Day - 21

1] 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.

Solution:-

Code:-

```
☑ DatabaseConnection.java ×
1 package com.assignments;
  3 import java.sql.Connection;
  4 import java.sql.DriverManager;
  5 import java.sql.SQLException;
  7 public class DatabaseConnection {
        public static void main(String[] args) {
 9⊝
 10
            String url = "jdbc:mysql://localhost:3306/wipdb";
 11
 12
 13
            String username = "root";
 14
            String password = "root1";
 15
 16
 17
            try {
 18
 19
 20
                Class.forName("com.mysql.cj.jdbc.Driver");
 21
 22
 23
                Connection connection = DriverManager.getConnection(url, username, password);
 24
 25
 26
                System.out.println("Connection to MySQL database successful: " + connection);
 27
```

```
13
            String username = "root";
14
           String password = "root1";
15
16
17
           try {
18
19
               Class.forName("com.mysql.cj.jdbc.Driver");
21
22
               Connection connection = DriverManager.getConnection(url, username, password);
23
25
               System.out.println("Connection to MySQL database successful: " + connection);
26
27
28
29
               connection.close();
30
           } catch (ClassNotFoundException e) {
31
               System.err.println("MySQL JDBC driver not found.");
               e.printStackTrace();
32
33
           } catch (SQLException e) {
34
               System.err.println("Error connecting to MySQL database.");
35
               e.printStackTrace();
36
           }
37
       }
38 }
```

Output:-

© Console ×

Sterminated> DatabaseConnection [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\
Connection to MySQL database successful: com.mysql.cj.jdbc.ConnectionImpl@663c9e7a

2] 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.

Solution:-

Code:-

```
☑ Hash.java ×
 1 package com.wipro.Database;
 3 import java.security.MessageDigest;
 4 import java.security.NoSuchAlgorithmException;
 6 public class Hash {
 89
            public static String hashPassword(String password) {
 9
 10
                    MessageDigest md = MessageDigest.getInstance("SHA-256");
 11
                    byte[] hash = md.digest(password.getBytes());
 12
                    StringBuilder hexString = new StringBuilder();
                    for (byte b : hash) {
 13
 14
                        hexString.append(String.format("%02x", b));
 15
                    return hexString.toString();
 16
                } catch (NoSuchAlgorithmException e) {
 17
                    throw new RuntimeException(e);
 18
 19
                }
 20
            }
 21
        }
 22
 23
 24
```

```
1 package com.wipro.Database;
 3⊕
       import java.sql.Connection;
 8
 9
       public class UserDAO {
10
           private static final String URL = "jdbc:mysql://localhost:3306/UserDB";
           private static final String USER = "root";
11
           private static final String PASSWORD = "root1";
12
13
149
           public void addUser(String userID, String password) {
15
               String hashedPassword = Hash.hashPassword(password);
16
               String sql = "INSERT INTO User (UserID, Password) VALUES (?, ?)";
17
               try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
18
19
                    PreparedStatement pstmt = conn.prepareStatement(sql)) {
20
                    pstmt.setString(1, userID);
                   pstmt.setString(2, hashedPassword);
21
22
                   pstmt.executeUpdate();
23
               } catch (SQLException e) {
24
                    e.printStackTrace();
25
26
           }
           public boolean validateUser(String userID, String password) {
27⊝
28
               String hashedPassword = Hash.hashPassword(password);
29
               String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";
30
pstmt.setString(1, useriu);
21
                    pstmt.setString(2, hashedPassword);
22
                    pstmt.executeUpdate();
23
                } catch (SQLException e) {
24
                    e.printStackTrace();
25
                }
26
            }
27⊜
            public boolean validateUser(String userID, String password) {
28
                String hashedPassword = Hash.hashPassword(password);
29
                String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";
30
31
                try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
32
                     PreparedStatement pstmt = conn.prepareStatement(sql)) {
33
                    pstmt.setString(1, userID);
34
                    pstmt.setString(2, hashedPassword);
35
36
                    try (ResultSet rs = pstmt.executeQuery()) {
37
                        return rs.next();
38
39
                } catch (SQLException e) {
40
                    e.printStackTrace();
41
42
                return false;
43
           }
       }
44
45
```

```
ℳ Main.java ×

 1 package com.wipro.Database;
 3
        import java.util.Scanner;
 4
 5
 6
        public class Main {
 7⊝
            public static void main(String[] args) {
 8
                UserDAO userDAO = new UserDAO();
9
                Scanner scanner = new Scanner(System.in);
10
                System.out.print("Enter UserID: ");
11
12
                String userID = scanner.nextLine();
13
14
                System.out.print("Enter Password: ");
                String password = scanner.nextLine();
15
16
17
                userDAO.addUser(userID, password); // Adding a user (for testing)
19
                if (userDAO.validateUser(userID, password)) {
20
                    System.out.println("Access granted.");
21
                    System.out.println("Access denied.");
22
23
           }
24
        }
25
26
```

Output:-

```
Console X

<terminated> Main [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20

Enter UserID: root

Enter Password: root1

Access granted.
```

3] Task 3: PreparedStatement

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

Solution:-

Code:-

```
UserRegistration.java ×
   DataStructure/src/com/assignments/UserRegistration.java
         import java.sql.Connection;
         import java.sql.DriverManager;
        import java.sql.PreparedStatement;
  6
       import java.sql.ResultSet;
  7
         import java.sql.SQLException;
  8
         import java.security.MessageDigest;
         import java.security.NoSuchAlgorithmException;
 10
         import java.util.Scanner;
 11
         public class UserRegistration {
             private static final String JDBC_URL =
 12⊖
                     "jdbc:mysql://localhost:3306/data1";
 13
             private static final String JDBC_USER = "root";
 15
             private static final String JDBC_PASSWORD = "root1";
 16⊜
             public static void main(String[] args) {
 17
                     Class.forName("com.mysql.cj.jdbc.Driver");
 18
                     try (Connection connection =
                          DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD)) {
 21
                          createTable(connection);
 22
                          Scanner scanner = new Scanner(System.in);
                          System.out.println("Enter User ID:");
 23
                          String userId = scanner.nextLine();
 25
                          System.out.println("Enter Password:");
 26
                          String password = scanner.nextLine();
```

```
create (able (connection);
 21
22
                         Scanner scanner = new Scanner(System.in);
 23
                         System.out.println("Enter User ID:");
                         String userId = scanner.nextLine();
  24
  25
                         System.out.println("Enter Password:");
  26
                         String password = scanner.nextLine();
                         insertUser(connection, userId, password);
  27
  28
                         System.out.println("Enter User ID to validate:");
 29
                         String userIdToValidate = scanner.nextLine();
 30
                         System.out.println("Enter Password to validate:");
 31
                         String passwordToValidate = scanner.nextLine();
  32
                         boolean isValid = validateUser(connection,
 33
                                 userIdToValidate, passwordToValidate);
  34
                         System.out.println("User access allowed: " +
                                 isValid);
  35
  36
                     } catch (SQLException e) {
  37
                         e.printStackTrace();
  38
  39
                 } catch (ClassNotFoundException e) {
  40
                     e.printStackTrace();
  41
  42
  43⊜
             private static void createTable(Connection connection) throws
  44
             SQLException {
  45
                 String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
  46
                         + "user_id VARCHAR(255) PRIMARY KEY,"
 47
                         + "password VARCHAR(255) NOT NULL)";
UserRegistration.java ×
 44
             SQLEXCEPTION {
 45
                 String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
 46
                         + "user_id VARCHAR(255) PRIMARY KEY,"
 47
                         + "password VARCHAR(255) NOT NULL)";
 48
                 try (PreparedStatement preparedStatement =
 49
                         connection.prepareStatement(createTableSQL)) {
 50
                     preparedStatement.execute();
 51
                 }
 52
             }
 53⊜
             private static String hashPassword(String password) {
 54
                 try {
 55
                     MessageDigest md = MessageDigest.getInstance("SHA-256");
 56
                     byte[] hashedPassword = md.digest(password.getBytes());
 57
                     StringBuilder sb = new StringBuilder();
 58
                     for (byte b : hashedPassword) {
 59
                         sb.append(String.format("%02x", b));
 60
                     }
                     return sb.toString();
 61
 62
                 } catch (NoSuchAlgorithmException e) {
 63
                     throw new RuntimeException(e);
                 }
 64
```

private static void insertUser(Connection connection, String

String hashedPassword = hashPassword(password);

userId, String password) throws SQLException {

try (PreparedStatement preparedStatement =

String insertUserSQL = "INSERT INTO User (user_id, password)VALUES (?, ?)";

65

66⁶

68

69 70

```
bb<sup>c</sup>
             private static void insertUser(Connection connection, String
 67
                     userId, String password) throws SQLException {
 68
                 String hashedPassword = hashPassword(password);
 69
                 String insertUserSQL = "INSERT INTO User (user_id, password)VALUES (?, ?)";
 70
                         try (PreparedStatement preparedStatement =
 71
                         connection.prepareStatement(insertUserSQL)) {
 72
                             preparedStatement.setString(1, userId);
 73
                             preparedStatement.setString(2, hashedPassword);
 74
                             System.out.println("Executing query: " +
 75
                                      preparedStatement.toString());
 76
                             preparedStatement.executeUpdate();
                         }
 77
 78
 79⊜
             private static boolean validateUser(Connection connection,
                     String userId, String password) throws SQLException
 80
 81
 82
                 String hashedPassword = hashPassword(password);
                 String selectUserSQL = "SELECT password FROM User WHERE user_id = ?";
 83
 84
                                 try (PreparedStatement preparedStatement =
 85
                                 connection.prepareStatement(selectUserSQL))
 86
 87
                                      preparedStatement.setString(1, userId);
 88
                                      try (ResultSet resultSet =
 89
                                              preparedStatement.executeQuery())
 90
 91
                                          if (resultSet.next())
 92
🔃 UserRegistration.java 💢
  78
  79e
             private static boolean validateUser(Connection connection,
  80
                      String userId, String password) throws SQLException
  81
             {
  82
                 String hashedPassword = hashPassword(password);
                 String selectUserSQL = "SELECT password FROM User WHERE user id = ?";
  83
  84
                                  try (PreparedStatement preparedStatement =
  85
                                  connection.prepareStatement(selectUserSQL))
  86
  87
                                       preparedStatement.setString(1, userId);
  88
                                       try (ResultSet resultSet =
  89
                                               preparedStatement.executeQuery())
  90
                                       {
                                           if (resultSet.next())
  91
  92
                                           {
  93
                                               String storedHashedPassword =
  94
                                                        resultSet.getString("password");
  95
  96
                                               storedHashedPassword.equals(hashedPassword);
  97
                                             else {return false;
  98
 99
                                       }
 100
                                  }
 101
             }
102
103
104
```

Output:-

```
Console X
<terminated> UserRegistration [Java Application] C:\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe (Jun 22, 2024, 4:01::
Enter User ID:
Enter Password:
shweta
Executing query: com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO User (user_id, password)VALUES ('shweta',
'6a85d736a79a0277282ca339dd8e4156dd40b4ecdabffd4a3f5a31e859551f99')
Enter User ID to validate:
Enter Password to validate:
shweta
User access allowed: true
                                                                                                    ■ Console ×
<terminated> UserRegistration [Java Application] C.\Users\Skynet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe (Jun 22, 2024, 4:07:55
Enter User ID:
purva
Enter Password:
purva
Executing query: com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO User (user_id, password)VALUES ('purva',
'9689be17e5b02f777aeb69ea61f0ce4138e6ec65ee8e37a071d9491b550841b1')
Enter User ID to validate:
Enter Password to validate:
User access allowed: false
```