北京郵電大學

Beijing University of Posts and Telecommunications

实验报告：数据库编程

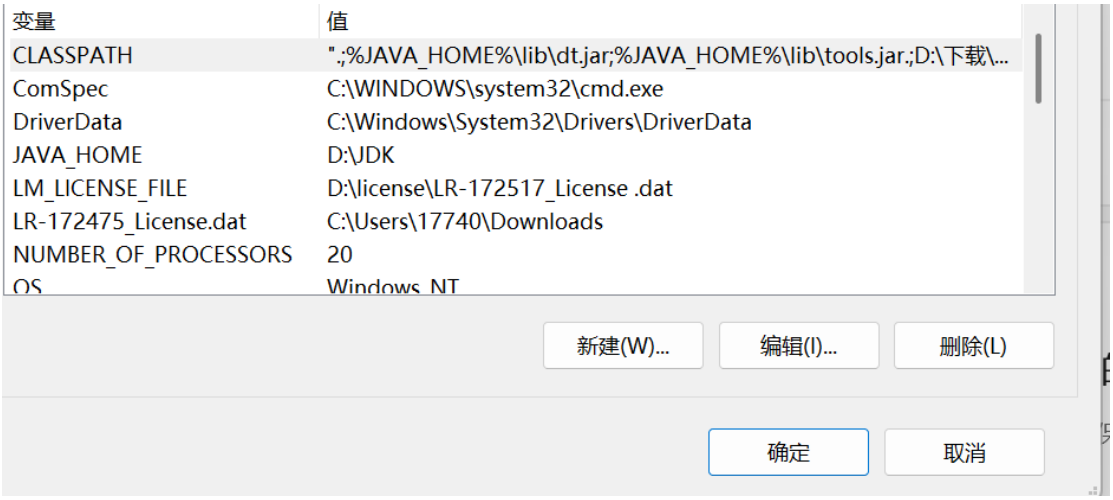
姓名: 蔡昱良 詹冲

学号: 2023211617 2023211616

班级: 2023211807 2023211801

专业: 网络空间安全 信息安全

一.课前配置 java 环境和 jdbc 驱动



下载和安装好相关文件之后，在系统环境中配置
Classpath 变量中加入 ".;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar.;D:\下载\mysql-connector-java-8.0.27\mysql-connector-java-8.0.27.jar";
Path 变量加入
%JAVA_HOME%\bin
用 javac java java –version 这三个命令来检查是否配置成功

二.跑通 jdbcdemo1 文件：

使用 javac jdbcDemo1.java 先对文件编译
再用 java jdbcDemo 运行文件
编译前要先修改，在程序中填上自己的用户名和密码，并根据自己使用的数据库来调整连接数据库部分的语句，老师给的程序连接的是 SQL_server 数据库，连接 MySQL 的语句为注释，将这两个注释换一下就好
执行成功后，连接到 demodb 数据库，并执行 CREATE TABLE Personnel
语句创建表格。
之后再按照同样的方式编译和运行 jdbcDemo2.java 直到 jdbcDemo4.java，里面对数据库的操作是往 Personnel 对数据进行增加和修改等操作。

```
// 恢复 MySQL 驱动（移除 SQL Server 驱动）
            Class.forName("com.mysql.cj.jdbc.Driver");

            // 恢复 MySQL 连接 URL（移除 SQL Server URL）
            String url = "jdbc:mysql://127.0.0.1:3306/demodb?serverTimezone=UTC";
            String userName = "root"; // MySQL 用户名
            String password = "123456789@Cyl"; // MySQL 密码
```

三.解压运行 java 学生选课程序

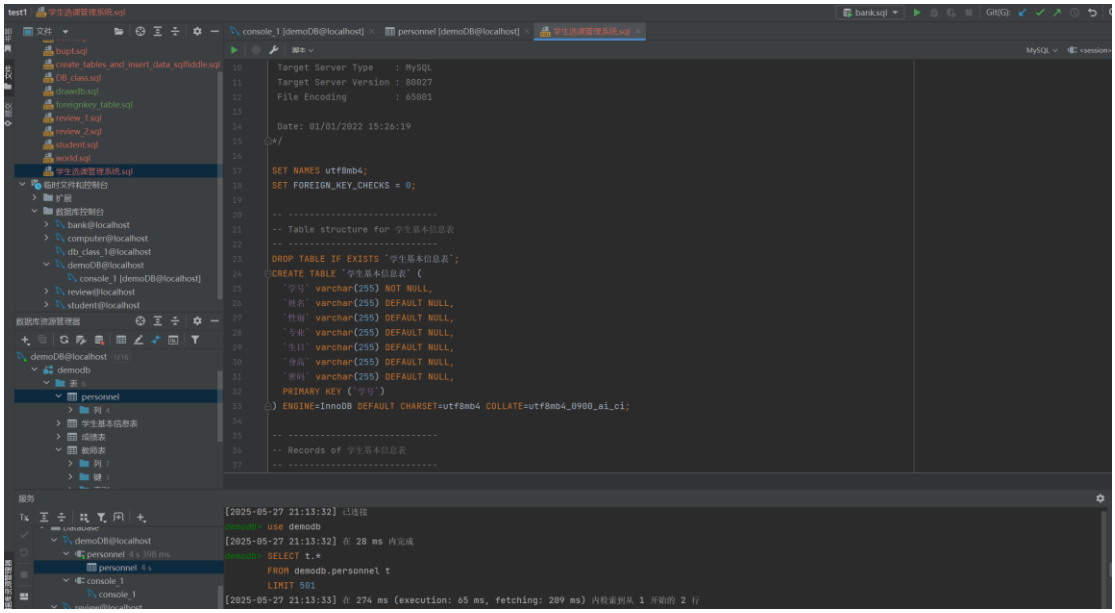
在“源代码”文件夹中，找到/db.properties 或者/db.properties.MySQL 文件，因为我使用 MySQL 数据库，将他们的内容写为

```
url="jdbc:mysql://localhost:3306/demoDB"//demoDB 是我使用的数据库名
username="root"
password="123456789@Cyl"
drive="com.mysql.cj.jdbc.Driver"
```

在 DataBaseInfo 文件里，确保开头某行是这么写的:

```
BufferedReader in = new BufferedReader(new FileReader("./db.properties"));
```

然后还要运行老师给的“学生管理系统.sql”文件，在 demoDB 数据库完成建表和插入数据的操作:



然后就可以将“源代码”文件夹在终端打开，编译和运行 Main.java 文件，连接到 MySQL 中的 demoDB 数据库，并且打开登录界面。自行查看数据库中的数据，自选账号密码登录。

四.查看 logininfo.java 文件，查看登录判断逻辑

关键在于 logindipose 代码段定义的 SQL 语句:

```
if(selectedItem.equals(anObject:"教师"))
    loginQuery = "SELECT * FROM 教师表 WHERE(登陆帐号='" + loginUserName  + "' AND 登陆密码 ='" + loginPassword + "')";
else if(selectedItem.equals(anObject:"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, message:"没有此用户或密码错误" );
    return;
}
else
{
    login = 1 ;
}
```

若输入的用户名和密码能在数据库对应的表匹配上，则 login=1，让登录可以成功。

```
            loginDispose();

            if(login == 1)
            {
                if(selectedItem.equals("学生"))
                {
                    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();
                }
            }
```

Logindipose（）调用，login=1 就展示不同界面

五.尝试 SQL 注入，无账号密码，实现登录

定义的验证 SQL 语句类似

```
loginQuery = "SELECT * FROM 教师表 WHERE(登陆帐号='" + loginUserName  + "' AND 登陆密码 ='"
+ loginPassword + "')";
```

```
```

这种直接进行字符串拼接的方式存在严重的 SQL 注入风险。

使用—或#可以将之后的语句注释掉，可以插入带有‘#’或者‘--’的语句，让判断条件变为“登录账号=‘’ or 某个永远为真的条件”

例如，输入登录账号：

' or 1=1)#

根据

```
loginQuery = "SELECT * FROM 教师表 WHERE(登陆帐号='" + loginUserName  + "' AND 登陆密码 ='"
+ loginPassword + "')";
```

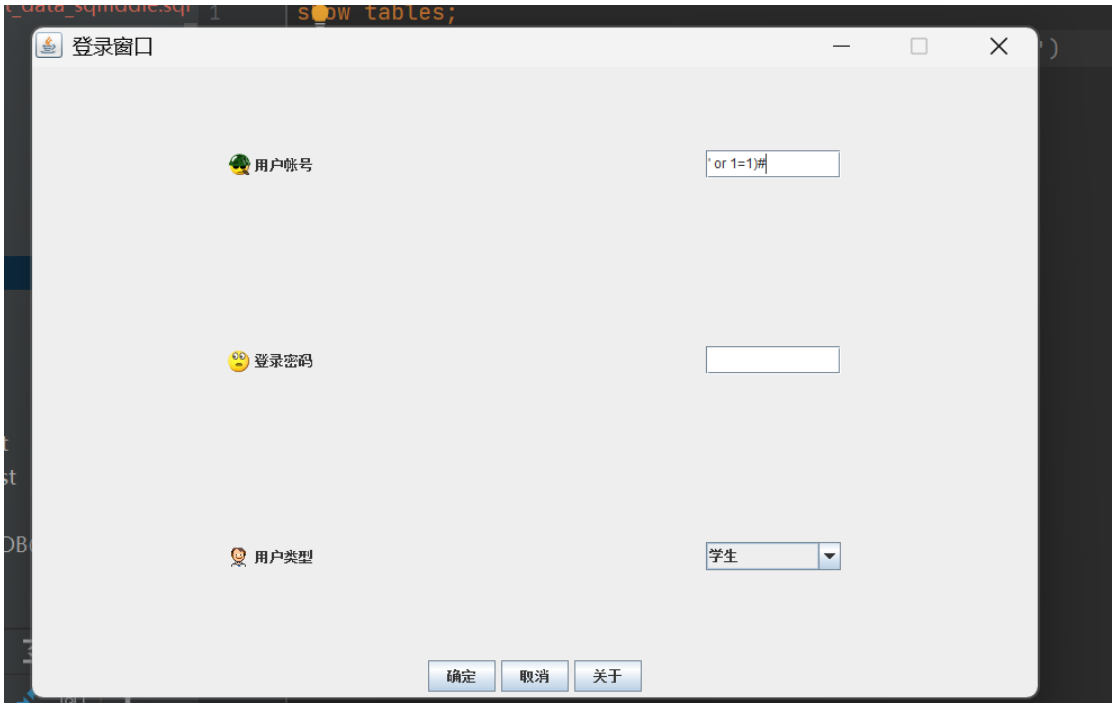
语句会变成：

```
SELECT * FROM 管理员 WHERE(用户名='' or 1=1)#' AND 密码 ='')
```

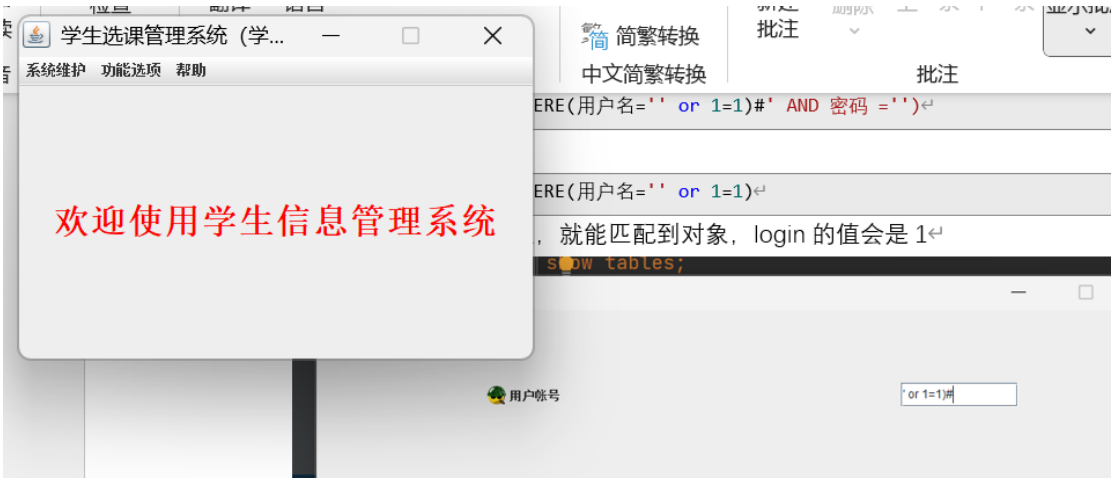
也就是执行

```
SELECT * FROM 管理员 WHERE(用户名='' or 1=1)
```

这样的话只要表不为空，就能匹配到对象，login 的值会是 1



最终登录成功



六.实验总结

技术能力提升

环境配置与调试：掌握 Java 开发环境及 JDBC 驱动的配置流程，理解环境变量对程序运行的影响，提升了问题排查与调试能力。
JDBC 编程：通过实践掌握了 JDBC 连接数据库、执行 SQL 语句及处理查询结果的基本流程，加深了对数据库应用开发的理解。
系统部署：学会了通过修改配置文件、执行数据库脚本等方式部署 Java 应用程序，了解了程序与数据库协同工作的机制。

安全意识增强

SQL 注入原理：通过实际攻击案例，深刻理解了 SQL 注入的本质是字符串拼接导致的代码与数据未分离，攻击者可利用特殊字符篡改 SQL 语句逻辑。在今后的学习和开发中，需始终将安全性纳入考量，以严谨的编码习惯和安全意识构建可靠的数据库应用系统。