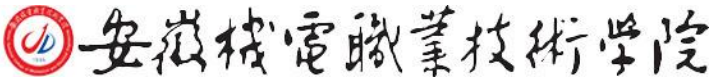

《Java 开发项目实训》实训报告

学生信息管理系统

姓　　名：郝宏鑫

班　　级：软件 3194 班

小组名称：PureK1t Dev

小组成员：郝宏鑫

指导教师：汪学文　汪强

完成日期：2020 年 11 月 25 日

信息工程系课程实训评价表

自我评价：

对项目的整体把控能力较弱，Java 基础还比较薄弱，缺乏对 GUI 窗体程序的界面布局经验。

通过本次实训我学到了很多同时也发现了自己的很多不足，在今后的学习生活中还需巩固基础知识。

教师评价：

成绩（等级）		教师签名		日期	2020-11-26

摘　要

目前学校工作繁杂、资料众多，虽然各类信息管理系统已进入高校，但还未普及，而对于学生信息管理来说目前还没有一套完整的、统一的系统。因此开发一套适合大众的兼容性好的系统是很有必要的。

刚好最近学习到了有关的 Java AWT & Swing GUI 窗体程序设计的知识我萌生了利用自己所学的知识实现一个简约、美观、跨平台的学生信息管理系统。

本系统依据开发要求主要应用于教育系统，完成对日常的教育工作中学生基本信息的数字化管理。开 2 发本系统可使教职员工减轻工作压力，比较系统的对教务、教学上的各项服务和信息化管理，同时，可以减少劳动力的使用，加快查询速度、加强管理，以及国家各部门的关于信息化的步伐，使各项管理更加规范化。

关键词：Java、Swing、AWT、学生信息管理

目录

第 1 章 需求分析

1.1 需求描述

系统分析是开发管理信息系统的关键性阶段，是一个从不断认识和逐步细化的过程，是下一阶段的工作基础，是为下一阶段进行物理方案设计、解决“怎么做”提供依据，基关键性主要体现在“理解需求”和“表达需求”两方面。

通过对系统的整体分析，我们应该更能理解系统所要实现的功能，所需的环境支持，并提出这些需求的实现条件以及需求应达到的标准，也就是确定新系统要做什么，做到什么程度。

本项目的系统功能需求是:本项目面向用户是学校的管理人员，所以对计算机的人性化和易用性比较高，应用于学校学生信息管理，总体任务是实现学生信息系关系的系统化、规范化和自动化，其主要任务是对学生的各种日常信息进行日常管理，如查询、修改、增加、删除，还考虑到了用户修改密码等，做到界面简单易懂，容易操作，提高了管理效率以及提升了学生信息的安全性和完整性。

1.2 功能描述

最终计划实现以下功能:

1）注销和退出功能
   点击注销按钮注销用户状态调起登录验证重新登录
   点击退出按钮可以退出软件
2）信息加载
   点击按钮对学生数据进行加载，获取到数据库的学生信息
3）学生信息的录入
   有关学生信息的录入，包括输入学生基本信息、学号、班级、联系方式等
4）学生信息的修改
   有关学生信息的修改，有可以修改项目和不可修改项目等
5）学生信息的查询
   利用学生姓名进行查询其他的基本信息
6）关于页面
   显示软件的图标、软件名称、版本号、版权信息、作者等

第 2 章 总体设计

2.1 设计概述

学生信息管理系统的主窗体设计是建立在 Java Swing 的 JFrame 基础上，信息管理页面使用 JDialog 控件进行设计由主窗体提供的 Menu 学生管理菜单跳转触发，关于页面使用 JDialog 控件进行设计由主窗体提供的 Menu 帮助菜单点击跳转触发，另外为了使设计更合理添加了修改管理员用户密码的功能，修改密码页面使用了由主窗体提供的 Menu 管理员菜单跳转触发。

学生信息的显示使用了 JTable 控件进行设计，并且使用了自定义表格模型进行数据的显示处理，因为要处理很多按钮事件大量的使用到了 JOptionPane 控件用来处理事件判定信息，因为没有找到专门用来显示图片的控件，选择变通的使用了 JLabel 通过设置标签 ICON 的方式进行图片处理。

对于关于页面也使用了相同的设计方式，通过对图标和文字的合理排版呈现出简约而不失美观的风格。没有过多杂乱的无关信息，只显示用户关心的内容，所见即所得。

2.2 软件设计分析

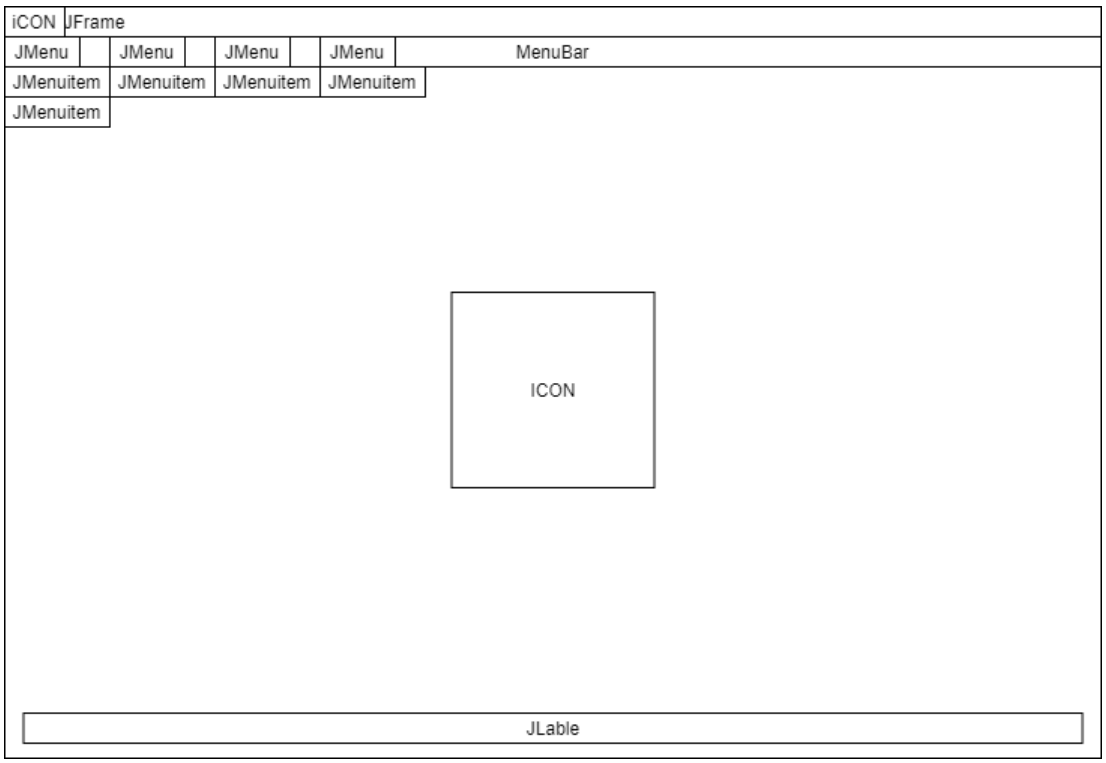

图 2.2.1

1） 主窗体使用 JFrame 进行构建，以及一些用于用户交互的控件（如图 2.2.1 所示）

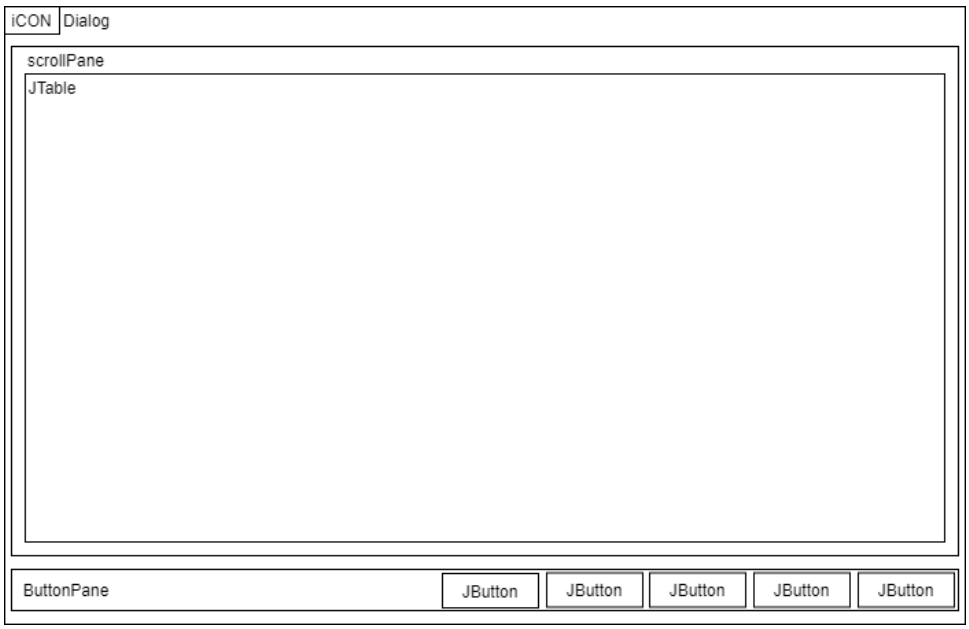

图 2.2.2

2） 信息管理窗体使用 JDialog 控件进行设计主要功能就是用于处理信息显示学生信息，窗体设计简洁大方、直观。（如图 2.2.2）

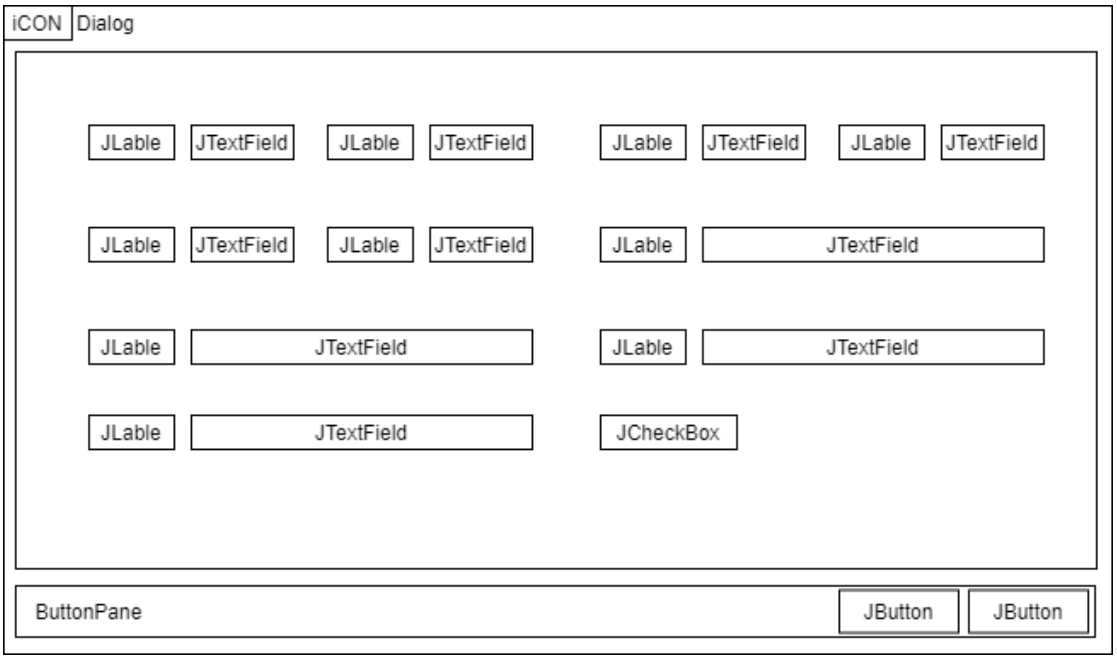

图 2.2.3

3) 修改信息和录入信息窗口设计使用了 JDialog 控件进行构建，学生的在籍状态利用 JCheckBox 复选框控件实现用于给用户提供更好的交互功能。（如图 2.2.3）

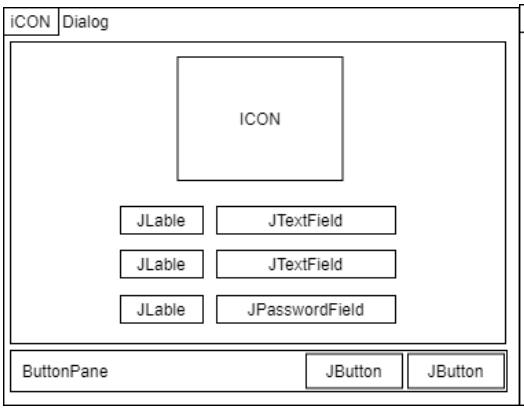

图 2.2.4

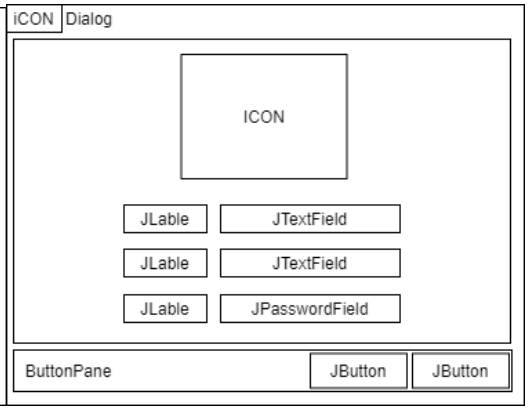

图 2.2.5

4) 用户登录和用户修改密码都是利用 JDialog 控件进行设计，我们在窗体设计中和密码安全的考虑使用了 JPasswordField 控件用于密码的输入以保证数据安全。（如图 2.2.4、2.2.5）

第 3 章 界面设计

3.1 主界面设计

1）主界面设计如：图 3.1、图 3.2、图 3.3、图 3.4、图 3.5、图 3.6

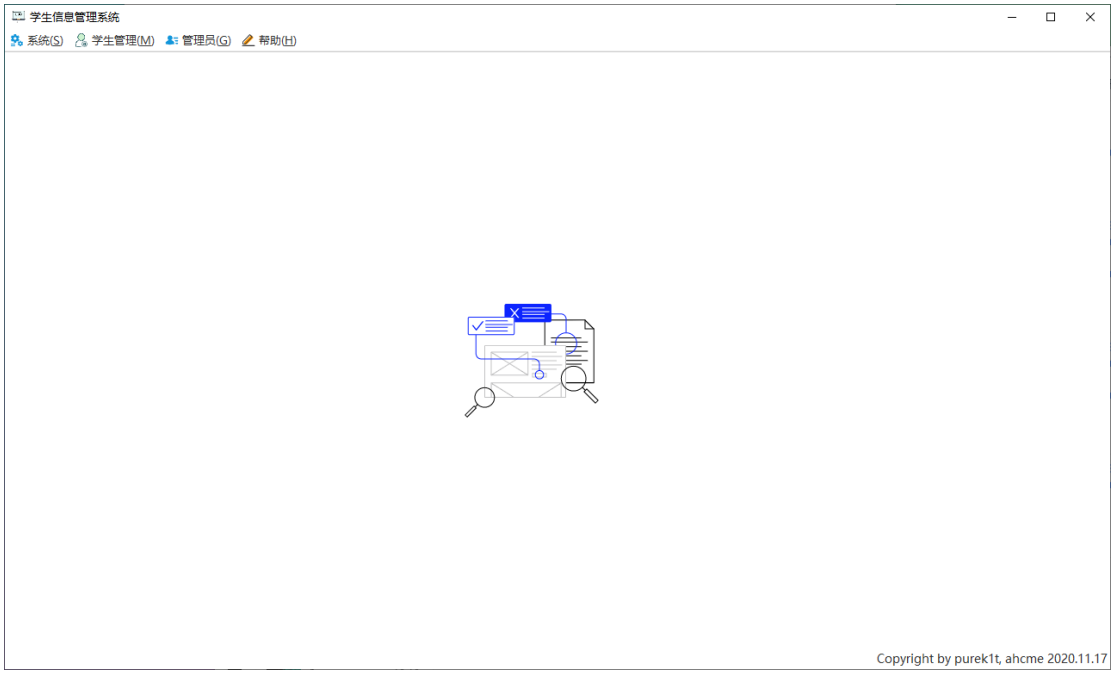

图 3.1 程序主窗体(StudentsFrame)

图 3.2 学生信息管理窗体(StudentMessageDialog(加载数据前))

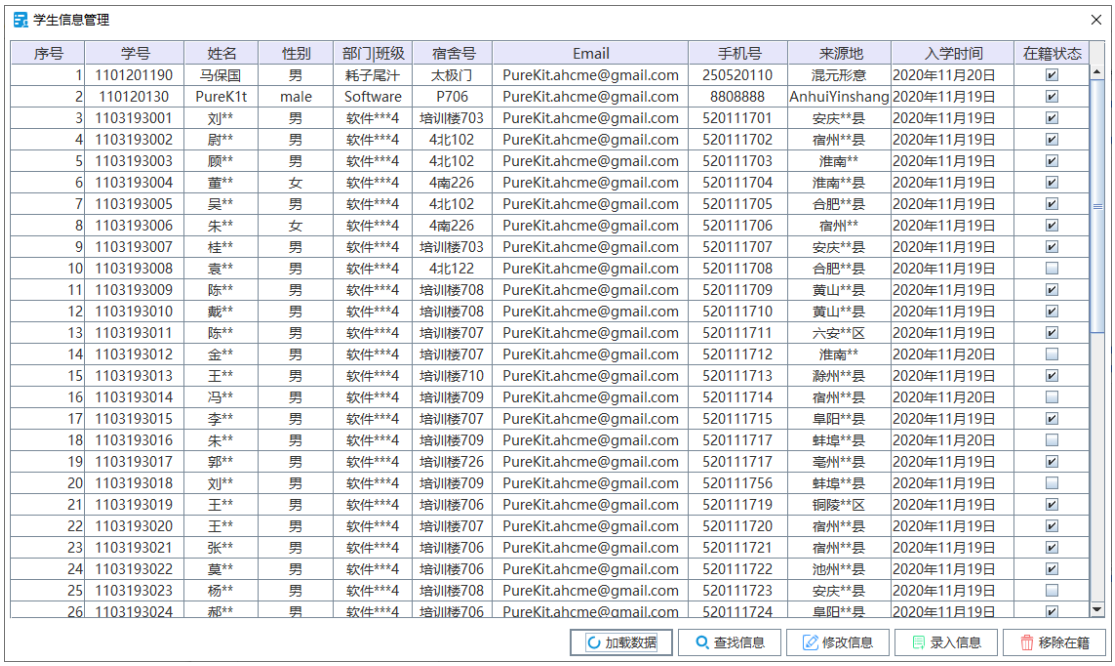

图 3.3 学生信息管理窗体(StudentsMessageDialog(加载数据后))

图 3.4 修改学生信息窗体(EditStudentsMessageDialog)

图 3.5 录入学生信息(InsertStudentsMessageDialog)

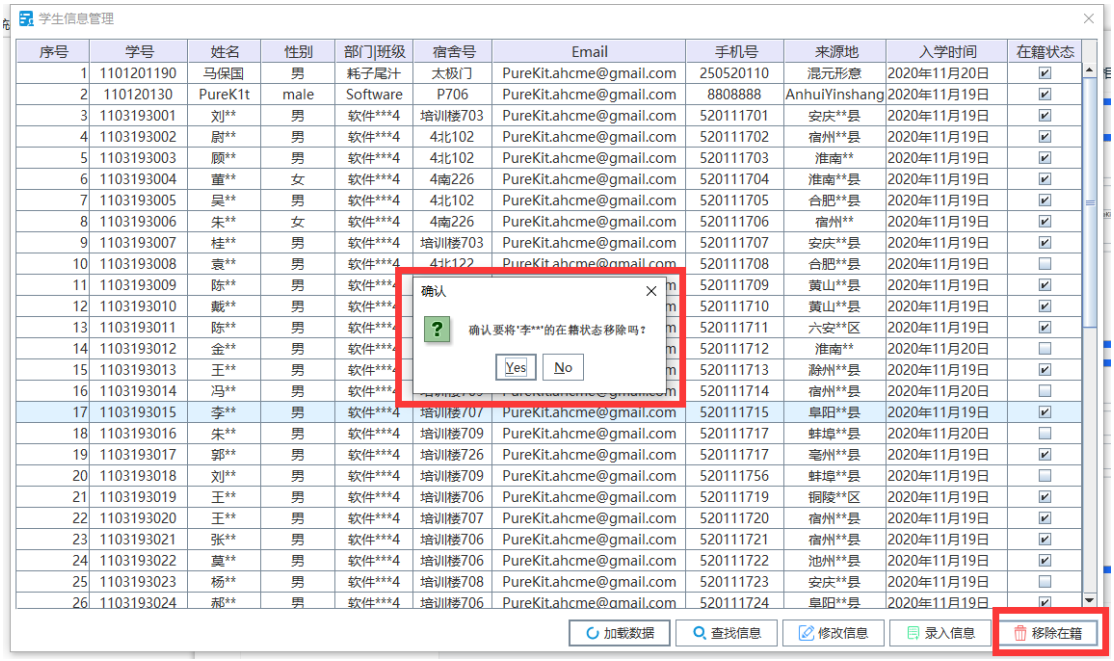

图 3.6 移除在籍(JOptionPane.showConfirmDialog)

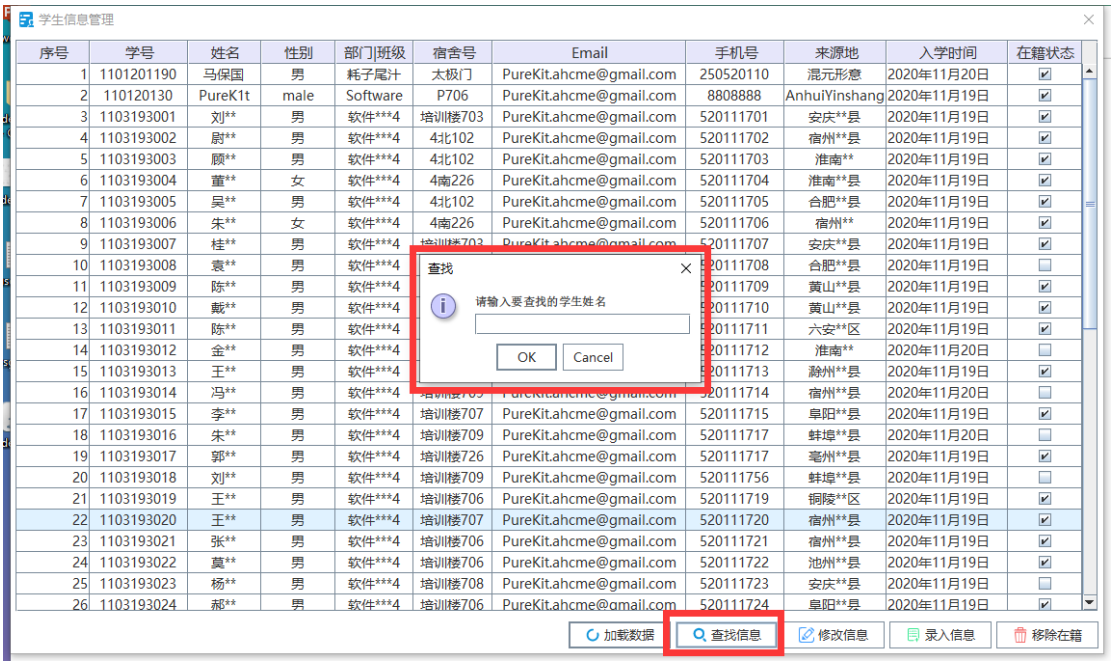

图 3.7 查找学生信息窗口(JOptionPane.showInputDialog)

图 3.8 用户修改密码窗体(EditAdminPasswordDialog)

图 3.9 用户登录窗体(LoginDialog)

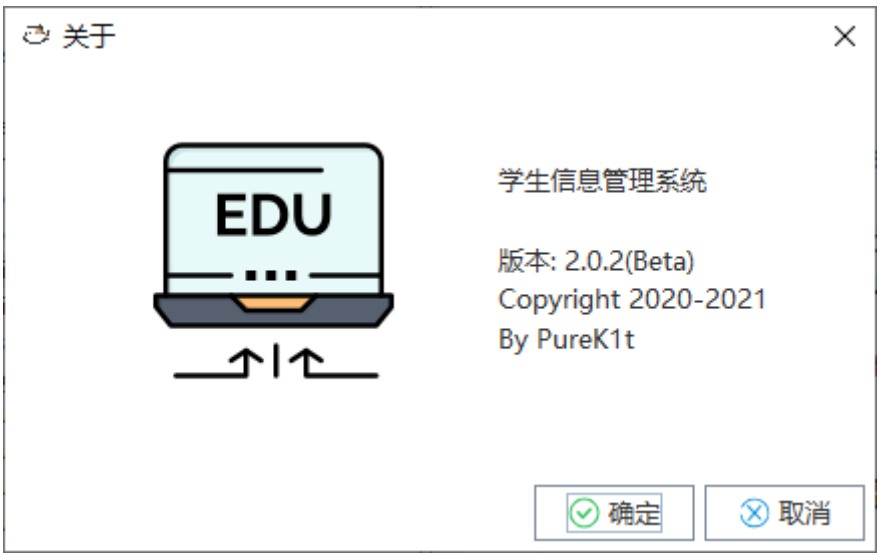

图 3.10 关于页面(AboutDialog)

第 4 章 详细设计

4.1 程序流程图

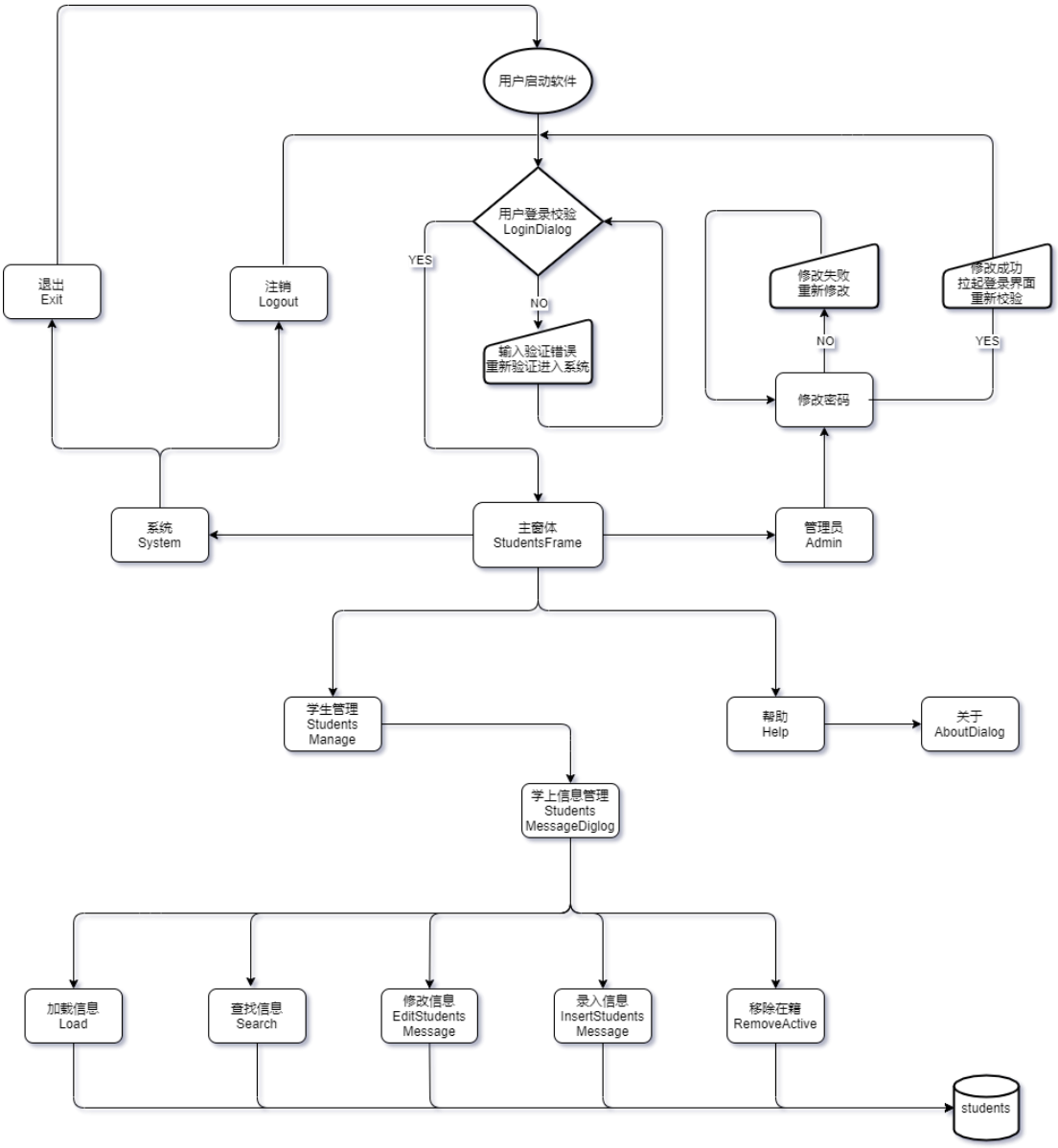

图 4.1.1

4.2 程序模块设计

为了学生信息管理相关功能，程序一共设计了 11 个类其中 StudentsFrame、StudentsMessageDialog、EditStudentsMessageDialog、InsertStudentsMessageDialog、LoginDialog、EditAdminPasswordDialog、AboutDialog 类主要提供给界面 UI 部分的展示，分别基于 JFrame 和 JDialog 类，它们也是用户能直接接触的部分。

创建 DbUtil 类提供 getStudentsMessage()、updateAdminPassword()、getLoginUser()、updateStudentsMessage()等方法用于处理数据。创建 getConnection()方法用于调用创建连接数据库方法，每次使用连接数据库处理完成之后都会随之调用 close()关闭连接以防占用通道，因为 MySQL 连接速度很快几乎可以忽略不计没有必要保持一直连接继而实现处理学生信息的修改、增加、删除、查找等。

使用 AbstractTableModel 方法实现自定义表格模型，AbstractTableModel 是一个抽象类，这个类帮我们实现大部份的 TableModel 方法，除了 getRowCount(),getColumnCount()，getValueAt()方法等，利用这个抽象类就可以设计出不同格式的表格。

创建实体类 StudentsMessage 使用 Eclipse 的 source＼getter＆setter 工具生成各个字段对应的 get 属性和 set 属性，创建实体类主要目的是存储数据并提供对这些数据的访问，用于获取数据和处理数据。

多处使用 JOptionPane 控件弹出一个标准对话框，提示用户输入值或通知某些信息，给用户更直观的提示。

使用加密算法密码进行加密后存储，在此项目使用 SHA1 加密算法对密码进行加密，同时配合 JPasswordField 控件使用，提高用户数据的安全性。

同时也在数据库 DbUtil 类中使用了 PreparedStatement 对象，PreparedStatemen 是一个特殊的 Statement 对象，可读性和维护性更好，安全性更好，使用 PreparedStatement 能够有效的预防 sql 注入，提高数据的安全性。

主要模块：

1) getStudentsMessage()　　获取 Students_message 表的数据

2) updateAdminPassword()　　修改 Students_message 表的数据

3) insertStudentsMessage()　插入录入学生信息到 Students_message 表

4) updateSutdentsMessageActive()　移除学生在籍状态

5) getLoginUser()　　获取 Admin 表中的数据进行用户登录校验

6) updateAdminPassword()　　使用 getLoginUser()获取验证数据在进行密码修改

7) getStudentsMessage()　　获取一条学生信息到修改 Dialog 对话框

第 5 章 代码设计

5.1 实现数据库的连接

1) 使用 JDBC 连接数据库

```
//注册 MySQL 驱动使用 JDBC 连接数据库    (DbUtil.java)
public class DbUtil {
   private static final String url = "jdbc:mysql://localhost/students?serverTimezone=PRC";
   private static final String user = "root";
   private static final String password = "root";
```

2) 创建获取连接数据库的方法

```
//获取数据库连接的方法   (DbUtil.java)
public static Connection getConnection() throws SQLException {
   return DriverManager.getConnection(url,user,password);
}
```

3) 创建关闭数据库连接的方法

```
//关闭数据库连接的方法      (DbUtil.java)
public static void Close(Connection conn) throws SQLException {
   if (conn!=null) {
      conn.close();
   }
}
```

5.2 加载数据的实现

1）创建 getStudentsMessage()方法获取 Students_message 表的数据

```
//获取 studentsMessage 的数据（DbUtil.java）
public static Object[][] getStudentsMessage() throws SQLException {
   Connection conn = DbUtil.getConnection();
   //建立 sql 查询语句，获取查询结果
   String sql =  "select  id,students_id, name, sex, "
         + " department, room_id, email, "
         + " phone_id, address, start_school,"
         + " active from students_message";
   PreparedStatement pstmt = conn.prepareStatement(sql,
         ResultSet.TYPE_SCROLL_INSENSITIVE,
         ResultSet.CONCUR_READ_ONLY);
   ResultSet rs = pstmt.executeQuery();//预处理查询语句
   //获取当前集合的行数，定义数组大小
   rs.last();
   Object[][] studentsMessage = new Object[rs.getRow()][];
   rs.beforeFirst();
   int row = 0;
   while(rs.next()) {
      Object[] stu = new Object[] {
            rs.getInt("id"),
            rs.getString("students_id"),
            rs.getString("name"),
            rs.getString("sex"),
            rs.getString("department"),
            rs.getString("room_id"),
            rs.getString("email"),
            rs.getString("phone_id"),
            rs.getString("address"),
            rs.getDate("start_school"),
            rs.getInt("active")==1?true:false
      };
      //每读取一行 ++
      studentsMessage[row++] = stu;
   }
   //关闭数据库
   DbUtil.Close(conn);
   return studentsMessage;
}
```

2）使用 AbstractTableModel 方法实现自定义表格模型

```
//创建自定义表格模型 （StudetnsMessageTableModel.java）
public class StudentsMessageTableModel extends AbstractTableModel{
   private String[] ColumnNames = { "序号","学号","姓名","性别",
           "部门|班级","宿舍号","Email", "手机号", "来源地","入学时间","在籍状态"};
   private Object[][] studentsMessage ;


   public StudentsMessageTableModel() {
      try {
         this.studentsMessage = DbUtil.getStudentsMessage();
      } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }
```

3) 使用创建的自定义表格模型显示数据

```
//使用自定义表格模型显示Students_message 表的数据      （StudentsMessageDialog.java）
private void bindStudentsMessageTableModel() {

   StudentsMessageTableModel model = new StudentsMessageTableModel(); //利用new
实例化
   tblStudentsMessage.setModel(model);
    //设置各列宽度
   tblStudentsMessage.getColumnModel().getColumn(0).setPreferredWidth(10);
    tblStudentsMessage.getColumnModel().getColumn(1).setPreferredWidth(40);
    tblStudentsMessage.getColumnModel().getColumn(2).setPreferredWidth(10);
    tblStudentsMessage.getColumnModel().getColumn(3).setPreferredWidth(5);
    tblStudentsMessage.getColumnModel().getColumn(4).setPreferredWidth(20);
    tblStudentsMessage.getColumnModel().getColumn(5).setPreferredWidth(20);
    tblStudentsMessage.getColumnModel().getColumn(6).setPreferredWidth(140);
    tblStudentsMessage.getColumnModel().getColumn(7).setPreferredWidth(40);
    tblStudentsMessage.getColumnModel().getColumn(8).setPreferredWidth(45);
```

```
tblStudentsMessage.getColumnModel().getColumn(9).setPreferredWidth(65);
tblStudentsMessage.getColumnModel().getColumn(10).setPreferredWidth(1);
//设置单元格字体
tblStudentsMessage.setFont(new Font("Microsoft YaHei UI", Font.PLAIN, 14));
//设置表格内容居中
DefaultTableCellRenderer r= new DefaultTableCellRenderer();
r.setHorizontalAlignment(JLabel.CENTER);
tblStudentsMessage.setDefaultRenderer(Object.class, r);
//设置选中行的颜色
tblStudentsMessage.setSelectionBackground(new Color(224, 242, 255));
//设置行高
tblStudentsMessage.setRowHeight(22);
//获取表头 更改表头标题字体
JTableHeader tab_header = this.tblStudentsMessage.getTableHeader();
tab_header.setFont(new Font("Microsoft YaHei UI", Font.PLAIN, 15));
//更改标题的颜色
tab_header.setBackground(new Color(230 ,230 ,250));
```

4) 加载数据按钮事件绑定

```
//  (StudentsMessageDialog.java)
            public void actionPerformed(ActionEvent e) {
//              bindTableModel();
                //绑定自定义表格模型
            bindStudentsMessageTableModel();
            }
```

5.3 修改学生信息功能实现

1) 创建修改学生信息 updateStudentsMessage()方法

```
//更新 修改 保存数据库  （DbUtil.java）
public  static boolean updateStudentsMessage(StudentsMessage s) throws
SQLException {
   //建立数据库连接
   Connection conn = DbUtil.getConnection();
   //定义语句 update
   String sql = "update students_message set  students_id=?,"
         + " name=?, sex=?, department=?, room_id=?,"
         + " email=?, phone_id=?, address=?, active=?"
         + " where id=?";
   PreparedStatement pstmt = conn.prepareStatement(sql);
   pstmt.setString(1, s.getStudentsID());
   pstmt.setString(2, s.getName());
   pstmt.setString(3, s.getSex());
   pstmt.setString(4, s.getDepartment());
   pstmt.setString(5, s.getRoomID());
   pstmt.setString(6, s.getEmail());
   pstmt.setString(7, s.getPhoneID());
   pstmt.setString(8, s.getAddress());
   pstmt.setBoolean(9,s.isActive());
   pstmt.setInt(10, s.getId());
   //更新数据表
   int sucess = pstmt.executeUpdate();
   //关闭数据库
   DbUtil.Close(conn);
   //判断更新是否成功
   return sucess>0;
}
```

2) 创建 showStudentsMessage 方法将 StudentsMessage 信息显示在修改窗体的文本框控件中等

```
//显示选中数据据到修改窗体     （EditStudentsMessageDialog.java）
private void showStudentsMessage() {
   // TODO Auto-generated method stub
   if(studentsMessage != null ) {
      tbID.setText(String.format("%d", this.studentsMessage.getId()));
      tbStudentsID.setText(studentsMessage.getStudentsID());
```

```
        tbName.setText(studentsMessage.getName());
        tbSex.setText(studentsMessage.getSex());
        tbDepartment.setText(studentsMessage.getDepartment());
        tbRoomID.setText(studentsMessage.getRoomID());
        tbEmail.setText(studentsMessage.getEmail());
        tbPhoneID.setText(studentsMessage.getPhoneID());
        tbAddress.setText(studentsMessage.getAddress());
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

tbStartSchool.setText(sdf.format(studentsMessage.getStartSchoolDate()));
        chkActive.setSelected(this.studentsMessage.isActive());
    }

}
```

3) 修改学生信息的按钮事件绑定

```
//更新修改学生信息      (StudentsMessageDialog.java)
public void actionPerformed(ActionEvent e) {
    if(tblStudentsMessage.getSelectedRowCount()<=0) {
        JOptionPane.showMessageDialog(null, "请选择一个学生");
        return ;
    }
    //获取当前行的编号
   int id = (int)tblStudentsMessage.getValueAt(
          tblStudentsMessage.getSelectedRow(),
          0);
    EditStudentMessageDialog dialog = new EditStudentMessageDialog(id);
    dialog.setModal(true);
    dialog.setVisible(true);
    if (dialog.okReturn()) {
        // 更新JTable 数据
      bindStudentsMessageTableModel();
    }
}
```

5.4 录入学生信息功能实现

1) 在 DbUtil 中创建 InsertStudentsMessage 方法、

```
//插入学生信息记录     （DbUtil.java）
public static boolean insertStudentsMessage(StudentsMessage s) throws
SQLException {
   Connection conn = DbUtil.getConnection();
   String sql = "insert into students_message(students_id,name,sex,"
         + " department,room_id,email, phone_id,address, active)"
         + " values(?,?,?,?,?,?,?,?,?)";
   PreparedStatement pstmt = conn.prepareStatement(sql);
   pstmt.setString(1, s.getStudentsID());
   pstmt.setString(2, s.getName());
   pstmt.setString(3, s.getSex());
   pstmt.setString(4, s.getDepartment());
   pstmt.setString(5, s.getRoomID());
   pstmt.setString(6, s.getEmail());
   pstmt.setString(7, s.getPhoneID());
   pstmt.setString(8, s.getAddress());
   pstmt.setInt(9, s.isActive()?1:0);
   int success = pstmt.executeUpdate();
   DbUtil.Close(conn);
   return success>0;
}
```

2) 确定保存按钮事件绑定

```
//录入学生信息     （InsertStudentsMessageDialog.java）
public void actionPerformed(ActionEvent e) {
   //保存客户信息
  setStudentsMessage();

   try {
      if(DbUtil.insertStudentsMessage(studentsMessage)) {
         JOptionPane.showMessageDialog(InsertStudentsMessageDialog.this,
               " 录入学生信息成功","成功",
               JOptionPane.INFORMATION_MESSAGE);

      okReturn =  true;
```

```
            InsertStudentsMessageDialog.this.dispose();
               return ;
            }
            else {
               JOptionPane.showMessageDialog(InsertStudentsMessageDialog.this,
                     "录入学生信息失败！","失败",
                     JOptionPane.ERROR_MESSAGE);
            }
         } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
         }
         JOptionPane.showMessageDialog(InsertStudentsMessageDialog.this,
               "更新 students 表失败！");
      }
```

3) 录入信息按钮事件绑定

```
//录入学生信息    （StudensMessageDialog.java）
public void actionPerformed(ActionEvent e) {
   //实例化对象 执行事件
   InsertStudentsMessageDialog dialog = new InsertStudentsMessageDialog();
   dialog.setModal(true);
   dialog.setVisible(true);
   if(dialog.okReturn()) {
      //更新JTable 数据
     bindStudentsMessageTableModel();

   }
}
```

5.4 移除学生在籍状态功能实现

1) 在 DbUtil 中创建 updateStudentsMessageActive()方法

```
//删除学生学籍 在籍状态   (DbUtil.java)
public static boolean updateSutdentsMessageActive(int id, boolean  isActive)
      throws SQLException {
   Connection conn = getConnection();
   String sql = "update students_message set active=? "
         + " where id=?";
   PreparedStatement pstmt = conn.prepareStatement(sql);
   pstmt.setBoolean(1, isActive);
   pstmt.setInt(2, id);
   //更新数据表
   int success = pstmt.executeUpdate();
   //关闭数据库
   conn.close();
   //判断更新是否成功并且返回
   return success>0;
}
```

2) 移除在籍按钮事件绑定

```
//移除选中学生的在籍状态   (StudentsMessageDialog.java)
public void actionPerformed(ActionEvent e) {
   if(tblStudentsMessage.getSelectedRowCount()<=0) {
      JOptionPane.showConfirmDialog(StudentsMessageDialog.this, "请先选择一个学
生信息",
            "提示",
            JOptionPane.INFORMATION_MESSAGE);
      return;
   }
   //获取当前行的学生信息编号
   int row = tblStudentsMessage.getSelectedRow();
   int id = (int)tblStudentsMessage.getValueAt(row, 0);
   String studentName = (String)tblStudentsMessage.getValueAt(row, 2);
   int sel = JOptionPane.showConfirmDialog(StudentsMessageDialog.this,
         "确认要将'"+studentName+"'的在籍状态移除吗？",
         "确认", JOptionPane.YES_NO_OPTION);
   if(sel == JOptionPane.YES_OPTION) {
```

```
            try {
                DbUtil.updateSutdentsMessageActive(id, false);
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
                JOptionPane.showConfirmDialog(StudentsMessageDialog.this,
                        "移除学生在籍状态失败！", "失败",
                        JOptionPane.ERROR_MESSAGE);
                return;
            }
        }
        //更新数据
        tblStudentsMessage.setValueAt(false, row, 10);
        bindStudentsMessageTableModel();
    }
```

5.5 用户登录功能实现

1) 在 DbUtil 创建 getLoginUser 方法查询 admin 表对登录进行校验

```
//判断给定用户名和密码能否登录     （DbUtil.java）
//能登录返回User 对象 不能登录返回null
public static User getLoginUser(String username, String password)
      throws SQLException {
   User = null;
   Connection conn = getConnection();
   String sql = "select admin_id, admin_name, admin_email, create_date,"
         + " admin_active, username from admin"
         + " where username=? and `password` = sha(?)";
   PreparedStatement pstmt = conn.prepareStatement(sql);
   pstmt.setString(1, username);
   pstmt.setString(2, password);
   ResultSet rs = pstmt.executeQuery();
   if(rs.next()) {
      user = new User();
      user.setAdminID(rs.getInt("admin_id"));
      user.setAdminName(rs.getString("admin_name"));
      user.setAdminEmail(rs.getString("admin_email"));
      user.setAdminCreateDate(rs.getDate("create_date"));
      user.setAdminActive(rs.getInt("admin_active"));
      user.setUserName(rs.getString("username"));
   }
   //关闭数据库连接
  conn.close();
   return user;
   }
```

2) 登录确定按钮事件绑定

```
//登录校验      (LonginDialog.java)
public void actionPerformed(ActionEvent e) {
   //获取内容到user
   //验证用户登录 对比数据库表 数据
  String userName = tbUserName.getText();
   String password = new String(tbPassword.getPassword());
   try {

      user = DbUtil.getLoginUser(userName, password);
   } catch (SQLException e1) {
```

```
            JOptionPane.showConfirmDialog(LoginDialog.this,
                    "访问 students 数据库失败！", "错误",
                    JOptionPane.ERROR_MESSAGE);
            e1.printStackTrace();
            return;
        }
        //判断内容是否为空
       if(user==null) {
            JOptionPane.showMessageDialog(LoginDialog.this,
                    "输入的用户名和密码不存在！", "错误",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        LoginDialog.this.dispose();
    }
```

5.6 修改用户登录密码

1) 在 DbUtil 中创建 updateAdminPassword()方法更新密码

```
//管理员修改密码     （DbUtil.java）
  public  static boolean updateAdminPassword(String username, String
newPassword)
         throws SQLException {

      Connection conn = getConnection();
      String sql = "update admin set `password`=sha(?) "
            + " where username=?";
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setString(1,newPassword);
      pstmt.setString(2, username);

      int success = pstmt.executeUpdate();
      conn.close();
      return success>0;
   }

}
```

2) 确定修改按钮事件绑定

```
//修改管理员用户密码     （EditAdminPassword.java）
public void actionPerformed(ActionEvent e) {
   String username = tbUserName.getText();
   String password = tbOldPassword.getText();
   String newPassword =new String(tbNewPassword.getPassword());

    try {
      user = DbUtil.getLoginUser(username, password);
   } catch (SQLException e1) {
      // TODO Auto-generated catch block
      JOptionPane.showMessageDialog(EditAdminPassword.this,
            "用户名或密码错误！", "错误",
            JOptionPane.ERROR_MESSAGE);
      e1.printStackTrace();
   }//利用用户名判断密码是否正确
```

```
    if(user == null) {
        JOptionPane.showMessageDialog(EditAdminPassword.this,
              "用户名或者密码错误!!!!!! ", "错误",
              JOptionPane.ERROR_MESSAGE);
        return;
     }else {
        JOptionPane.showConfirmDialog(EditAdminPassword.this,
              "密码正确!!!!!!", "成功",
              JOptionPane.YES_OPTION);
        try {
          if(DbUtil.updateAdminPassword(username, newPassword)==true) {
              JOptionPane.showConfirmDialog(EditAdminPassword.this,
                      "管理员密码修改成功", "成功",
                       JOptionPane.YES_OPTION);
           }else {
              JOptionPane.showMessageDialog(EditAdminPassword.this,
                    "密码修改失败！", "失败",
                    JOptionPane.ERROR_MESSAGE);
           }
        } catch (HeadlessException | SQLException e1) {
           // TODO Auto-generated catch block
           e1.printStackTrace();
        }



    }

     LoginDialog dialog = new LoginDialog();
       dialog.setModal(true);
       dialog.setVisible(true);

}
```

5.7 修改 JOptionPane 的背景和按钮背景颜色等

1) 修改修改 JOptionPane 控件的背景和按钮背景颜色

```
//修改本页面的 showMessage/Confirm/dialog 的显示主题
UIManager.put("OptionPane.background",Color.white);
UIManager.put("Panel.background", Color.white);
UIManager.put("Button.background", Color.white);
UIManager.put("OptionPane.buttonFont", new FontUIResource(new Font("Microsoft
YaHei UI", Font.PLAIN, 13)));
UIManager.put("OptionPane.font", new FontUIResource(new Font("Microsoft YaHei
UI", Font.PLAIN, 13)));
UIManager.put("Panel.font", new FontUIResource(new Font("Microsoft YaHei UI",
Font.PLAIN, 13)));
```

第 6 章 总结

6.1 实训总结

通过本次实训，我对这学期所学习的 Java 面向对象编程和 AWT、Swing GUI 窗体界面的相关知识有了进一步的认识和理解，但也深深的认识到自己知识的欠缺和诸多不足之处。

本次实训我选择的课题是做一个简单的学生信息管理系统。在设计之初需求分析的时候，由于缺乏软件架构分析的经验导致我在一开始就没有完全想好自己到底要如何去做心中也只有大致的轮廓。当基本功能接近完成的时候，我才发现之前定义的一些方法不适应于我想要实现的需求，于是在原有设计上一改再改。

最终需求实现之后，程序暴露出代码模块之间高耦合的问题，影响了项目后续的可维护性和可扩展性。这其中的主要原因在于我的 Java 基础还不够扎实，写的太少看的也太少以后要多去 GitHub、StackOverflow 等开源和问题社区学习别人的设计思路和优秀的编程经验。

由于时间和知识的欠缺，学生信息管理系统还存在一些暂时没有解决的问题，大致有以下几个方面:

1）功能欠缺

最初打算在添加一个管理员注册和管理员修改基本信息的功能窗体，由于时间问题还是个人开发，不得不放弃。

2）界面布局问题

由于缺少界面布局经验，在开发设置界面时我使用了绝对布局，导致最终界面控件位置出现偏移不够美观。

3）表格数据居中问题

使用 DefaultTableCellRenderer 设置表格内容居中时，ID 列数据不可以随之居中。

5） 代码太长不够精简

由于缺少开发经验，以及 Java 基础不够扎实等，造成了代码模块之间高耦合的问题。

参考文献

[1] Thinbug.JOptionPane如何更改标题框的颜色

https://www.thinbug.com/q/53577990

[2]ashui811.JTable单元格内同居中的方法

https://blog.csdn.net/jarniyy/article/details/52076021

[3]Hiro. 使用Eclipse将工程打包成一个可以执行的Jar文件

http://hirocube.github.io/2015-10-29/eclipse-export-runnable-jar/

[4]阿里巴巴.矢量标题库

https://www.iconfont.cn/