

## Manjula Nannuri

### Day 22\_Assignment

#### 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.

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class Main {

    public static void main(String[] args) {

        String url = "jdbc:sqlite:sample.db";

        try (Connection conn = DriverManager.getConnection(url)) {

            if (conn != null) {

                System.out.println("Connected to the SQLite database");

                System.out.println(conn);

            }

        } catch (SQLException e) {

            System.out.println(e.getMessage());

        }

    }

}
```

## 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.

First, ensure you have a database (e.g., MySQL, PostgreSQL) set up and accessible from your Java application.

```
( UserIDVARCHAR(300) PRIMARY
KEY,
PasswordHashVARCHAR(300) NOT NULL
);

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class PasswordUtil {

    public static String hashPassword(String password) {

        try {

            MessageDigest md = MessageDigest.getInstance("SHA-256");

            byte[] hashBytes = md.digest(password.getBytes());

            StringBuilder sb = new StringBuilder();

            for (byte b :hashBytes) {
```

```

sb.append(String.format("%02x", b));

    }

    return sb.toString();

} catch (NoSuchAlgorithmException e) {

    throw new RuntimeException(e);

}

}

}

import java.sql.*;

import java.util.Scanner;

public class UserAuth {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/your_database_name";

    private static final String DB_USER = "your_db_username";

    private static final String DB_PASSWORD = "your_db_password";

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter User ID: ");

        String userId = scanner.nextLine();

        System.out.print("Enter Password: ");

        String password = scanner.nextLine();

        String hashedPassword = PasswordUtil.hashPassword(password);

```

```

        if (authenticateUser(userId, hashedPassword)) {
System.out.println("Access granted.");

        } else {

System.out.println("Access denied.");

        }
scanner.close();

    }

    public static boolean authenticateUser(String userId, String hashedPassword) {

        String query = "SELECT PasswordHash FROM User WHERE UserID= ?";

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString(1, userId);

            ResultSet rs = stmt.executeQuery();

                if (rs.next()) {

                    String storedHash = rs.getString("PasswordHash");

                    return storedHash.equals(hashedPassword);

                }

            } catch (SQLException e) {

e.printStackTrace();

            }

            return false;

```

```
}  
  
}
```

### **Output:**

Test data inserted successfully.

Enter User ID: user1

Enter Password: wrongpassword

Access denied.

### **Task 3: PreparedStatement**

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

```
import java.sql.*;  
  
import java.util.Scanner;  
  
public class UserAuth {  
  
    private static final String DB_URL = "jdbc:mysql://localhost:3306/your_database_name";  
  
    private static final String DB_USER = "your_db_username";  
  
    private static final String DB_PASSWORD = "your_db_password";  
  
    public static void main(String[] args) {  
  
        Scanner scanner = new Scanner(System.in);  
  
        insertTestData();  
  
        System.out.print("Enter User ID: ");  
  
        String userId = scanner.nextLine();
```

```

System.out.print("Enter Password: ");

String password = scanner.nextLine();

String hashedPassword = PasswordUtil.hashPassword(password);

    if (authenticateUser(userId, hashedPassword)) {
System.out.println("Access granted.");

        } else {
System.out.println("Access denied.");

        }

scanner.close();

    }

    public static boolean authenticateUser(String userId, String hashedPassword) {

        String query = "SELECT PasswordHash FROM User WHERE UserID= ?";

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setString(1, userId);

            ResultSets = stmt.executeQuery();

            if (rs.next()) {

                String storedHash = rs.getString("PasswordHash");

                return storedHash.equals(hashedPassword);

            }

```

```

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

        }

        return false;
    }

    public static void insertTestData() {

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);

            Statement stmt = conn.createStatement()) {

            String passwordHash1 = PasswordUtil.hashPassword("password1");

            String passwordHash2 = PasswordUtil.hashPassword("password2");

            String sql1 = "INSERT INTO User (UserID, PasswordHash) VALUES ('user1', '" +
passwordHash1 + "')";

            String sql2 = "INSERT INTO User (UserID, PasswordHash) VALUES ('user2', '" +
passwordHash2 + "')";

            stmt.executeUpdate(sql1);

            stmt.executeUpdate(sql2);

            System.out.println("Test data inserted successfully.");

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

        }

    }
}

```

**Output:**

Test data inserted successfully.

Enter User ID: user1

Enter Password: password1

Access granted.