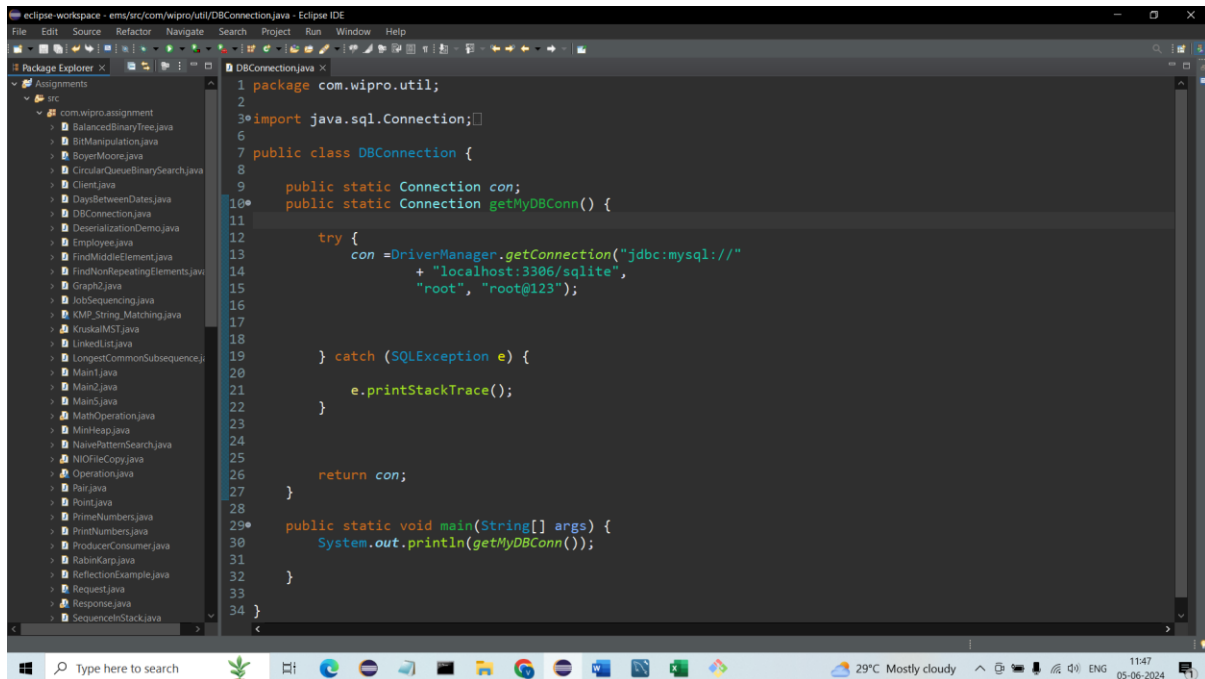


Task 1: Establishing Database Connections

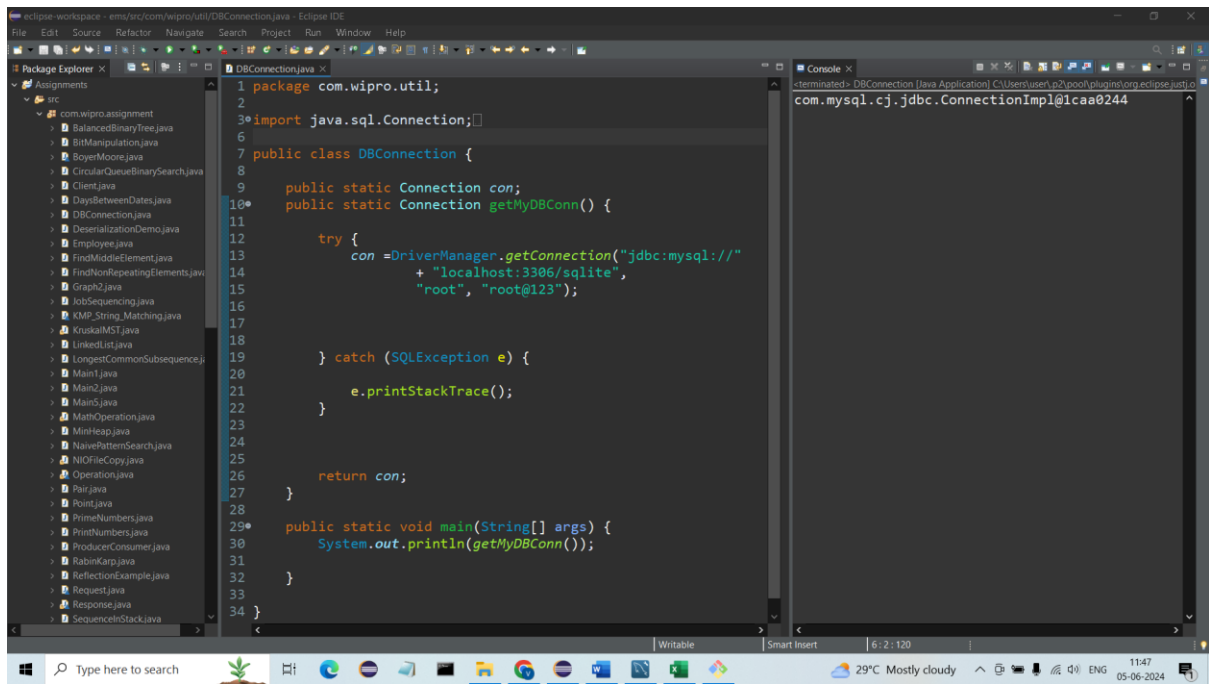
Write a Java program that connects to a SQLite database and prints out the connection object to confirm successful connection.

Program: -

A screenshot of the Eclipse IDE interface. The Package Explorer on the left shows a project named 'com.wipro.assignment' with various Java files. The main editor window displays the code for 'DBConnection.java'. 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 SQLite database at 'localhost:3306/sqlite' with 'root' credentials. A 'try-catch' block handles 'SQLException'. The 'main' method calls 'getMyDBConn()' and prints the result.

```
1 package com.wipro.util;
2
3 import java.sql.Connection;
4
5
6
7 public class DBConnection {
8
9     public static Connection con;
10    public static Connection getMyDBConn() {
11
12        try {
13            con = DriverManager.getConnection("jdbc:mysql://"
14                + "localhost:3306/sqlite",
15                "root", "root@123");
16
17        } catch (SQLException e) {
18
19            e.printStackTrace();
20        }
21
22        return con;
23    }
24
25    public static void main(String[] args) {
26        System.out.println(getMyDBConn());
27    }
28
29 }
30
31
32
33
34 }
```

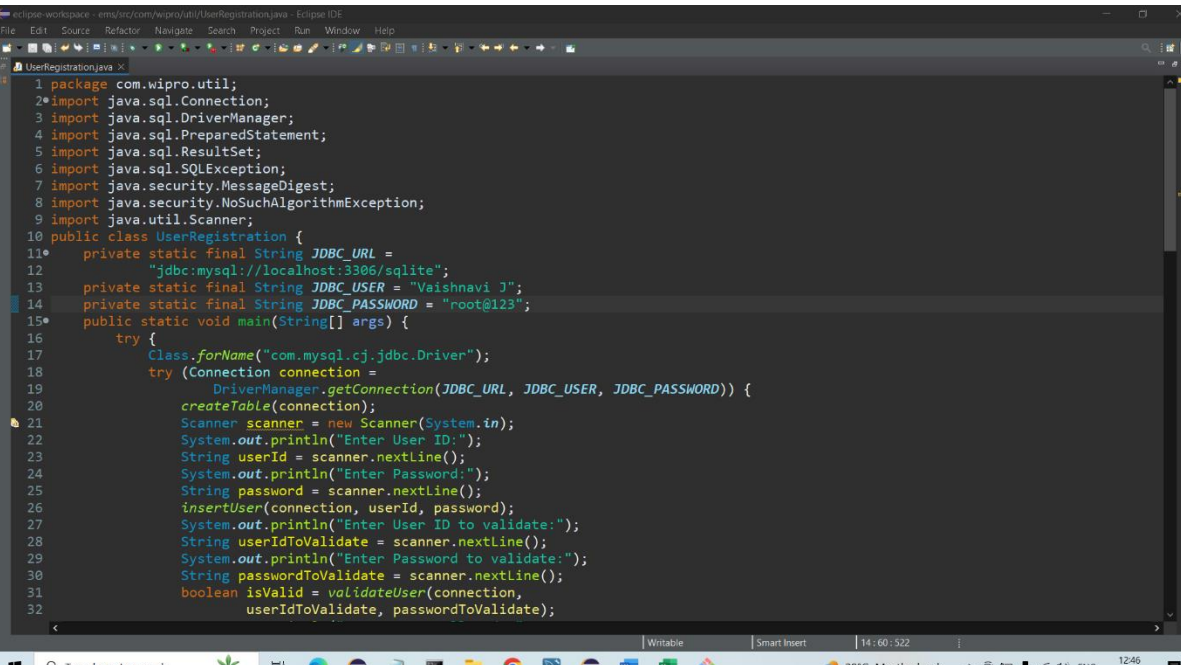
Output: -



Task 2: SQL Queries using JDBC

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.

Program:



```

1 package com.wipro.util;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.security.MessageDigest;
8 import java.security.NoSuchAlgorithmException;
9 import java.util.Scanner;
10 public class UserRegistration {
11     private static final String JDBC_URL =
12         "jdbc:mysql://localhost:3306/sqlite";
13     private static final String JDBC_USER = "Vaishnavi J";
14     private static final String JDBC_PASSWORD = "root@123";
15     public static void main(String[] args) {
16         try {
17             Class.forName("com.mysql.cj.jdbc.Driver");
18             try (Connection connection =
19                 DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD)) {
20                 createTable(connection);
21                 Scanner scanner = new Scanner(System.in);
22                 System.out.println("Enter User ID:");
23                 String userId = scanner.nextLine();
24                 System.out.println("Enter Password:");
25                 String password = scanner.nextLine();
26                 insertUser(connection, userId, password);
27                 System.out.println("Enter User ID to validate:");
28                 String userIdToValidate = scanner.nextLine();
29                 System.out.println("Enter Password to validate:");
30                 String passwordToValidate = scanner.nextLine();
31                 boolean isValid = validateUser(connection,
32                     userIdToValidate, passwordToValidate);

```

```
eclipse-workspace - ems/hrz.com/wipro/utl/UserRegistration.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

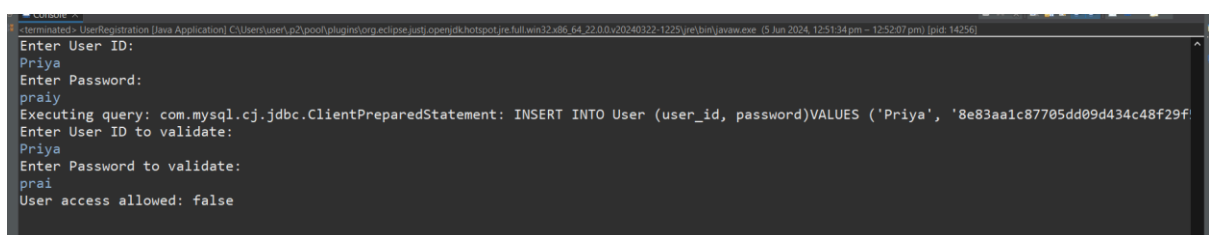
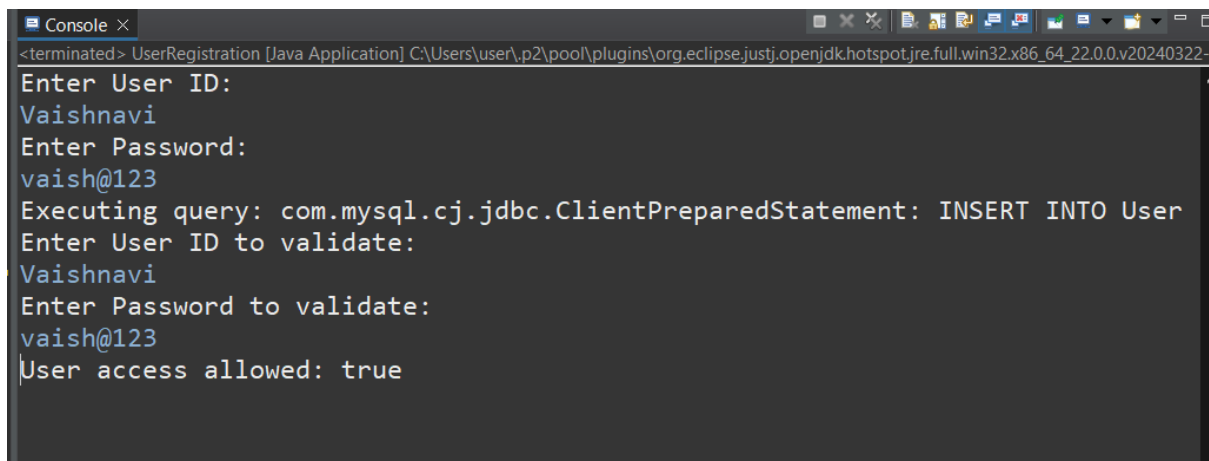
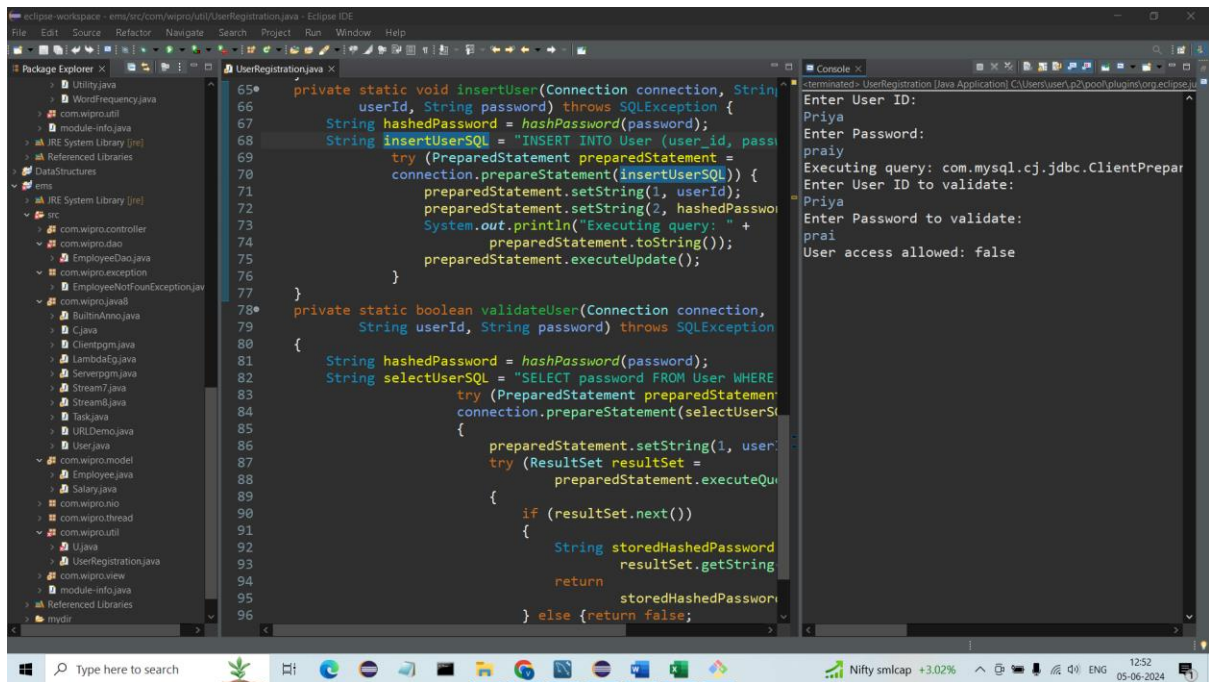
UserRegistration.java x
32         userIdToValidate, passwordToValidate);
33         System.out.println("User access allowed: " +
34             isValid);
35     } catch (SQLException e) {
36         e.printStackTrace();
37     }
38     } catch (ClassNotFoundException e) {
39         e.printStackTrace();
40     }
41 }
42 private static void createTable(Connection connection) throws
43     SQLException {
44     String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
45         + "user_id VARCHAR(255) PRIMARY KEY,"
46         + "password VARCHAR(255) NOT NULL)";
47     try (PreparedStatement preparedStatement =
48         connection.prepareStatement(createTableSQL)) {
49         preparedStatement.execute();
50     }
51 }
52 private static String hashPassword(String password) {
53     try {
54         MessageDigest md = MessageDigest.getInstance("SHA-256");
55         byte[] hashedPassword = md.digest(password.getBytes());
56         StringBuilder sb = new StringBuilder();
57         for (byte b : hashedPassword) {
58             sb.append(String.format("%02x", b));
59         }
60         return sb.toString();
61     } catch (NoSuchAlgorithmException e) {
62         throw new RuntimeException(e);
63     }
64 }

Writtable Smart Insert 63:10:2326 29°C Mostly cloudy 12:46 05-06-2024
```

```
eclipse-workspace - ems/hrz.com/wipro/utl/UserRegistration.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

UserRegistration.java x
65 private static void insertUser(Connection connection, String
66     userId, String password) throws SQLException {
67     String hashedPassword = hashPassword(password);
68     String insertUserSQL = "INSERT INTO User (user_id, password) VALUES (?, ?)";
69     try (PreparedStatement preparedStatement =
70         connection.prepareStatement(insertUserSQL)) {
71         preparedStatement.setString(1, userId);
72         preparedStatement.setString(2, hashedPassword);
73         System.out.println("Executing query: " +
74             preparedStatement.toString());
75         preparedStatement.executeUpdate();
76     }
77 }
78 private static boolean validateUser(Connection connection,
79     String userId, String password) throws SQLException
80 {
81     String hashedPassword = hashPassword(password);
82     String selectUserSQL = "SELECT password FROM User WHERE user_id = ?";
83     try (PreparedStatement preparedStatement =
84         connection.prepareStatement(selectUserSQL))
85     {
86         preparedStatement.setString(1, userId);
87         try (ResultSet resultSet =
88             preparedStatement.executeQuery())
89         {
90             if (resultSet.next())
91             {
92                 String storedHashedPassword =
93                     resultSet.getString("password");
94                 return
95                     storedHashedPassword.equals(hashedPassword);
96             } else {return false;
97         }
98     }
99 }

Writtable Smart Insert 96:54:3635 Nifty midcap +3.00% 12:47 05-06-2024
```



Task 3: PreparedStatement

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

Program:

```
eclipse-workspace - ems/hrz/com/wipro/uttl/UserRegistration.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

UserRegistration.java
1 package com.wipro.uttl;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.security.MessageDigest;
8 import java.security.NoSuchAlgorithmException;
9 import java.util.Scanner;
10 public class UserRegistration {
11     private static final String JDBC_URL =
12         "jdbc:mysql://localhost:3306/sqlite";
13     private static final String JDBC_USER = "Vaishnavi J";
14     private static final String JDBC_PASSWORD = "root@123";
15     public static void main(String[] args) {
16         try {
17             Class.forName("com.mysql.cj.jdbc.Driver");
18             try (Connection connection =
19                 DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD)) {
20                 createTable(connection);
21                 Scanner scanner = new Scanner(System.in);
22                 System.out.println("Enter User ID:");
23                 String userId = scanner.nextLine();
24                 System.out.println("Enter Password:");
25                 String password = scanner.nextLine();
26                 insertUser(connection, userId, password);
27                 System.out.println("Enter User ID to validate:");
28                 String userIdToValidate = scanner.nextLine();
29                 System.out.println("Enter Password to validate:");
30                 String passwordToValidate = scanner.nextLine();
31                 boolean isValid = validateUser(connection,
32                     userIdToValidate, passwordToValidate);
33             }
34         } catch (SQLException e) {
35             e.printStackTrace();
36         } catch (ClassNotFoundException e) {
37             e.printStackTrace();
38         }
39     }
40     private static void createTable(Connection connection) throws
41         SQLException {
42         String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
43             + "user_id VARCHAR(255) PRIMARY KEY,"
44             + "password VARCHAR(255) NOT NULL)";
45         try (PreparedStatement preparedStatement =
46             connection.prepareStatement(createTableSQL)) {
47             preparedStatement.execute();
48         }
49     }
50     private static String hashPassword(String password) {
51         try {
52             MessageDigest md = MessageDigest.getInstance("SHA-256");
53             byte[] hashedPassword = md.digest(password.getBytes());
54             StringBuilder sb = new StringBuilder();
55             for (byte b : hashedPassword) {
56                 sb.append(String.format("%02x", b));
57             }
58             return sb.toString();
59         } catch (NoSuchAlgorithmException e) {
60             throw new RuntimeException(e);
61         }
62     }
63 }
```

```
32         userIdToValidate, passwordToValidate);
33     System.out.println("User access allowed: " +
34         isValid);
35     } catch (SQLException e) {
36         e.printStackTrace();
37     }
38     } catch (ClassNotFoundException e) {
39         e.printStackTrace();
40     }
41 }
42 private static void createTable(Connection connection) throws
43     SQLException {
44     String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
45         + "user_id VARCHAR(255) PRIMARY KEY,"
46         + "password VARCHAR(255) NOT NULL)";
47     try (PreparedStatement preparedStatement =
48         connection.prepareStatement(createTableSQL)) {
49         preparedStatement.execute();
50     }
51 }
52 private static String hashPassword(String password) {
53     try {
54         MessageDigest md = MessageDigest.getInstance("SHA-256");
55         byte[] hashedPassword = md.digest(password.getBytes());
56         StringBuilder sb = new StringBuilder();
57         for (byte b : hashedPassword) {
58             sb.append(String.format("%02x", b));
59         }
60         return sb.toString();
61     } catch (NoSuchAlgorithmException e) {
62         throw new RuntimeException(e);
63     }
64 }
```



```
eclipse-workspace - ems/hrz/com/wipro/utl/UserRegistration.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

UserRegistration.java
65* private static void insertUser(Connection connection, String
66     userId, String password) throws SQLException {
67     String hashedPassword = hashPassword(password);
68     String insertUserSQL = "INSERT INTO User (user_id, password)VALUES (?, ?)";
69     try (PreparedStatement preparedStatement =
70         connection.prepareStatement(insertUserSQL)) {
71         preparedStatement.setString(1, userId);
72         preparedStatement.setString(2, hashedPassword);
73         System.out.println("Executing query: " +
74             preparedStatement.toString());
75         preparedStatement.executeUpdate();
76     }
77 }
78* private static boolean validateUser(Connection connection,
79     String userId, String password) throws SQLException
80 {
81     String hashedPassword = hashPassword(password);
82     String selectUserSQL = "SELECT password FROM User WHERE user_id = ?";
83     try (PreparedStatement preparedStatement =
84         connection.prepareStatement(selectUserSQL))
85     {
86         preparedStatement.setString(1, userId);
87         try (ResultSet resultSet =
88             preparedStatement.executeQuery())
89         {
90             if (resultSet.next())
91             {
92                 String storedHashedPassword =
93                     resultSet.getString("password");
94                 return
95                     storedHashedPassword.equals(hashedPassword);
96             } else {return false;
97         }
98     }
99 }
```

```
eclipse-workspace - ems/hrz/com/wipro/utl/UserRegistration.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
Utility.java
WordFrequency.java
com.wipro.utl
module-info.java
JRE System Library [jre]
Referenced Libraries
Data Structures
JRE System Library [jre]
src
com.wipro.controller
com.wipro.dao
EmployeeDao.java
com.wipro.exception
EmployeeNotFoundException.java
com.wipro.java8
BuiltInAnnotation.java
C.java
ClientPgm.java
LambdaEg.java
ServerPgm.java
Stream7.java
Stream8.java
Task.java
URLDemo.java
User.java
com.wipro.model
Employee.java
Salary.java
com.wipro.nio
com.wipro.thread
com.wipro.utl
UserRegistration.java
com.wipro.view
module-info.java
Referenced Libraries
mydir

UserRegistration.java
65* private static void insertUser(Connection connection, String
66     userId, String password) throws SQLException {
67     String hashedPassword = hashPassword(password);
68     String insertUserSQL = "INSERT INTO User (user_id, pass
69     try (PreparedStatement preparedStatement =
70         connection.prepareStatement(insertUserSQL)) {
71         preparedStatement.setString(1, userId);
72         preparedStatement.setString(2, hashedPasswo
73         System.out.println("Executing query: " +
74             preparedStatement.toString());
75         preparedStatement.executeUpdate();
76     }
77 }
78* private static boolean validateUser(Connection connection,
79     String userId, String password) throws SQLException
80 {
81     String hashedPassword = hashPassword(password);
82     String selectUserSQL = "SELECT password FROM User WHERE
83     try (PreparedStatement preparedStatement =
84         connection.prepareStatement(selectUserSQL)
85     {
86         preparedStatement.setString(1, user
87         try (ResultSet resultSet =
88             preparedStatement.executeQuery
89         {
90             if (resultSet.next())
91             {
92                 String storedHashedPassword
93                 resultSet.getString
94                 return
95                 storedHashedPassword
96             } else {return false;
97         }
98     }
99 }
```

```
<terminated> UserRegistration [Java Application] C:\Users\user\p2\pool\plugins\org.eclipse.ju
Enter User ID:
Priya
Enter Password:
praia
Executing query: com.mysql.cj.jdbc.ClientPrepar
Enter User ID to validate:
Priya
Enter Password to validate:
praia
User access allowed: false
```

```
<terminated> UserRegistration [Java Application] C:\Users\user\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.0.v20240322-
Enter User ID:
Vaishnavi
Enter Password:
vaish@123
Executing query: com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO User
Enter User ID to validate:
Vaishnavi
Enter Password to validate:
vaish@123
User access allowed: true
```

