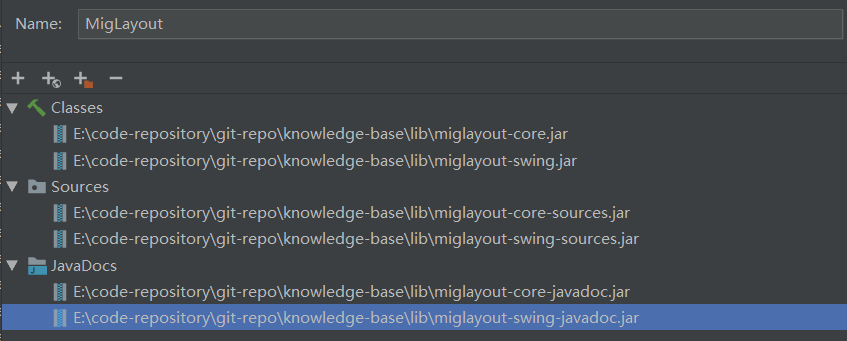
### 1. 先运行sql文件，初始化数据库和表

sql文件路径：gui-swing\src\main\resources\sqlScripts\init.sql

### 2. 用idea导入项目

导入或者打开时时选择Maven工程的方式导入

### 3. 在projectstruct面板导入相关lib



### 4. 运行入口类

gui-swing\src\main\java\Main.java

### 5. 关键代码

* 数据库操作dao模块代码

数据库连接工具类 [需要修改url、user、password为你自己的]

public final class JDBCUtils {  
 private static String url = "jdbc:sqlserver://192.168.3.128:1433;DatabaseName=book\_manage";   
 private static String user = "sa";  
 private static String password = "xx@xg123";  
  
 static {  
 try {  
 Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver"); // 保证注册驱动只会做一次  
 } catch (ClassNotFoundException e) {  
 throw new ExceptionInInitializerError(e);  
 }  
 }  
  
 private JDBCUtils(){} // 私有的构造方法保证不能被创建  
  
 public static Connection getConnection(){  
 try {  
 return DriverManager.getConnection(url,user,password);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public static void free(ResultSet rs, Statement st, Connection conn){  
 try {  
 if (rs != null){  
 rs.close(); // 如果这一行抛异常 也会进行  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if(st != null){  
 st.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 if(conn != null){  
 try {  
 conn.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 }  
}

数据库操作示例 [查询、更新操作]  
book、search模块代码与login近似

public User queryUserByName(String name) throws SQLException {  
 Connection conn = null;  
 PreparedStatement st = null;  
 ResultSet rs = null;  
 String sql;  
 User user = null;  
  
 try {  
 sql = "select \* from dbo.te\_user where username = ?";  
  
 conn = JDBCUtils.getConnection();  
  
// st = conn.createStatement();  
 st = conn.prepareStatement(sql);  
 st.setString(1,name);  
  
 rs = st.executeQuery();  
  
 while (rs.next()){  
 user = new User();  
 user.setUsername(rs.getString("username"));  
 user.setPassword(rs.getString("password"));  
 }  
 }finally {  
 JDBCUtils.free(rs,st,conn);  
 }  
 return user;  
 }  
  
 // 插入一个User，注册功能  
 public int insertUser(String name,String password) throws SQLException {  
 // 增删改都是update 返回的是一个int 类型，不需要对结果集做处理  
 Connection conn = null;  
 PreparedStatement st = null;  
 ResultSet rs = null;  
 String sql;  
 int i;  
 // 先验证数据库有没有重名的用户  
 if (null != queryUserByName(name)){  
 return 0;  
 }  
  
 try {  
 sql = "insert into dbo.te\_user values (?,?)";  
  
 conn = JDBCUtils.getConnection();  
  
 st = conn.prepareStatement(sql);  
  
 st.setString(1, name);  
 st.setString(2, password);  
  
 i = st.executeUpdate();  
 // 数据库客户端里面修改的话 就会 显示有多少条受影响 这个同理  
  
 }finally {  
 JDBCUtils.free(rs,st,conn);  
 }  
 return i;  
 }

* 登陆、注册按钮的事件与逻辑[login模块]

private void loginbuttonActionPerformed(ActionEvent e) {  
 // TODO add your code here  
 String username = textField1.getText(); // 用户名  
 String password = new String(passwordField1.getPassword()); // 密码  
 System.out.println(username+"\t"+password);  
  
 try {  
 user = userDao.queryUserByName(username);  
 } catch (SQLException e1) {  
 System.out.println("数据库操作出错");  
 e1.printStackTrace();  
 }  
 if (user!=null && password.equals(user.getPassword())){  
 label4.setText("登陆成功");  
 label4.setVisible(true);  
 SearchFrame searchFrame = new SearchFrame(user);  
  
 }else {  
 label4.setText("登陆失败");  
 label4.setVisible(true);  
 }  
 }  
  
  
 private void registryActionPerformed(ActionEvent e) {  
 // TODO add your code here  
 String username = textField1.getText(); // 用户名  
 String password = new String(passwordField1.getPassword()); // 密码  
 System.out.println(username+"\t"+password);  
 int i = 0;  
 try {  
 i = userDao.insertUser(username, password);  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
 if(0 == i){  
 label4.setText("注册失败,用户名已经存在");  
 label4.setVisible(true);  
 }else {  
 label4.setText("注册成功");  
 label4.setVisible(true);  
 }  
 }

* 查询按钮的事件与逻辑[search模块]

private void searchActionPerformed(ActionEvent e) {  
 // TODO add your code here  
 String name = textField1.getText();  
  
 try {  
 book = bookDao.queryByName(name);  
 } catch (SQLException e1) {  
 System.out.println("数据库操作出错");  
 e1.printStackTrace();  
 }  
 if( null == book ){  
 label2.setText("查询出错: 没有这本书，换个名字试试");  
 label2.setVisible(true);  
 }else {  
 BookDialog bookDialog = new BookDialog(this,book,user );  
  
 }  
 // 查询出错: 没有这本书，换个名字试试  
  
 }

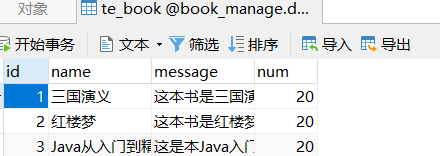
* 借书还书按钮的事件与逻辑[book模块]

private void borrowActionPerformed(ActionEvent e) {
  
 label10.setText("");
  
 try {
  
 // 关系表中 插入这条记录
  
 if (1 == (userbookDao.insert(user.getUsername(), book.getName()))) {
  
 changeMessage(this.book, this.user);
  
 } else {
  
 label10.setText("借阅失败");
  
 }
  
 } catch (SQLException e1) {
  
 e1.printStackTrace();
  
 }
  
   
 }
  
   
 private void revertActionPerformed(ActionEvent e) {
  
 label10.setText("");
  
 try {
  
 if (1 == userbookDao.delete(user.getUsername(), book.getName())) {
  
 changeMessage(this.book, this.user);
  
 } else {
  
 label10.setText("归还失败");
  
 }
  
 } catch (SQLException e1) {
  
 e1.printStackTrace();
  
 }
  
 }

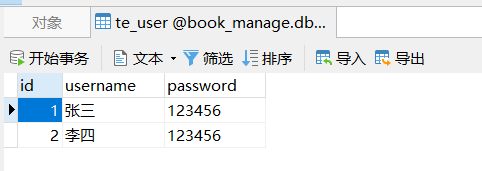
### 6.代码演示

#### 1. 表

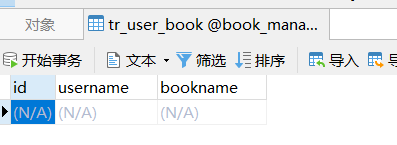
* 书籍实体表



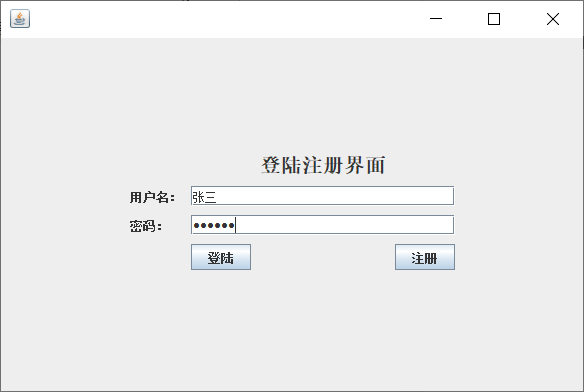
* 用户实体表



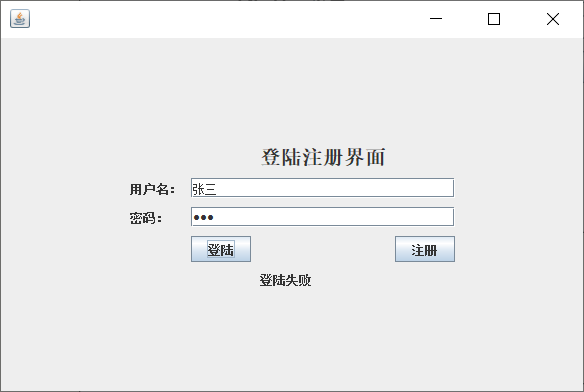
* 用户书籍关系表 [借书还书逻辑表]



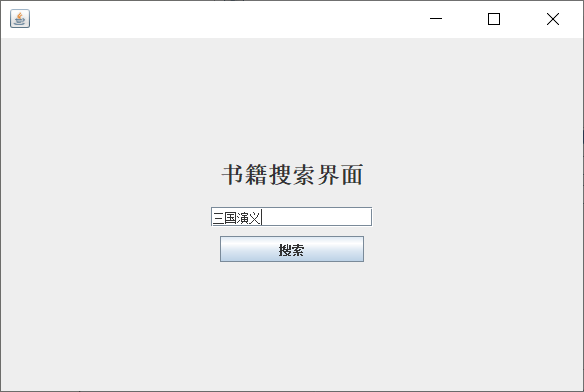
#### 2. 界面



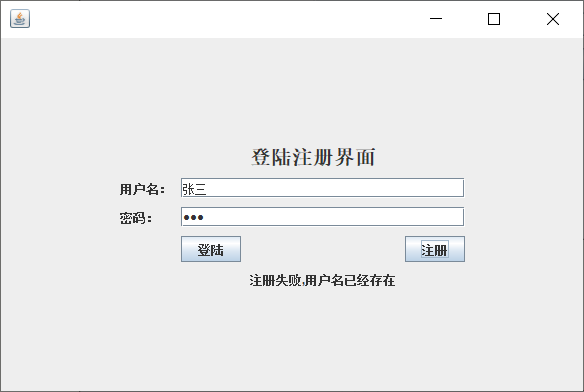
* 密码错误、登陆失败

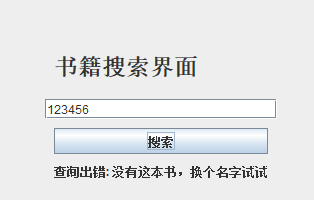


* 数据库有相同的用户名时注册失败

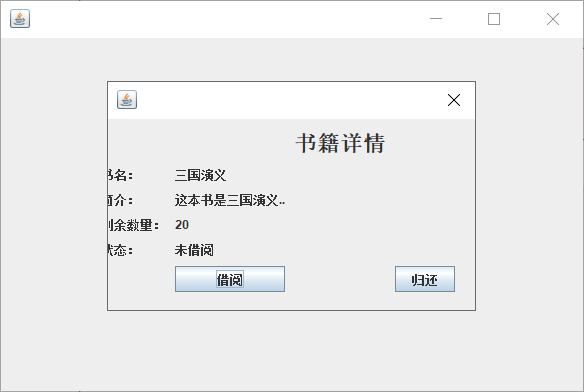


* 搜索与搜索失败

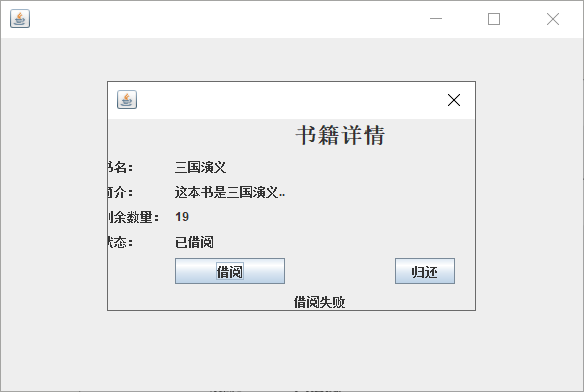




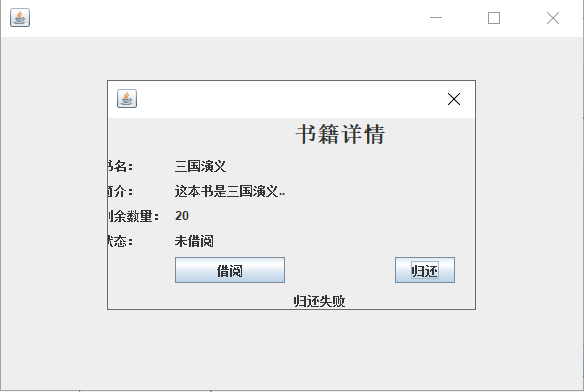
* 刚开始查询出来的时候



* 借书



* 重复借书



* 重复还书

