

Day 21:

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.

```
package com.wipro.util;

import java.sql.Connection;

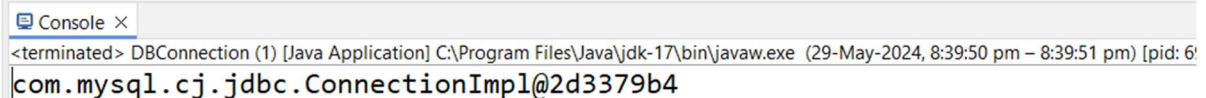
public class DBConnection {

    public static Connection con;
    public static Connection getMyDBConn() {
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/wipro", "root", "Sreenath@#512");
        } catch (SQLException e) {

            e.printStackTrace();
        }
        return con;
    }

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

Output:



The screenshot shows a console window titled "Console x" with the following output: `<terminated> DBConnection (1) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (29-May-2024, 8:39:50 pm - 8:39:51 pm) [pid: 6124] com.mysql.cj.jdbc.ConnectionImpl@2d3379b4`

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.

```

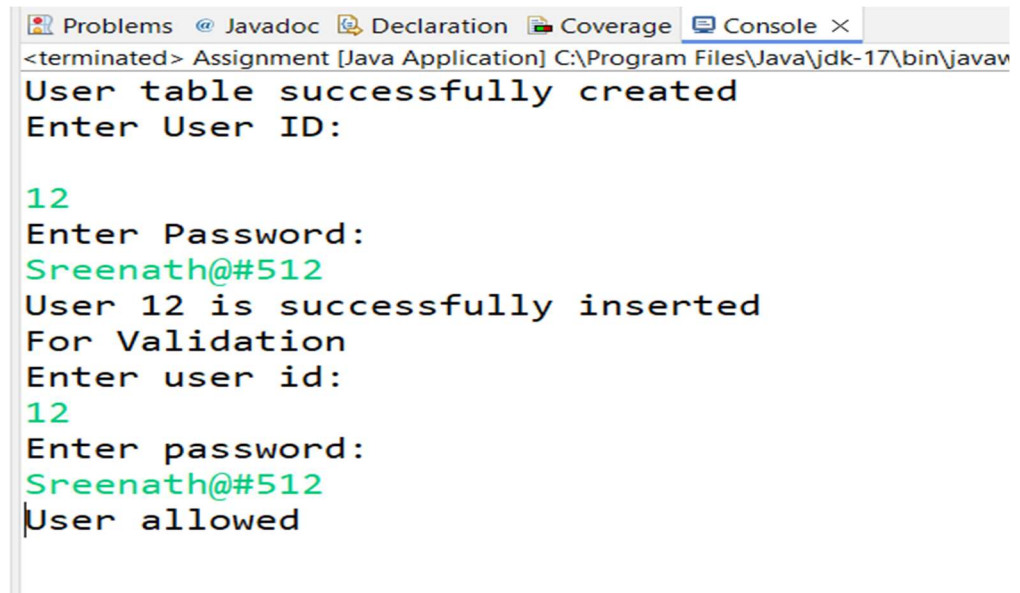
1 package com.assignment.sql;
2
3 import java.sql.Connection;
4
5 public class Assignment {
6
7     public static String createHashedPassword(String password) {
8         return Integer.toString(password.hashCode());
9     }
10
11     public static boolean checkValidation(int userid, String password, Connection con) {
12         String hashedPassword = createHashedPassword(password);
13         String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";
14
15         try (PreparedStatement preparedStatement = con.prepareStatement(sql)) {
16             preparedStatement.setInt(1, userid);
17             preparedStatement.setString(2, hashedPassword);
18             try (ResultSet resultSet = preparedStatement.executeQuery()) {
19                 return resultSet.next();
20             }
21         } catch (SQLException e) {
22             throw new RuntimeException(e);
23         }
24     }
25
26     public static void main(String[] args) {
27         try (Scanner scanner = new Scanner(System.in)) {
28             Connection con = DBConnection.getMyDBConn();
29             String createTableSQL = "CREATE TABLE IF NOT EXISTS User (UserID INT PRIMARY KEY, Password VARCHAR(50))";
30             con.createStatement().executeUpdate(createTableSQL);
31             System.out.println("User table successfully created");
32             System.out.println("Enter User ID: ");
33             int userid = scanner.nextInt();
34             System.out.println("Enter Password: ");
35
36

```

```

37         Connection con = DBConnection.getMyDBConn();
38         String createTableSQL = "CREATE TABLE IF NOT EXISTS User (UserID INT PRIMARY KEY, Password VARCHAR(50))";
39         con.createStatement().executeUpdate(createTableSQL);
40         System.out.println("User table successfully created");
41         System.out.println("Enter User ID: ");
42         int userid = scanner.nextInt();
43         System.out.println("Enter Password: ");
44         String password = scanner.next();
45         String hashedPassword = createHashedPassword(password);
46         String insertUserSQL = "INSERT INTO User (UserID, Password) VALUES(?, ?)";
47         try (PreparedStatement preparedStatement = con.prepareStatement(insertUserSQL)) {
48             preparedStatement.setInt(1, userid);
49             preparedStatement.setString(2, hashedPassword);
50             preparedStatement.executeUpdate();
51             System.out.println("User " + userid + " is successfully inserted");
52         }
53         System.out.println("For Validation");
54         System.out.println("Enter user id: ");
55         userid = scanner.nextInt();
56         System.out.println("Enter password: ");
57         password = scanner.next();
58         if (checkValidation(userid, password, con)) {
59             System.out.println("User allowed");
60         } else {
61             System.out.println("User not allowed");
62         }
63     } catch (SQLException e) {
64         System.out.println("SQL Exception: " + e.getMessage());
65     }
66 }
67

```



The screenshot shows a Java IDE with a console window. The console output is as follows:

```
<terminated> Assignment [Java Application] C:\Program Files\Java\jdk-17\bin\javaw
User table successfully created
Enter User ID:

12
Enter Password:
Sreenath@#512
User 12 is successfully inserted
For Validation
Enter user id:
12
Enter password:
Sreenath@#512
User allowed
```

Task 3: PreparedStatement

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

```
package com.assignment.sql;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;
public class Assignment2 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String sqlStatement = "INSERT INTO USER (UserID, Password) VALUES (?, ?)";
        System.out.println("Enter User ID:");
        int userId = scan.nextInt();

        System.out.println("Enter User Password:");
        String password = scan.next();
        password = Assignment.createHashedPassword(password);

        try {
            Connection con = DBConnection.getMyDBConn();
            PreparedStatement preparedStatement = con.prepareStatement(sqlStatement);

            preparedStatement.setInt(1, userId);
            preparedStatement.setString(2, password);

            preparedStatement.executeUpdate();

            System.out.println("User " + userId + " is successfully inserted.");
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Output:

```
21      preparedStatement.setInt(1, us
<terminated> Assignment2 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Enter User ID:
14
Enter User Password:
Sreenath@#512
User 14 is successfully inserted.
```

Result Grid

Filter Rows:

	UserID	Password
▶	12	-1612313907
	14	-1612313907
	123	1883691757
	234	-482646444
✱	NULL	NULL