北京郵電大學

Beijing University of Posts and Telecommunications

实 验 报 告

数据库编程

班 级	学 号	学生姓名
2023211804	2023211536	林宁
2023211804	2023211539	尹奥博
2023211805	2023211595	李昊伦

2025 年 5 月 27 日

一、实验环境

jdk 版本：jdk-8u202

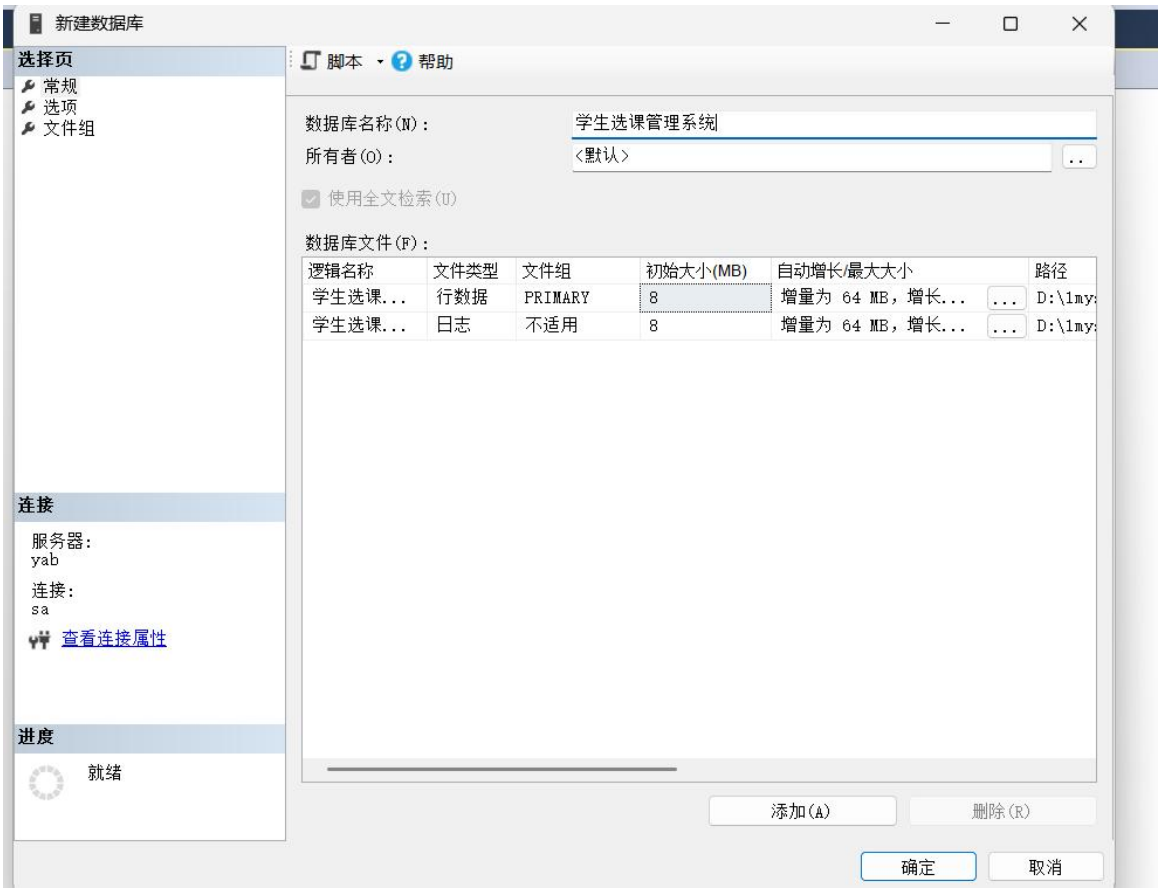
SQL Server jdbc 驱动：sqljdbc4-4.0;

数据库集成环境：SQL Server Management Studio 20

二、导入“学生选课管理系统”数据库

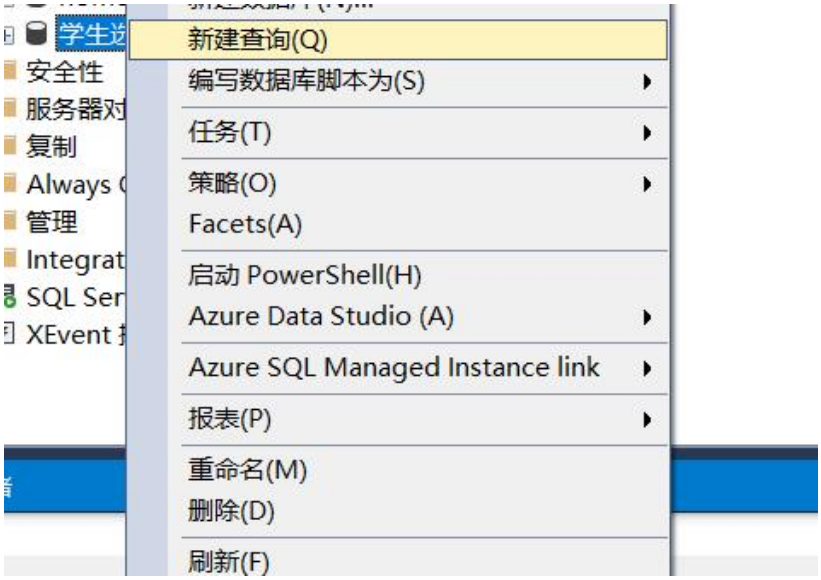
1.创建学生选课管理系统数据库

右键数据库，点击新建数据库，输入“学生管理系统“，点击确定



2.执行学生选课管理系统.sql 脚本

在所建的数据库右键新建查询，输入脚本内容，执行脚本




```
CREATE TABLE 学生基本信息表 (
  学号 varchar(255) NOT NULL,
  姓名 varchar(255) DEFAULT NULL,
  性别 varchar(255) DEFAULT NULL,
  专业 varchar(255) DEFAULT NULL,
  生日 varchar(255) DEFAULT NULL,
  身高 varchar(255) DEFAULT NULL,
  密码 varchar(255) DEFAULT NULL,
  PRIMARY KEY (学号)
);

INSERT INTO 学生基本信息表 VALUES ('104972072408', '杨剑', '男', '通信与信息系统', '1984-06-06', '1.76', '104972072408');
INSERT INTO 学生基本信息表 VALUES ('104972072409', '李春洋', '男', '通信与信息系统', '1984-08-06', '1.72', '104972072409')
INSERT INTO 学生基本信息表 VALUES ('104972072410', '孙磊君', '女', '通信与信息系统', '1984-08-09', '1.6', '104972072410');
INSERT INTO 学生基本信息表 VALUES ('104972072411', '谢惠琳', '女', '通信与信息系统', '1984-08-09', '1.6', '104972072411');
INSERT INTO 学生基本信息表 VALUES ('104972072412', '江根', '男', '通信与信息系统', '1984-08-09', '1.6', '104972072412');
```

108 %

消息

(1 行受影响)

(1 行受影响)

(1 行受影响)

(1 行受影响)

(1 行受影响)

(1 行受影响)

(1 行受影响)

三、以教师身份登录系统

1.修改 db.properties 文件

将 db.properties 文件修改为

```
url="jdbc:sqlserver://127.0.0.1:1433;DatabaseName=学生选课管理系统"
username="sa"
password="123456"
drive="com.microsoft.sqlserver.jdbc.SQLServerDriver"
```

2.获取教师账户密码

执行 select * from 教师表。

```
select * from 教师表
```

08 %

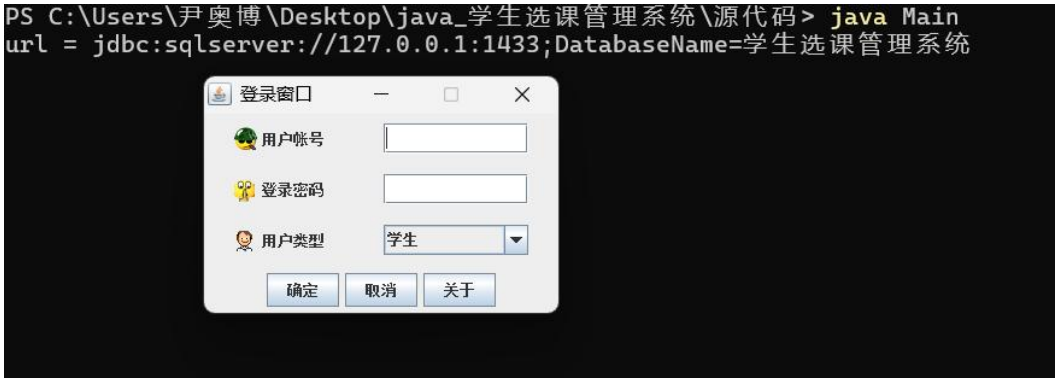
结果 消息

	登陆帐号	教师	登陆密码	院系	性别	年龄	职称
1	1001	李阳	000	外语学院	男	38	教授
2	1002	肖攸安	1002	信息工程学院	男	33	教授
3	1003	江雪梅	1003	信息工程学院	女	26	讲师
4	1004	吕锋	1004	信息工程学院	男	50	教授
5	1005	李方敏	1005	信息工程学院	男	36	教授
6	1006	曾祥金	1006	理学院	男	56	教授

这里选取账号 1002，登录密码 1002。

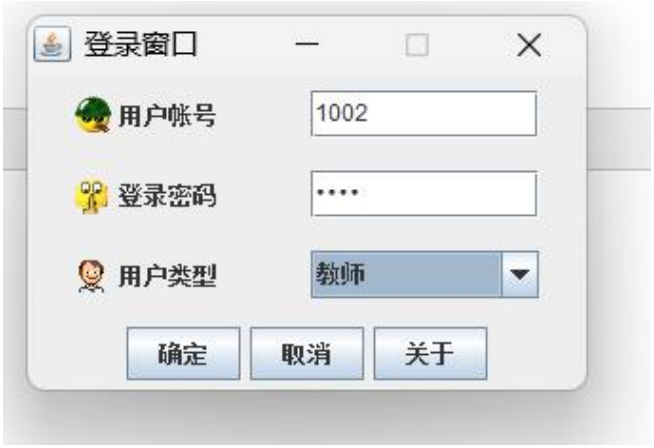
3.运行 java Main

在源代码目录执行 java Main。

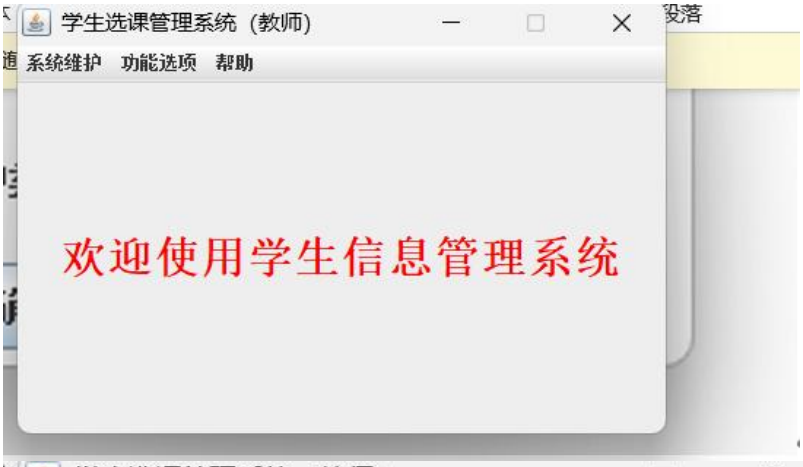


4.登录账户

输入账号 1002，密码 1002，用户类型选择教师。



5.登录成功



四、登录过程程序执行逻辑分析

1.用户界面初始化

（1）创建 LoginFrame 窗口，设置标题、大小和位置。

（2）使用 GridBagLayout 布局创建主面板，包含：用户名输入框（JTextField）、密码输入框（JPasswordField）、用户类型下拉框（JComboBox），选项为"学生"、

"教师"、"管理员"。
（3）底部添加按钮面板，包含"确定"、"取消"和"关于"按钮。

2.用户输入获取

（1）用户在界面中输入：用户名（文本字段）、密码（密码字段）、用户类型（下拉选择）。
（2）用户类型选择通过 ActionListener 监听，将选择的值存储在 selectedItem 变量中。

3.点击"确定"按钮触发登录过程

（1）调用 loginDispose()方法处理登录逻辑
（2）loginDispose()方法执行步骤：
数据库连接：加载 JDBC 驱动程序（Class.forName()）、建立数据库连接（DriverManager.getConnection()）、捕获并处理可能的连接异常
输入验证：检查用户名是否为空、如果为空，显示警告消息并返回
SQL 查询构建：根据选择的用户类型构建不同的 SQL 查询语句。
教师：查询"教师表"、管理员：查询"管理员"表、学生：查询"学生基本信息表"
（3）使用用户输入的用户名和密码作为查询条件
执行查询：创建 Statement 对象、执行 SQL 查询（executeQuery()）、检查是否有匹配记录（next()方法）
结果处理：如果没有匹配记录，显示"没有此用户或密码错误"消息
如果有匹配记录，设置 login 标志为 1（表示登录成功）
（4）登录成功后的处理：
根据用户类型创建并显示相应的界面：学生：StudentsFrame、教师：TeacherFrame、管理员：ManagerFrame
关闭当前登录窗口（dispose()）

4.其他按钮功能

（1）**取消按钮：**直接退出系统（System.exit(0)）
（2）**关于按钮：**显示作者和指导老师信息的对话框

5.数据库连接关闭

登录验证完成后关闭数据库连接（loginConnection.close()）

6.安全注意事项

（1）当前代码存在 SQL 注入风险，因为直接拼接用户输入到 SQL 语句中
（2）密码以明文形式存储在数据库中
（3）建议改进：使用预编译语句（PreparedStatement）防止 SQL 注入、对密码进行哈希处理后再存储和比较、使用连接池管理数据库连接

7.程序流程总结

用户输入 → 验证输入 → 构建 SQL 查询 → 执行查询 → 检查结果 → 根据结果打开相应界面或显示错误消息 → 关闭数据库连接

五、SQL 注入攻击

1.攻击

对账号为 1001 的教师账号进行攻击。

（1）将账号设置为“1001' or '1' = '1') --”，用户类型选择“教师”。

点击确定后，成功进入到学生信息管理系统。

（2）将账号设置为“1001”，密码设置为“' or '1' = '1') --”，用户类型选择“教师”。

点击确定后，成功进入到学生信息管理系统。

2.防御措施

（1）原代码的问题

原代码直接将用户输入（loginUserName 和 loginPassword）拼接到 SQL 查询字符串中。在将用户输入插入 SQL 字符串之前，没有对它们进行“清理”或“转义”的机制。这意味着如果用户输入包含在 SQL 中具有特殊含义的字符（例如单引号 '、分号 ；或注释标记 --），这些字符将被数据库解释为 SQL 命令的一部分，而不是字面量数据。

```
            JOptionPane.showMessageDialog( LoginFrame.this, "用户名必须为字母、数字和、汉字\n及其组合，不允许为空格键。",
                                           "登陆", JOptionPane.WARNING_MESSAGE );
            //setTitle( "无记录显示" );
            return;
        }
        if(selectedItem.equals("教师"))
            loginQuery = "SELECT * FROM 教师表 WHERE(登陆帐号='" + loginUserName  + "' AND 登陆密码 ='" + loginPassword + "')";
        else if(selectedItem.equals("管理员"))
            loginQuery = "SELECT * FROM 管理员 WHERE(用户名='" + loginUserName + "' AND 密码 ='" + loginPassword + "')";
        else //(selectedItem.equals("学生"))
            loginQuery = "SELECT * FROM 学生基本信息表 WHERE(学号='" + loginUserName + "' AND 密码 ='" + loginPassword + "')";
        loginStatement = loginConnection.createStatement();
        System.out.println(loginQuery);  // XD
        loginResultSet = loginStatement.executeQuery( loginQuery );
        boolean Records = loginResultSet.next();
        if ( ! Records )
        {
            JOptionPane.showMessageDialog(LoginFrame.this, "没有此用户或密码错误" );
            return;
        }
        else
```

（2）改进措施

①使用 PreparedStatement 代替 Statement

原问题：之前使用 Statement，它直接将用户输入字符串拼接到 SQL 查询中，如果用户输入恶意代码（比如 1001' or '1' = '1') --），数据库会将其视为 SQL 命令的一部分来执行。

优化：代码现在使用 PreparedStatement。PreparedStatement 会先将带有占位符（?）的 SQL 查询模板发送给数据库进行预编译，数据库明确了哪些是命令，哪些是数据。

②使用参数绑定 setString()

原问题：用户名和密码是直接与 SQL 字符串拼接的，没有进行任何处理。

优化：通过 localLoginStatement.setString(1, loginUserName);和 localLoginStatement.setString(2, loginPassword); 方法，将用户输入作为参数绑定到 SQL 模板中的 ？ 占位符上。setString() 方法会自动对这些参数进行转义，这意味着无论用户输入什么字符，包括 SQL 关键字或特殊符号，它们都会被视为纯粹的数据，而不会被数据库错误地解析为可执行的 SQL 代码。
（完整代码见附录）

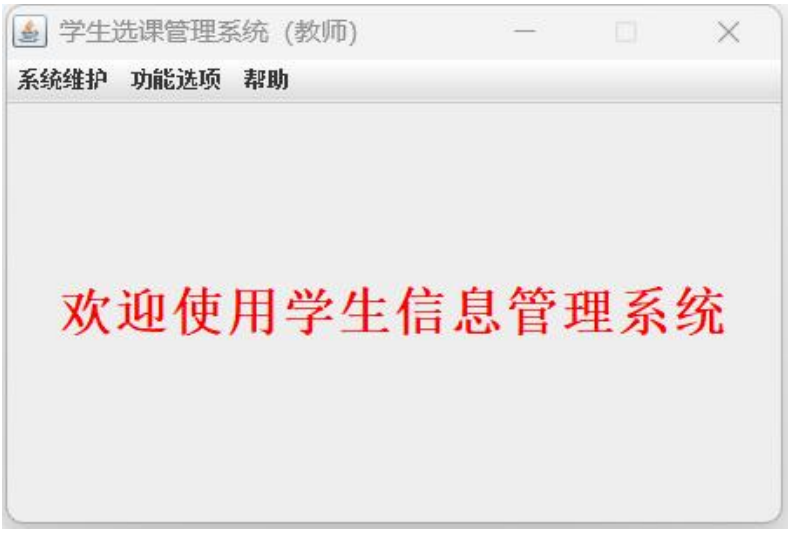
```
// --- SQL 注入防御的关键修改从这里开始 ---
// 将类成员的 loginStatement 和 loginResultSet 声明移到这里作为局部变量
// 这样可以确保它们在 try-finally 块中正确关闭，并避免与类成员混淆
PreparedStatement localLoginStatement = null;
ResultSet localLoginResultSet = null;

try
{
   String loginQuery;
   String loginUserName =  myTextField.getText();
   String loginPassword = new String(passwordField.getPassword());
   if(myTextField.getText().equals( "" ))
   {
      JOptionPane.showMessageDialog( LoginFrame.this, "用户名必须为字母、数字和、汉字\n及其组合，不允许为空格键。",
                                     "登陆", JOptionPane.WARNING_MESSAGE );
      return;
   }

   // 修改 SQL 查询字符串：使用占位符 '?' 代替直接拼接
   if(selectedItem.equals("教师"))
      loginQuery = "SELECT * FROM 教师表 WHERE(登陆帐号=? AND 登陆密码 =?)";
   else if(selectedItem.equals("管理员"))
      loginQuery = "SELECT * FROM 管理员 WHERE(用户名=? AND 密码 =?)";
   else //(selectedItem.equals("学生"))
      loginQuery = "SELECT * FROM 学生基本信息表 WHERE(学号=? AND 密码 =?)";
```

```
// 使用 PreparedStatement 预编译查询
localLoginStatement = loginConnection.prepareStatement(loginQuery);

// 通过 setString() 安全地设置参数，防止注入
localLoginStatement.setString(1, loginUserName);
localLoginStatement.setString(2, loginPassword);

// 保持原始的 System.out.println(loginQuery); 但请注意它只显示模板，不包含实际参数
System.out.println(loginQuery);  // XD

// 执行 PreparedStatement
localLoginResultSet = localLoginStatement.executeQuery();
boolean Records = localLoginResultSet.next();
if ( ! Records )
{
   JOptionPane.showMessageDialog(LoginFrame.this, "没有此用户或密码错误" );
   return;
}
else
{
   login = 1 ;
}
// 移除这里对 loginConnection.close() 的调用，因为它统一在 finally 块中处理
// loginConnection.close();
```

```
      login = 1 ;
   }
   // 移除这里对 loginConnection.close() 的调用，因为它统一在 finally 块中处理
   // loginConnection.close();
}
catch(SQLException sqlex)
{
   //sqlex.printStackTrace();
   JOptionPane.showMessageDialog (LoginFrame.this, sqlex,
                                  "学生选课管理系统", JOptionPane.WARNING_MESSAGE );
}
finally {
   // 新增 finally 块，确保数据库资源（ResultSet, PreparedStatement, Connection）关闭
   try {
      if (localLoginResultSet != null) {
         localLoginResultSet.close();
      }
      if (localLoginStatement != null) {
         localLoginStatement.close();
      }
      // 注意：loginConnection 是类成员，如果需要它在其他方法中保持打开，则不在这里关闭。
      // 但通常登录后连接应关闭，以避免资源泄露。根据您的项目设计决定是否关闭。
      // 如果登录后立即切换界面，且新界面也需要此连接，则可以不关。
      // 但最佳实践是在使用完后立即关闭。这里先保留关闭逻辑。
      if (loginConnection != null) {
         loginConnection.close();
      }
   } catch (SQLException e) {
      e.printStackTrace();
      JOptionPane.showMessageDialog(LoginFrame.this, "关闭数据库资源时发生错误: " + e.getMessage(),
            "资源关闭错误", JOptionPane.ERROR_MESSAGE);
   }
}
// --- SQL 注入防御的关键修改到此结束 ---
}
```

（3）改进后的效果

①若输入正确的账号和密码，则可以正常登录。



②将账号设置为“1001' or '1' = '1') --”，用户类型选择“教师”。

点击确定后，无法进入到学生信息管理系统。

③将账号设置为“1001”，密码设置为“' or '1' = '1') --”，用户类型选择“教师”。

点击确定后，成功进入到学生信息管理系统。

附录：

修改后的 LoginFrame.java 代码：

```
import java.sql.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import javax.swing.table.*;
import javax.xml.crypto.Data;


/**
 * 一个带有登录面板的登录窗口。
*/

public class LoginFrame extends JFrame
{
    public LoginFrame()
    {
        setTitle("登录窗口");
        setResizable(false);
        setLocation(350, 200);
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

        //add main panel to frame

        JPanel mainPanel = new JPanel(new GridBagLayout());
        constraints = new GridBagConstraints();

        label1 = new JLabel("用户帐号", new ImageIcon("01.gif"), SwingConstants.CENTER);
        constraints.weightx = 100;
        constraints.weighty = 100;
        constraints.gridx = 0;
        constraints.gridx = 0;
        constraints.gridwidth = 1;
        constraints.gridheight = 1;
```

```
mainPanel.add(label1, constraints);

myTextField = new JTextField();
myTextField.setPreferredSize(new Dimension(120, 25));
constraints.gridx = 0;
constraints.gridx = 1;
constraints.gridwidth = 1;
constraints.gridheight = 1;
mainPanel.add(myTextField, constraints);

label2 = new JLabel("登录密码", new ImageIcon("02.gif"), SwingConstants.CENTER);
constraints.gridx = 1;
constraints.gridx = 0;
constraints.gridwidth = 1;
constraints.gridheight = 1;
mainPanel.add(label2, constraints);

passwordField = new JPasswordField();
passwordField.setPreferredSize(new Dimension(120, 25));
constraints.gridx = 1;
constraints.gridx = 1;
constraints.gridwidth = 1;
constraints.gridheight = 1;
mainPanel.add(passwordField, constraints);

label3 = new JLabel("用户类型", new ImageIcon("03.gif"), SwingConstants.CENTER);
constraints.gridx = 2;
constraints.gridx = 0;
constraints.gridwidth = 1;
constraints.gridheight = 1;
mainPanel.add(label3, constraints);

// **此处修改：为 JComboBox 添加泛型 <String> 以消除编译警告**
userCombo = new JComboBox<String>();
userCombo.setPreferredSize(new Dimension(120, 25));
userCombo.setEditable(false);
userCombo.addItem("学生");
userCombo.addItem("教师");
userCombo.addItem("管理员");
```

```
userCombo.setSelectedItem("学生");
selectedItem = (String)userCombo.getSelectedItem();
constraints.gridx = 2;
constraints.gridx = 1;
constraints.gridwidth = 1;
constraints.gridheight = 1;
mainPanel.add(userCombo, constraints);

//get selected item, so we can decide to show which frame
userCombo.addActionListener(new
                                        ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        selectedItem = (String)userCombo.getSelectedItem();      //方法别忘了加括
号
    }
});
add(mainPanel, BorderLayout.CENTER);

//add button panel to the frame

JPanel buttonPanel = new JPanel();
//add login button
JButton loginButton = new JButton("确定");
loginButton.addActionListener(new
                                            ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        loginDispose();

        if(login == 1)
        {
            if(selectedItem.equals("学生"))
            {
                // 这些 Frame 应该是您项目中已有的类，这里不提供它们的定
义
                JFrame f = new StudentsFrame();
```

```
                    f.setVisible(true);
                    dispose();
                }

                else if(selectedItem.equals("教师"))
                {
                    JFrame f = new TeacherFrame();
                    f.setVisible(true);
                    dispose();
                }
                else if(selectedItem.equals("管理员"))
                {
                    JFrame f = new ManagerFrame();
                    f.setVisible(true);
                    dispose();
                }
            }
        }
});
buttonPanel.add(loginButton);

//add cancel button
JButton cancelButton = new JButton("取消");
cancelButton.addActionListener(new
                                        ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }
});
buttonPanel.add(cancelButton);

//add about button
JButton aboutButton = new JButton("关于");
aboutButton.addActionListener(new
                                        ActionListener()
{
    public void actionPerformed(ActionEvent e)
```

```
                {
                    JOptionPane.showMessageDialog(LoginFrame.this,
                                                  "          作者：李常友\n 指导老师：
龙毅宏",
                                                  "作品信息",

JOptionPane.INFORMATION_MESSAGE,
                                                  new ImageIcon("01.gif"));

                    return;



                }
            });
            buttonPanel.add(aboutButton);

            add(buttonPanel, BorderLayout.SOUTH);
        }

        private void loginDispose()
        {
            //加载驱动程序以连接数据库
            try
            {
                Class.forName(DataBaseInfo.drive); // DataBaseInfo  应是您项目中的类
                loginConnection                                                    =
DriverManager.getConnection(DataBaseInfo.url,DataBaseInfo.username,DataBaseInfo.password)
;
            }
            //捕获加载驱动程序异常
            catch ( ClassNotFoundException cnfex )
            {
                cnfex.printStackTrace();
                JOptionPane.showMessageDialog (LoginFrame.this, cnfex,
                                               " 学 生 选 课 管 理 系 统 ",
JOptionPane.WARNING_MESSAGE );
                System.exit( 1 );            // terminate program
            }
            //捕获连接数据库异常
```

```
        catch ( SQLException sqlex )
        {
            sqlex.printStackTrace();
            System.out.println(DataBaseInfo.url);
            JOptionPane.showMessageDialog (LoginFrame.this, "无法连接到 SQL SERVER ，
\n 请确认 SQL SERVER 是否运行\n 或数据源设置是否正确！ ",
                                                    " 学 生 选 课 管 理 系 统 ",
JOptionPane.WARNING_MESSAGE );
            System.exit( 1 );     // terminate program
        }

        // --- SQL 注入防御的关键修改从这里开始 ---
        // 将类成员的 loginStatement 和 loginResultSet 声明移到这里作为局部变量
        // 这样可以确保它们在 try-finally 块中正确关闭，并避免与类成员混淆
        PreparedStatement localLoginStatement = null;
        ResultSet localLoginResultSet = null;

        try
        {
            String loginQuery;
            String loginUserName =    myTextField.getText();
            String loginPassword = new String(passwordField.getPassword());
            if(myTextField.getText().equals( "" ))
            {
                JOptionPane.showMessageDialog( LoginFrame.this, "用户名必须为字母、数
字和、汉字\n 及其组合，不允许为空格键。",
                                                    "          登          陆          ",
JOptionPane.WARNING_MESSAGE );
                return;
            }

            // 修改 SQL 查询字符串：使用占位符 '?' 代替直接拼接
            if(selectedItem.equals("教师"))
                loginQuery = "SELECT * FROM 教师表 WHERE(登陆帐号=? AND 登陆
密码 =?)";
            else if(selectedItem.equals("管理员"))
                loginQuery = "SELECT * FROM 管理员 WHERE(用户名=? AND 密码
=?)";
            else //(selectedItem.equals("学生"))
```

```
                loginQuery = "SELECT * FROM 学生基本信息表 WHERE(学号=? AND
密码 =?)";

                // 使用 PreparedStatement 预编译查询
                localLoginStatement = loginConnection.prepareStatement(loginQuery);

                // 通过 setString() 安全地设置参数，防止注入
                localLoginStatement.setString(1, loginUserName);
                localLoginStatement.setString(2, loginPassword);

                // 保持原始的 System.out.println(loginQuery); 但请注意它只显示模板，不包含
实际参数
                System.out.println(loginQuery);   // XD

                // 执行 PreparedStatement
                localLoginResultSet = localLoginStatement.executeQuery();
                boolean Records = localLoginResultSet.next();
                if ( ! Records )
                {
                    JOptionPane.showMessageDialog(LoginFrame.this, "没有此用户或密码错
误" );
                    return;
                }
                else
                {
                    login = 1 ;
                }
                // 移除这里对 loginConnection.close() 的调用，因为它统一在 finally 块中处
理
                // loginConnection.close();
            }
            catch(SQLException sqlex)
            {
                //sqlex.printStackTrace();
                JOptionPane.showMessageDialog (LoginFrame.this, sqlex,
                                               "学生选课管理系统",
JOptionPane.WARNING_MESSAGE );
            }
            finally {
```

```
                // 新增 finally 块，确保数据库资源（ResultSet, PreparedStatement, Connection）关闭
                try {
                    if (localLoginResultSet != null) {
                        localLoginResultSet.close();
                    }
                    if (localLoginStatement != null) {
                        localLoginStatement.close();
                    }
                    // 注意：loginConnection 是类成员，如果需要它在其他方法中保持打开，则不在这里关闭。
                    // 但通常登录后连接应关闭，以避免资源泄露。根据您的项目设计决定是否关闭。
                    // 如果登录后立即切换界面，且新界面也需要此连接，则可以不关。
                    // 但最佳实践是在使用完后立即关闭。这里先保留关闭逻辑。
                    if (loginConnection != null) {
                        loginConnection.close();
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                    JOptionPane.showMessageDialog(LoginFrame.this, "关闭数据库资源时发生错误: " + e.getMessage(),
                            "资源关闭错误", JOptionPane.ERROR_MESSAGE);
                }
            }
            // --- SQL 注入防御的关键修改到此结束 ---
        }

        private static final int DEFAULT_WIDTH  = 300;
        private static final int DEFAULT_HEIGHT = 200;
        private GridBagConstraints constraints;
        private JLabel label1, label2, label3;
        private JPasswordField passwordField;

        // **此处修改：为类成员的 JComboBox 添加泛型 <String>**
        private JComboBox<String> userCombo;

        private String selectedItem;
        private int login = 0;
```

```
    private Connection loginConnection;

    // **此处修改：删除这些类成员变量，因为它们现在在 loginDispose() 中是局部变量**
    // private Statement loginStatement;
    // private ResultSet loginResultSet;

    public static JTextField myTextField;                    //声明登陆名为全局变量!!!!!
}
```