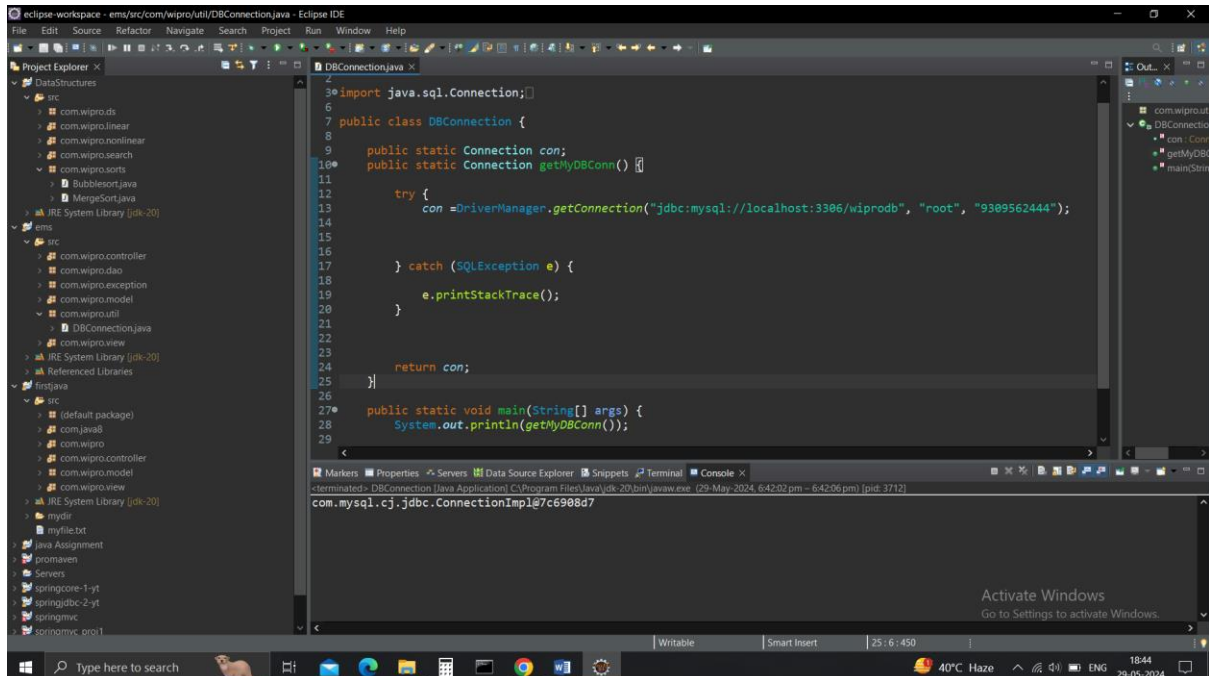


Task1: Write a Java program that connects to a SQLite database and prints out the connection object to confirm successful connection



The screenshot shows the Eclipse IDE with a project named 'wipro'. The 'DBConnection.java' file is open in the editor. The code imports 'java.sql.Connection' and defines a 'DBConnection' class with a static 'Connection con' and a 'getMyDBConn()' method. The 'getMyDBConn()' method uses 'DriverManager.getConnection()' to connect to a MySQL database at 'localhost:3306/wiprodb' with username 'root' and password '9309562444'. The 'main' method calls 'getMyDBConn()' and prints the result. The console shows the connection string: 'com.mysql.cj.jdbc.ConnectionImpl@7c6908d7'.

```
import java.sql.Connection;

public class DBConnection {

    public static Connection con;

    public static Connection getMyDBConn() {

        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/wiprodb", "root", "9309562444");

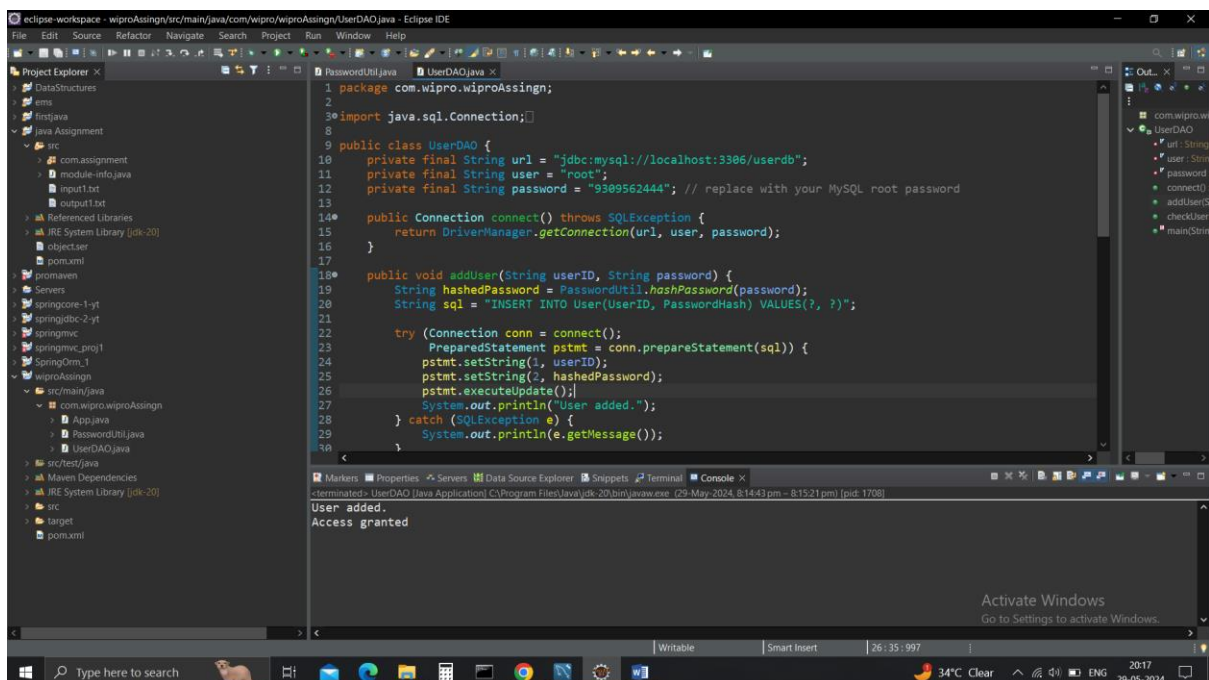
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return con;
    }

    public static void main(String[] args) {
        System.out.println(getMyDBConn());
    }
}
```

com.mysql.cj.jdbc.ConnectionImpl@7c6908d7

Task 2: Create a table 'User' with a following schema 'User ID' and 'Password' stored as hash format (note you have research on how to generate hash from a string), accept "User ID" and "Password" as input and check in the table if they match to confirm whether user access is allowed or not in java.



The screenshot shows the Eclipse IDE with a project named 'wipro'. The 'UserDAO.java' file is open in the editor. The code imports 'java.sql.Connection' and defines a 'UserDAO' class. The 'connect()' method returns a 'Connection' object. The 'addUser()' method takes 'userID' and 'password' as input, hashes the password using 'PasswordUtil.hashPassword()', and inserts the user into the 'User' table. The 'main' method calls 'addUser()' and prints the result. The console shows the output: 'User added. Access granted'.

```
package com.wipro.wiproAssignn;

import java.sql.Connection;

public class UserDAO {

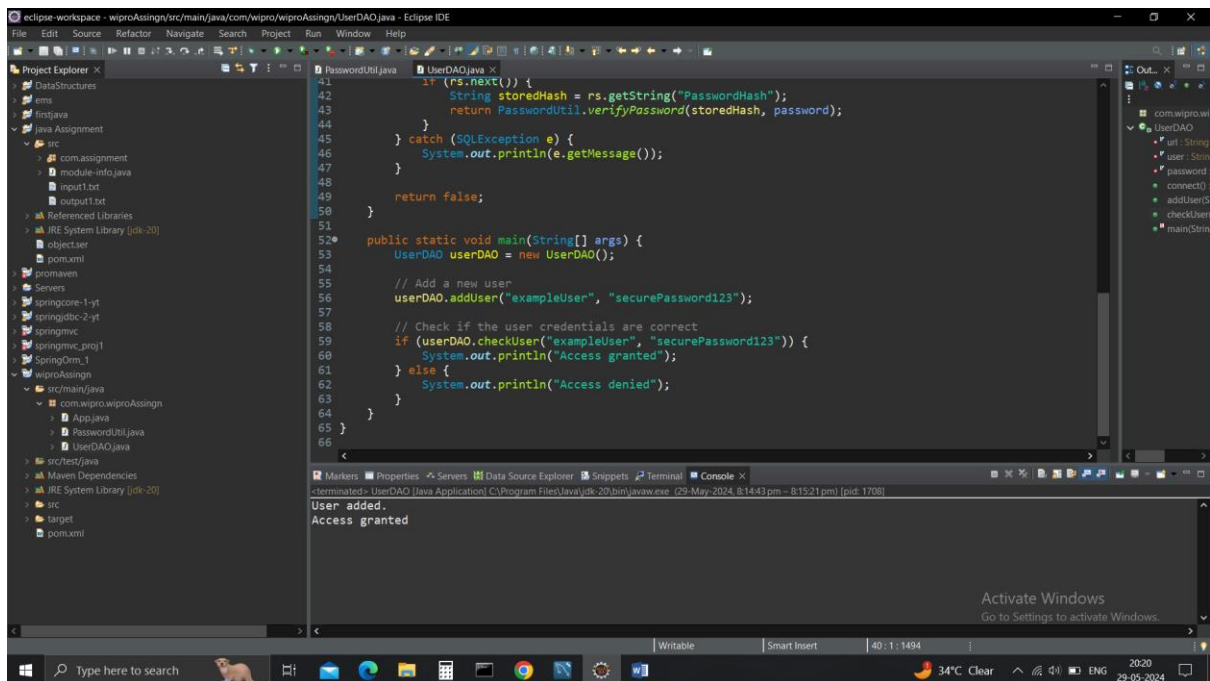
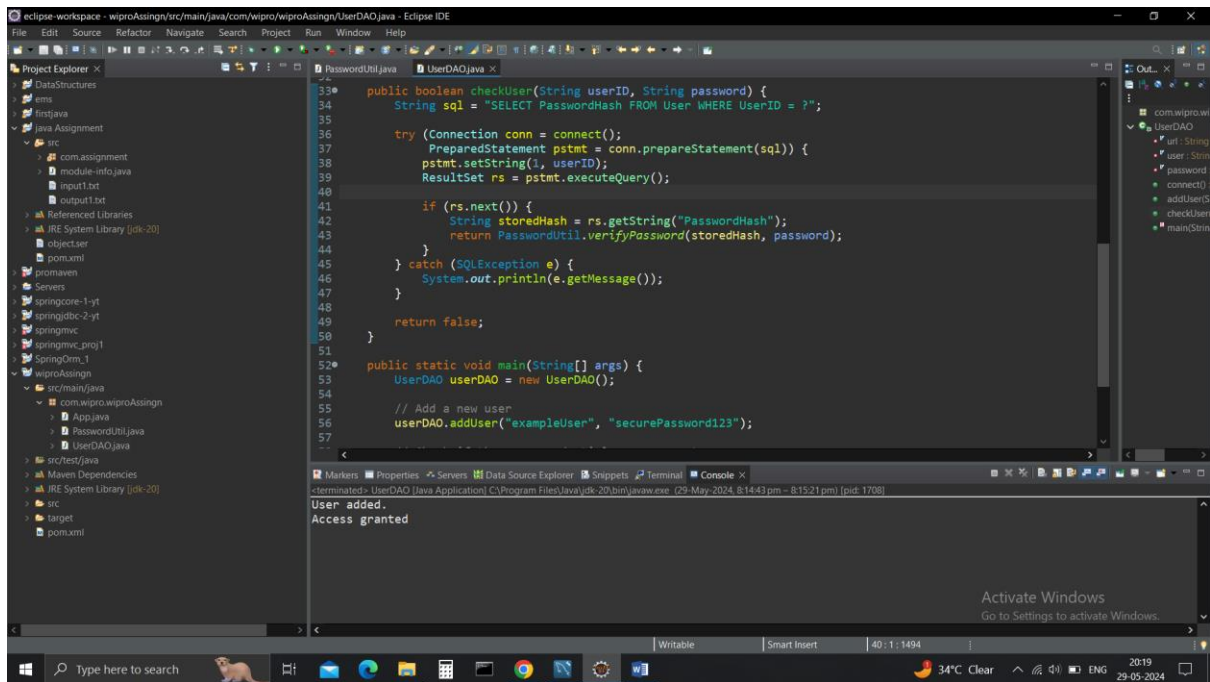
    private final String url = "jdbc:mysql://localhost:3306/userdb";
    private final String user = "root";
    private final String password = "9309562444"; // replace with your MySQL root password

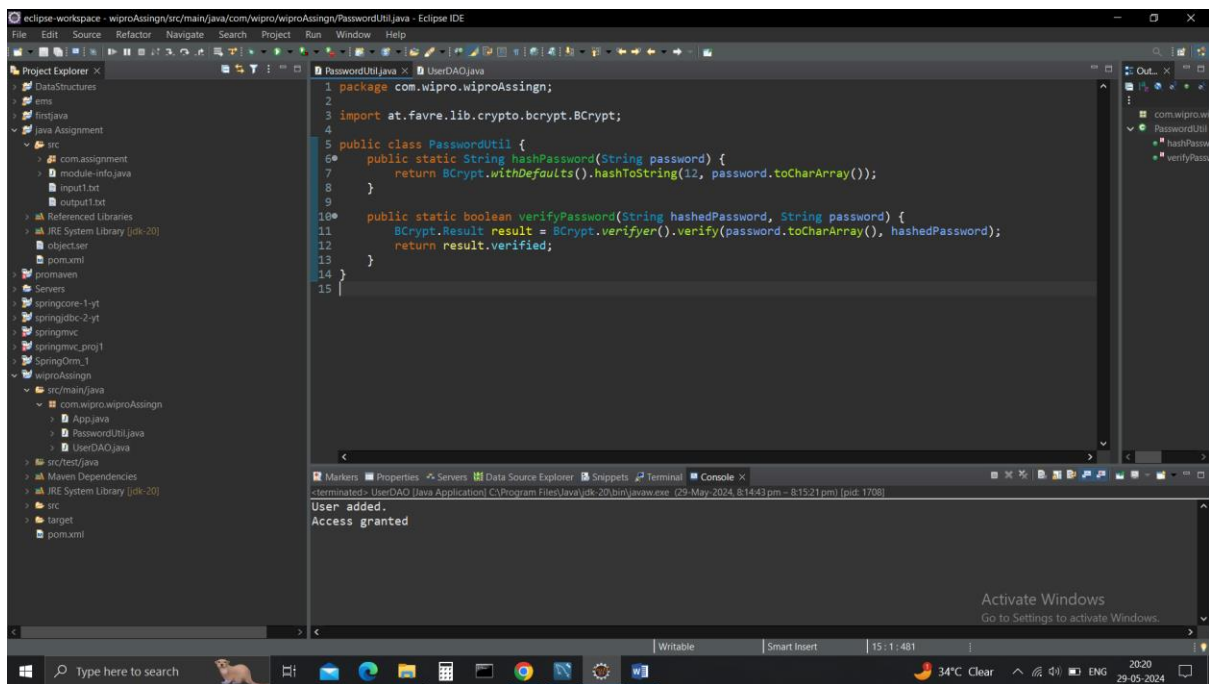
    public Connection connect() throws SQLException {
        return DriverManager.getConnection(url, user, password);
    }

    public void addUser(String userID, String password) {
        String hashedPassword = PasswordUtil.hashPassword(password);
        String sql = "INSERT INTO User(userID, PasswordHash) VALUES(?, ?)";

        try (Connection conn = connect();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, userID);
            pstmt.setString(2, hashedPassword);
            pstmt.executeUpdate();
            System.out.println("User added.");
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

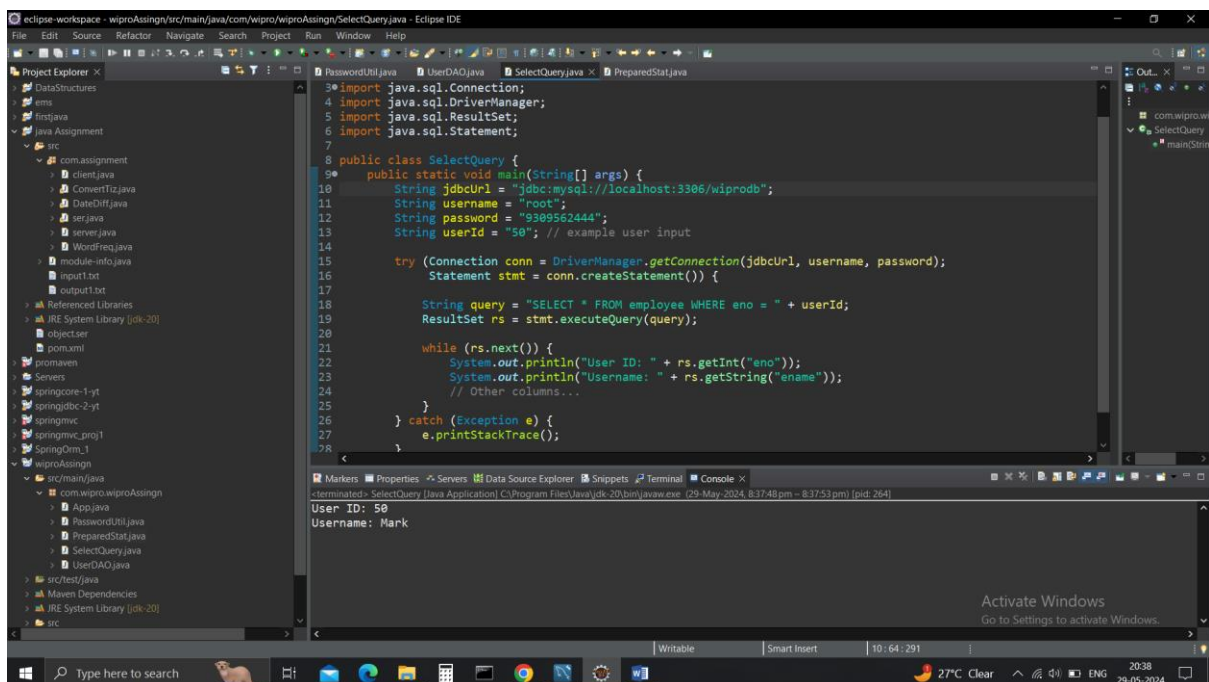
User added.  
Access granted



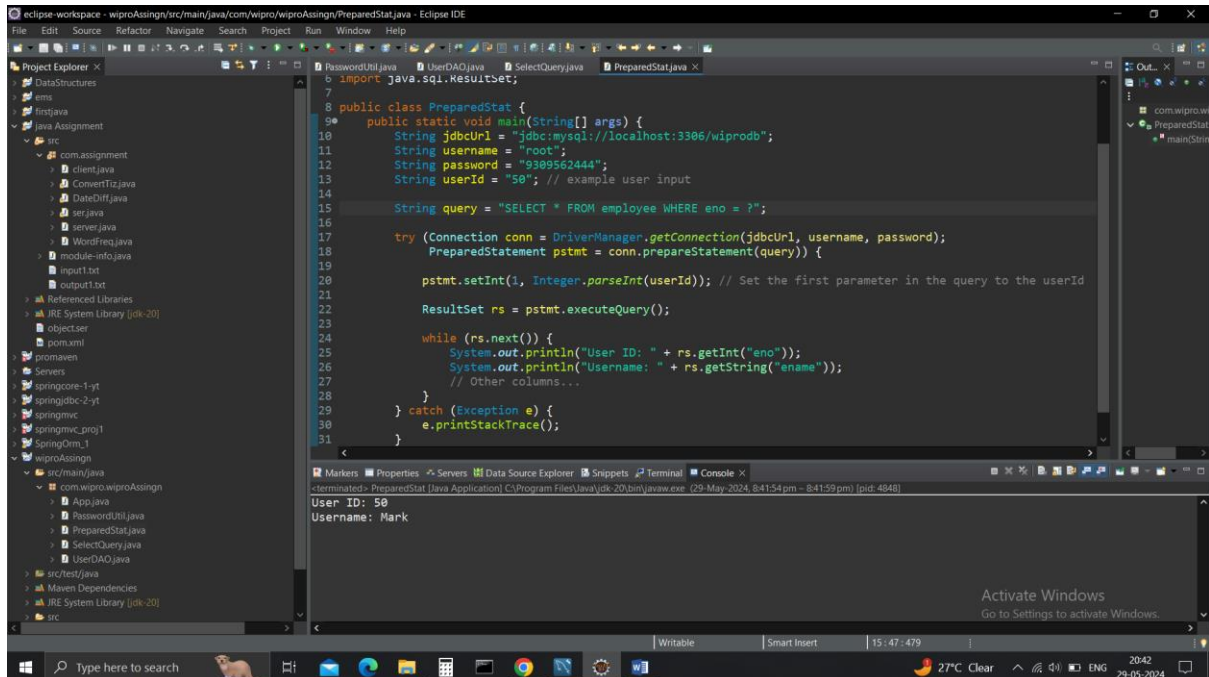


Task 3: Modify the SELECT query program to use PreparedStatement to parameterize the query and prevent SQL injection.

Using Statement



## Using Prepared Statement



The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays a project named 'wiproAssingn' with a package 'com.wiproAssingn'. The main editor window shows the file 'PreparedStat.java' with the following code:

```
1 import java.sql.ResultSet;
2
3 public class PreparedStat {
4     public static void main(String[] args) {
5         String jdbcUrl = "jdbc:mysql://localhost:3306/wiprodb";
6         String username = "root";
7         String password = "9389562444";
8         String userId = "50"; // example user input
9
10        String query = "SELECT * FROM employee WHERE eno = ?";
11
12        try (Connection conn = DriverManager.getConnection(jdbcUrl, username, password);
13             PreparedStatement pstmt = conn.prepareStatement(query)) {
14
15            pstmt.setInt(1, Integer.parseInt(userId)); // Set the first parameter in the query to the userId
16
17            ResultSet rs = pstmt.executeQuery();
18
19            while (rs.next()) {
20                System.out.println("User ID: " + rs.getInt("eno"));
21                System.out.println("Username: " + rs.getString("ename"));
22                // Other columns...
23            }
24        } catch (Exception e) {
25            e.printStackTrace();
26        }
27    }
28 }
```

The Console window at the bottom shows the output of the program:

```
<terminated> PreparedStat [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (29-May-2024, 8:41:54 pm - 8:41:59 pm) [pid: 4848]
User ID: 50
Username: Mark
```

The Windows taskbar at the bottom shows the date and time as 20:42 on 29-05-2024, along with system icons for temperature, network, and volume.