredit.requestFocus();
   }else if("".equals(txtTeacher.getText())){
    JOptionPane.showMessageDialog(InsertCourse.this.btnAdd, "授课老师不为空");
    txtTeacher.requestFocus();
   }else if("".equals(txtAddress.getText())){
    JOptionPane.showMessageDialog(InsertCourse.this.btnAdd, "上课地点不为空");
    txtAddress.requestFocus();
   }else {
           CourseDao s = new CourseDao();
           String cid = txtCId.getText();
           String cname = txtName.getText();
           String cp = txtPeriod.getText();
           String cc = txtCredit.getText();
           String ct = txtTeacher.getText();
           String ca = txtAddress.getText();
```

```
                boolean flag = s.insertCourse(cid,cname,cp,cc,ct,ca);
                if(flag)
                    JOptionPane.showMessageDialog(null, "添加成功","提示",
                        JOptionPane.INFORMATION_MESSAGE);
                else
                    JOptionPane.showMessageDialog(null, "添加失败,请检查!课程编号不能重
                    复,课程学时和课程学分是整数","提示",JOptionPane.WARNING_MESSAGE);
                    }} });
        cancel.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                txtCId.setText("");
                txtName.setText("");
                txtPeriod.setText("");
                txtCredit.setText("");
                txtTeacher.setText("");
                txtAddress.setText("");
                txtCId.requestFocus();
            }
      });
    }}
```

(3) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“添加课程”的事件处理。

```
insertCourse.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent arg0) {
        InsertCourse dsc = new InsertCourse();
        dsc.setVisible(true);
} });
```

## 十三、“修改课程”菜单项功能实现

### 1. 实现思路

当单击“修改课程”菜单项时,首先应该调出如图 3-60 所示的“修改课程”窗口。当输入一个正确的课程编号并单击“确定”按钮后,则从数据库中查询该课程的详细信息并在窗口中的相应文本框中显示出来,同时课程编号变为不可编辑状态。当修改完毕单击“修改”按钮时,则调出如图 3-61 所示的“修改成功”消息提示对话框;如果课程学时或课程学分不是整数,调出如图 3-62 所示的“修改失败:课程学时或课程学分不符合要求”消息提示对话框。当输入的课程编号不存在时,则会调出如图 3-63 所示的“课程编号不存在”消息提示对话框。

### 2. 代码实现

(1) 修改 CourseDao 类,添加 selectByCourseId()方法,按课程编号查询课程信息。

```
public Course selectByCourseId(String cid){
    Course c = null;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
        con = dbUtil.getConnection();
```


图 3-60　修改课程信息


图 3-61　“修改成功”消息提示对话框


图 3-62　“修改失败：课程学时或课程学分不符合要求”消息提示对话框


图 3-63　“课程编号不存在”消息提示对话框

```
        pstmt = con.prepareStatement("select * from course where course_id = ? ");
        pstmt.setString(1, cid);
        rs = pstmt.executeQuery();
        if(rs.next()){
            c = new Course();
            c.setCourseId(rs.getString(1));
            c.setCourseName(rs.getString(2));
            c.setCoursePeriod(rs.getInt(3));
            c.setCourseCredit(rs.getInt(4));
            c.setCourseTeacher(rs.getString(5));
            c.setCourseAddress(rs.getString(6));
         }
    }catch(Exception e){
```

```
            e.printStackTrace();
            //JOptionPane.showMessageDialog(null, "访问 course 表失败!");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
        }
     return c;
     }
```

(2) 修改 CourseDao 类,添加 modifyCourse()方法,修改课程信息。

```
public boolean modifyCourse(String cid,String cname,int tp,int tc,String tt,String ta){
  boolean bool = false;
  Connection con = null;
  PreparedStatement pstmt = null;
  DBUtil dbUtil = new DBUtil();
  try{
    con = dbUtil.getConnection();
    pstmt = con.prepareStatement("update course set course_name = ?,course_period = ?,
          course_credit = ?,course_teacher = ?,course_address = ? where course_id = ?");
    pstmt.setString(1, cname);
    pstmt.setInt(2, tp);
    pstmt.setInt(3, tc);
    pstmt.setString(4, tt);
    pstmt.setString(5, ta);
    pstmt.setString(6, cid);
    int n = pstmt.executeUpdate();
    if(n > 0)
     bool = true;
    }catch(Exception e){
     e.printStackTrace();
    }finally{
          try{
              if(pstmt!= null)
                  pstmt.close();
              if(con!= null)
                  con.close();
              }catch( Exception e){
                  e.printStackTrace( );
              }
    }
    return bool;
}
```

(3) 在 view 包中定义“修改课程”窗口。

```
public class ModifyCourse extends JFrame{
  private JPanel jp;
```

```
private JLabel lblid,lblname,lblp,lblc,lblt,lbla;
private JTextField txtid,txtname,txtp,txtc,txtt,txta;
private JButton btnModify,btnCancel,ok;
public ModifyCourse(){
    super("修改课程");
    jp = new JPanel(null);
    lblid = new JLabel("请输入要修改的课程编号");
    lblname = new JLabel("课程名称");
    lblp = new JLabel("课程学时");
    lblc = new JLabel("课程学分");
    lblt = new JLabel("授课老师");
    lbla = new JLabel("上课地点");
    txtid = new JTextField();
    txtname = new JTextField();
    txtp = new JTextField();
    txtc = new JTextField();
    txtt = new JTextField();
    txta = new JTextField();
    btnModify = new JButton("修改");
    btnCancel = new JButton("取消");
    ok = new JButton("确定");
    this.add(jp);
    jp.add(lblid);
    jp.add(lblname);
    jp.add(lblp);
    jp.add(lblc);
    jp.add(lblt);
    jp.add(lbla);
    jp.add(txtid);
    jp.add(txtname);
    jp.add(txtp);
    jp.add(txtc);
    jp.add(txtt);
    jp.add(txta);
    jp.add(btnModify);
    jp.add(btnCancel);
    jp.add(ok);
    ok.setBounds(500, 20, 80, 20);
    lblid.setBounds(50, 20, 150, 15);
    lblname.setBounds(50, 70, 150, 15);
    lblp.setBounds(50, 120, 150, 15);
    lblc.setBounds(50,170,150,15);
    lblt.setBounds(50,220,150,15);
    lbla.setBounds(50,270,150,15);
    txtid.setBounds(250, 20, 210, 21);
    txtname.setBounds(250, 70, 210, 21);
    txtp.setBounds(250, 120, 210, 21);
    txtc.setBounds(250,170,210,21);
    txtt.setBounds(250,220,210,21);
    txta.setBounds(250,270,210,21);
    btnModify.setBounds(100, 320, 100, 21);
    btnCancel.setBounds(260, 320, 100, 21);
    this.setSize(700,450);
    setLocationRelativeTo(null);
```

```
        setResizable(false);
        ok.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e) {
              CourseDao sd = new CourseDao();
              String s1 = txtid.getText();
              Course c = sd.selectByCourseId(s1);
              if(c == null){
                  JOptionPane.showMessageDialog(null, "课程编号不存在!","提示",
                          JOptionPane.INFORMATION_MESSAGE);
              }else{
                  txtname.setText(c.getCourseName());
                  txtp.setText("" + c.getCoursePeriod());
                  txtc.setText("" + c.getCourseCredit());
                  txtt.setText(c.getCourseTeacher());
                  txta.setText(c.getCourseAddress());
                  txtid.setEditable(false);
              }}
      });
      btnModify.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent ex){
           try{CourseDao s = new CourseDao();
              String cid = txtid.getText();
              String cname = txtname.getText();
              int tp = Integer.parseInt(txtp.getText());
              int tc = Integer.parseInt(txtc.getText());
              String tt = txtt.getText();
              String ta = txta.getText();
              boolean i = s.modifyCourse(cid,cname,tp,tc,tt,ta);
              if(i == true)
                JOptionPane.showMessageDialog(null, "修改成功","提示",
                     JOptionPane.INFORMATION_MESSAGE);
              else
              JOptionPane.showMessageDialog(null, "修改失败","提示",
                  JOptionPane.WARNING_MESSAGE);
            }catch(Exception e){
              JOptionPane.showMessageDialog(null, "修改失败:课程学时或课程学分不符合要求",
              "提示",JOptionPane.WARNING_MESSAGE);
          }}});
          btnCancel.addActionListener(new ActionListener(){
              public void actionPerformed(ActionEvent e){
                  txtid.setText("");
                  txtname.setText("");
                  txtp.setText("");
                  txtc.setText("");
                  txtt.setText("");
                  txta.setText("");
                  txtid.setEditable(true);
                  txtid.requestFocus();
              }
          });
  }}
```

(4) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“修改课程”的事件处理。

```
modifyCourse.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent arg0) {
            modifyCourse dsc = new modifyCourse ();
            dsc.setVisible(true);
    } });
```

## 十四、“删除课程”菜单项功能实现

### 1. 实现思路

当单击“删除课程”菜单项时，首先应该调出如图 3-64 所示的“删除课程”窗口。当选择一行并单击“删除”按钮时，则调出如图 3-65 所示的确认对话框，这时如果单击“是”按钮，则调出 “删除成功”或“删除失败”消息提示对话框。


图 3-64 “删除课程”窗口

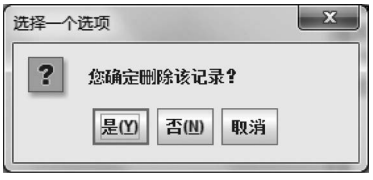



图 3-65 确认对话框

### 2. 代码实现

(1) 修改 CourseDao 类，添加 selectByCourseId()方法，删除指定课程编号的课程信息。

```
public boolean deleteByCourseId(String cid){
        boolean bool = false;
        Connection con = null;
        PreparedStatement pstmt = null;
        try{
            DBUtil dbUtil = new DBUtil();
```

```
            con = dbUtil.getConnection();
            pstmt = con.prepareStatement("delete from course where course_id = ?");
            pstmt.setString(1,cid);
            int n = pstmt.executeUpdate();
            if(n > 0)
                bool = true;
            }catch(Exception e){
                JOptionPane.showMessageDialog(null, "访问 course 表失败!");
            }finally{
                    try{
                        if(pstmt!= null)
                            pstmt.close();
                        if(con!= null)
                            con.close();
                    }catch( Exception e){
                        e.printStackTrace( );
                        }
                }
    return bool;
    }
```

(2) 在 view 包中定义“修改课程”窗口。

```
public class DeleteCourse extends JFrame implements ActionListener{
 private JButton delete;
 private DefaultTableModel model;
 private JTable table;
 private JScrollPane sp_table;
 int rows = 0;
 ArrayList < Course > allCourse = null;
 CourseDao sd = new CourseDao();
 Object content[ ][ ] = null;
 String title[ ] = {"课程编号","课程名称","课程学时","课程学分","授课老师","上课地点"};
 public DeleteCourse(){
   model = new DefaultTableModel();
   table = new JTable(model);
   table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
   sp_table = new JScrollPane(table);
   allCourse = sd.selectAllCourse();
   rows = allCourse.size();
   setTableContent(rows);
   delete = new JButton("删除");
   delete.addActionListener(this);
   this.add(sp_table,BorderLayout.CENTER);
   this.add(delete,BorderLayout.SOUTH);
   this.setTitle("删除课程");
  this.setBounds(100, 100, 550, 500);
  }
public void actionPerformed(ActionEvent e){
    int index[ ] = table.getSelectedRows();
     if(index.length == 0){
         JOptionPane.showMessageDialog(null, "请选择要删除的记录");
     }else{
         int k = JOptionPane.showConfirmDialog(null, "您确定删除该记录?");
             if(k == JOptionPane.YES_OPTION){
                String cid = table.getValueAt(index[0],0).toString();
                boolean bool = sd.deleteByCourseId(cid);
                if(bool){
                  JOptionPane.showMessageDialog(null, "删除成功!");
```

```
                    }else{
                        JOptionPane.showMessageDialog(null, "删除失败!");
                    }     }  }
        allCourse = sd.selectAllCourse();
        rows = allCourse.size();
        setTableContent(rows);
    }
     public void setTableContent(int rows){
             content = new Object[rows][6];
             int i = 0;
             for(Course s:allCourse){
                 content[i][0] = s.getCourseId();
                 content[i][1] = s.getCourseName();
                 content[i][2] = s.getCoursePeriod();
                 content[i][3] = s.getCourseCredit();
                  content[i][4] = s.getCourseTeacher();
                 content[i][5] = s.getCourseAddress();
                 i++;
             }
             model.setDataVector(content, title);
         }
    }
```

(3) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“删除课程”的事件处理。

```
deleteCourse.addActionListener(new ActionListener(){
 public void actionPerformed(ActionEvent arg0) {
     DeleteCourse dsc = new DeleteCourse();
     dsc.setVisible(true);
}});
```

## 十五、“查看个人奖惩信息”菜单项功能实现

### 1. 实现思路

当单击“查看个人奖惩信息”菜单项时,调出如图 3-66 所示的窗口。通过该窗口可以了解自己已有的奖惩信息。


图 3-66 “个人奖惩”窗口

**2. 代码实现**

(1) 定义数据表 prize 的实体类 Prize。

```
public class Prize {
 private int prizeId;
 private String studentId;
 private Date prizeDate;
 private String prizeName;
 private String prizeLevel;
 public Prize(int prizeId, String studentId, Date prizeDate, String prizeName, String prizeLevel) {
     this.prizeId = prizeId;
     this.studentId = studentId;
     this.prizeDate = prizeDate;
     this.prizeName = prizeName;
     this.prizeLevel = prizeLevel;
 }
 public Prize() {
     super();
 }
 public int getPrizeId() {
     return prizeId;
 }
 public void setPrizeId(int prizeId) {
     this.prizeId = prizeId;
 }
 public String getStudentId() {
     return studentId;
 }
 public void setStudentId(String studentId) {
     this.studentId = studentId;
 }
 public Date getPrizeDate() {
     return prizeDate;
 }
 public void setPrizeDate(Date prizeDate) {
     this.prizeDate = prizeDate;
 }
 public String getPrizeName() {
     return prizeName;
 }
 public void setPrizeName(String prizeName) {
     this.prizeName = prizeName;
 }
 public String getPrizeLevel() {
     return prizeLevel;
 }
 public void setPrizeLevel(String prizeLevel) {
     this.prizeLevel = prizeLevel;
 }
 }
```

（2）定义访问数据表 prize 的 Dao 类，在该类中定义方法 selectBySId()，实现通过学生的学号得到该生奖惩信息的功能。

```
public class PrizeDao {
 //通过学生的学号得到该生的奖惩信息
 public ArrayList < Prize > selectBySId(String pid){
            ArrayList < Prize > prizes = new ArrayList < Prize >();
            Connection con = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            DBUtil dbUtil = new DBUtil();
            Prize prize = null;
            try{
                con = dbUtil.getConnection();
                pstmt = con.prepareStatement("select * from prize where student_id = ?");
                pstmt.setString(1, pid);
                rs = pstmt.executeQuery();
                while(rs.next()){
                    prize = new Prize(rs.getString(1),rs.getString(2),rs.getDate(3),
                          rs.getString(4),rs.getString(5));
                    prizes.add(prize);
                }
            }catch(Exception e){
                JOptionPane.showMessageDialog(null, "访问 prize 表失败!");
            }finally{
                try{
                    if(rs!= null)
                        rs.close();
                    if(pstmt!= null)
                        pstmt.close();
                    if(con!= null)
                        con.close();
                    }catch( Exception e){
                        e.printStackTrace( );
                    }
            }
            return prizes;
        }
    }}
```

（3）在 view 包中定义“个人奖惩”窗口。

```
public class DisplaySelfPrize extends JFrame {
    private JTable table;
    private DefaultTableModel model;
    private JScrollPane jsp;
    private JPanel jp;
    ArrayList < Prize > sc = null;
    int rows = 0;
    Object content[ ][ ] = null;
    String title[ ] = {"奖惩编号","学号","奖惩时间","奖惩名称","奖惩级别"};
    public DisplaySelfPrize(User user){
```

```
        jp = new JPanel(new BorderLayout());
        model = new DefaultTableModel();
        table = new JTable(model);
        jsp = new JScrollPane(table);
        this.add(jp);
        jp.add(jsp,BorderLayout.CENTER);
        PrizeDao sd = new PrizeDao();
        sc = sd.SelectById(user.getUserId());
        rows = sc.size();
        setTableContent(rows);
        this.setTitle("个人奖惩");
        this.setSize(600, 400);
        this.setLocationRelativeTo(null);
    }
    private void setTableContent(int rows) {
        content = new Object[rows][5];
        int i = 0;
        for(Prize s1:sc){
            content[i][0] = s1.getPrizeId();
            content[i][1] = s1.getStudentId();
            content[i][2] = s1.getPrizeDate();
            content[i][3] = s1.getPrizeName();
            content[i][4] = s1.getPrizeLevel();
            i++;
    }
        model.setDataVector(content, title);
}
}
```

(4) 修改 MainFrame 窗口,在构造方法中添加如下代码,实现菜单项“查看个人奖惩信息”的事件处理。

```
displaySelfPrize.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      DisplaySelfPrize f = new DisplaySelfPrize(user);
      f.setVisible(true);
    }
});
```

### 十六、“查询 /修改 /删除奖惩信息”菜单项功能实现

#### 1. 实现思路

当单击“查询/修改/删除奖惩信息”菜单项时,应该调出如图 3-67 所示的窗口。为了更便捷地管理学生的奖惩信息,这里将查询、修改、删除功能集于一个窗口中实现。

(1) 查询功能。该窗口默认显示所有学生的奖惩信息。在该窗口中可以实现如图 3-68～图 3-70 所示的按学号查询、按奖惩名称查询以及按奖惩级别查询。

(2) 修改功能。选择要修改的行,双击单元格进行内容修改,修改完后按 Enter 键,然后单击“修改”按钮完成修改功能。

注意,不能修改奖惩编号和学号。

(3) 删除功能。选择要删除的行,然后单击“删除”按钮即可。


图 3-67 “查询/修改/删除奖惩信息”窗口


图 3-68 按学号查询

## 2. 代码实现

(1) 修改 PrizeDao 类,添加 getAllPrizes()方法,查询所有奖惩信息。

```
public ArrayList < Prize > getAllPrizes(){
    ArrayList < Prize > prizes = new ArrayList < Prize >();
```


图 3-69　按奖惩名称查询


图 3-70　按奖惩级别查询

```
Connection con = null;
PreparedStatement pstmt = null;
ResultSet rs = null;
DBUtil dbUtil = new DBUtil();
Prize prize = null;
```

```
        try{
            con = dbUtil.getConnection();
            pstmt = con.prepareStatement("select * from prize");
            rs = pstmt.executeQuery();
            while(rs.next()){
                prize = new Prize(rs.getString(1),rs.getString(2),rs.getDate(3),
                    rs.getString(4),rs.getString(5));
                prizes.add(prize);
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, "访问 prize 表失败!");
        }finally{
            try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
                }catch( Exception e){
                    e.printStackTrace( );
                    }
        }
        return prizes;
    }
```

(2) 修改 PrizeDao 类,添加 selectByName()方法,按奖惩名称查询。

```
public ArrayList < Prize > selectByName(String name) {
    ArrayList < Prize > prizes = new ArrayList < Prize >();
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    Prize prize = null;
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from prize where prize_name = ?");
        pstmt.setString(1,name);
        rs = pstmt.executeQuery();
        while(rs.next()){
            prize = new Prize(rs.getString(1),rs.getString(2),rs.getDate(3),
                rs.getString(4),rs.getString(5));
            prizes.add(prize);
        }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 prize 表失败!");
    }finally{
        try{
            if(rs!= null)
                rs.close();
            if(pstmt!= null)
                pstmt.close();
            if(con!= null)
                con.close();
```

```
            }catch( Exception e){
                e.printStackTrace( );
        }
    }
    return prizes;
}
```

(3) 修改 PrizeDao 类,添加 selectByLevel()方法,按奖惩级别查询。

```
public ArrayList < Prize > selectByLevel(String level) {
    ArrayList < Prize > prizes = new ArrayList < Prize >();
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    Prize prize = null;
    try{
        con = dbUtil.getConnection();
        pstmt = con.prepareStatement("select * from prize where prize_level = ?");
        pstmt.setString(1,level);
        rs = pstmt.executeQuery();
        while(rs.next()){
            prize = new Prize(rs.getString(1),rs.getString(2),rs.getDate(3),
                rs.getString(4),rs.getString(5));
            prizes.add(prize);
        }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, "访问 prize 表失败!");
    }finally{
        try{
                if(rs!= null)
                    rs.close();
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }
    }
    return prizes;
}
```

(4) 修改 PrizeDao 类,添加 modifyPrize()方法,按奖惩编号和学号修改获奖信息。

```
public boolean modifyPrize(int pid,String sid,Date pd,String pn,String pl){
    boolean bool = false;
    Connection con = null;
        PreparedStatement pstmt = null;
        DBUtil dbUtil = new DBUtil();
        try{
            con = dbUtil.getConnection();
            pstmt = con.prepareStatement("update prize set prize_date = ?,prize_name = ?,
                        prize_level = ? where prize_id = ? and student_id = ?");
            pstmt.setDate(1,pd);
            pstmt.setString(2,pn);
```

```
            pstmt.setString(3,pl);
            pstmt.setInt(4,pid);
            pstmt.setString(5,sid);
            int n = pstmt.executeUpdate();
            if(n > 0)
                bool = true;
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            try{
                if(pstmt!= null)
                    pstmt.close();
                if(con!= null)
                    con.close();
            }catch( Exception e){
                e.printStackTrace( );
                }     }
        return bool;
        }
```

(5) 修改 PrizeDao 类,添加 deleteByPId()方法,按编号删除奖惩记录。

```
public boolean deleteByPId(int Pid){
    boolean bool = false;
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    DBUtil dbUtil = new DBUtil();
    try{
      con = dbUtil.getConnection();
      pstmt = con.prepareStatement("delete from PRIZE where prize_id = ? ");
      pstmt.setInt(1, Pid);
      int n = pstmt.executeUpdate();
      if(n > 0)
          bool = true;
      }catch(Exception e){
          JOptionPane.showMessageDialog(null, "访问 prize 表失败!");
      }finally{
          try{
              if(rs!= null)
                  rs.close();
              if(pstmt!= null)
                  pstmt.close();
              if(con!= null)
                  con.close();
              }catch( Exception e){
                  e.printStackTrace( );
          }
      }
       return bool;
}
```

(6) 在 view 包中定义“查询/修改/删除奖惩信息”窗口。

```
public class selectModifyDeletePrize extends JFrame {
      private JPanel jp1,jp11,jp12,jp2,jp3;
```

```
private JRadioButton selectById,selectByName,selectByLevel,selectAll;
private ButtonGroup bg;
private JLabel lblSId,lblName,lblLevel;
private JTextField txtSId;
private JTextField name,level;
private DefaultTableModel model;
private JTable table;
private JScrollPane sp;
private JButton ok;                          //查询按钮
private JButton modify;                      //修改按钮
private JButton del;                         //删除按钮
int rows = 0;
ArrayList < Prize > prizes = null;
PrizeDao sd = new PrizeDao();
Object content[ ][ ] = null;
String title[ ] = {"奖惩编号","学号","奖惩时间","奖惩名称","奖惩级别"};
public selectModifyDeletePrize(){
    setTitle("查询/修改/删除奖惩信息");
    jp1 = new JPanel();
    jp1.setLayout(new GridLayout(2,1));
    jp11 = new JPanel(new GridLayout(1,5));
    jp12 = new JPanel();
    jp2 = new JPanel();
    jp3 = new JPanel();
    jp3.setBorder(new EmptyBorder(5,5,5,5));
    jp3.setLayout(new BorderLayout(0,0));
    bg = new ButtonGroup();
    selectAll = new JRadioButton("查询所有奖惩",true);
    selectById = new JRadioButton("按学号查询");
    selectByName = new JRadioButton("按奖惩名称查询");
    selectByLevel = new JRadioButton("按奖惩级别查询");
    bg.add(selectAll);
    bg.add(selectById);
    bg.add(selectByName);
    bg.add(selectByLevel);
    ok = new JButton("查询");
    modify = new JButton("修改");
    del = new JButton("删除");
    jp11.add(selectAll);
    jp11.add(selectById);
    jp11.add(selectByName);
    jp11.add(selectByLevel);
    jp11.add(ok);
    jp1.add(jp11);
    lblSId = new JLabel("请输入学号:");
    txtSId = new JTextField(20);
    lblLevel = new JLabel("请输入奖惩级别:");
    lblName = new JLabel("请输入奖惩名称");
    level = new JTextField(20);
    name = new JTextField(20);
    jp12.add(lblSId);
    jp12.add(txtSId);
    jp12.add(lblLevel);
    jp12.add(level);
```