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

```java
package com.assignments;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/wipdb";


        String username = "root";
        String password = "root1";


        try {

            Class.forName("com.mysql.cj.jdbc.Driver");


            Connection connection = DriverManager.getConnection(url, username, password);


            System.out.println("Connection to MySQL database successful: " + connection);
```

DatabaseConnection.java

```java
        String username = "root";
        String password = "root1";


        try {

            Class.forName("com.mysql.cj.jdbc.Driver");


            Connection connection = DriverManager.getConnection(url, username, password);


            System.out.println("Connection to MySQL database successful: " + connection);


            connection.close();
        } catch (ClassNotFoundException e) {
            System.err.println("MySQL JDBC driver not found.");
            e.printStackTrace();
        } catch (SQLException e) {
            System.err.println("Error connecting to MySQL database.");
            e.printStackTrace();
        }
    }
}
```

Output :-

Console

```
<terminated> 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

```java
package com.wipro.Database;

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

public class Hash {

        public static String hashPassword(String password) {
            try {
                MessageDigest md = MessageDigest.getInstance("SHA-256");
                byte[] hash = md.digest(password.getBytes());
                StringBuilder hexString = new StringBuilder();
                for (byte b : hash) {
                    hexString.append(String.format("%02x", b));
                }
                return hexString.toString();
            } catch (NoSuchAlgorithmException e) {
                throw new RuntimeException(e);
            }
        }
    }
```

UserDAO.java

```java
package com.wipro.Database;

    import java.sql.Connection;

    public class UserDAO {
        private static final String URL = "jdbc:mysql://localhost:3306/UserDB";
        private static final String USER = "root";
        private static final String PASSWORD = "root1";

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

            try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
                 PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setString(1, userID);
                pstmt.setString(2, hashedPassword);
                pstmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        public boolean validateUser(String userID, String password) {
            String hashedPassword = Hash.hashPassword(password);
            String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";

```

UserDAO.java

```java
                pstmt.setString(1, userID);
                pstmt.setString(2, hashedPassword);
                pstmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        public boolean validateUser(String userID, String password) {
            String hashedPassword = Hash.hashPassword(password);
            String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";

            try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
                 PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setString(1, userID);
                pstmt.setString(2, hashedPassword);

                try (ResultSet rs = pstmt.executeQuery()) {
                    return rs.next();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return false;
        }
    }

```

Main.java

```java
package com.wipro.Database;


    import java.util.Scanner;

    public class Main {
        public static void main(String[] args) {
            UserDAO userDAO = new UserDAO();
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter UserID: ");
            String userID = scanner.nextLine();

            System.out.print("Enter Password: ");
            String password = scanner.nextLine();

            userDAO.addUser(userID, password);  // Adding a user (for testing)

            if (userDAO.validateUser(userID, password)) {
                System.out.println("Access granted.");
            } else {
                System.out.println("Access denied.");
            }
        }
    }
```

Output :-

Console

```
<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

```java

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
public class UserRegistration {
    private static final String JDBC_URL =
            "jdbc:mysql://localhost:3306/data1";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASSWORD = "root1";
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection connection =
                DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD)) {
                createTable(connection);
                Scanner scanner = new Scanner(System.in);
                System.out.println("Enter User ID:");
                String userId = scanner.nextLine();
                System.out.println("Enter Password:");
                String password = scanner.nextLine();
```

UserRegistration.java

```java
                    createTable(connection);
                    Scanner scanner = new Scanner(System.in);
                    System.out.println("Enter User ID:");
                    String userId = scanner.nextLine();
                    System.out.println("Enter Password:");
                    String password = scanner.nextLine();
                    insertUser(connection, userId, password);
                    System.out.println("Enter User ID to validate:");
                    String userIdToValidate = scanner.nextLine();
                    System.out.println("Enter Password to validate:");
                    String passwordToValidate = scanner.nextLine();
                    boolean isValid = validateUser(connection,
                            userIdToValidate, passwordToValidate);
                    System.out.println("User access allowed: " +
                            isValid);
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
        private static void createTable(Connection connection) throws
        SQLException {
            String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
                    + "user_id VARCHAR(255) PRIMARY KEY,"
                    + "password VARCHAR(255) NOT NULL)";
```

UserRegistration.java

```java
        SQLException {
            String createTableSQL = "CREATE TABLE IF NOT EXISTS User ("
                    + "user_id VARCHAR(255) PRIMARY KEY,"
                    + "password VARCHAR(255) NOT NULL)";
            try (PreparedStatement preparedStatement =
                    connection.prepareStatement(createTableSQL)) {
                preparedStatement.execute();
            }
        }
        private static String hashPassword(String password) {
            try {
                MessageDigest md = MessageDigest.getInstance("SHA-256");
                byte[] hashedPassword = md.digest(password.getBytes());
                StringBuilder sb = new StringBuilder();
                for (byte b : hashedPassword) {
                    sb.append(String.format("%02x", b));
                }
                return sb.toString();
            } catch (NoSuchAlgorithmException e) {
                throw new RuntimeException(e);
            }
        }
        private static void insertUser(Connection connection, String
                userId, String password) throws SQLException {
            String hashedPassword = hashPassword(password);
            String insertUserSQL = "INSERT INTO User (user_id, password)VALUES (?, ?)";
                    try (PreparedStatement preparedStatement =
```

UserRegistration.java

```java
        private static void insertUser(Connection connection, String
                userId, String password) throws SQLException {
            String hashedPassword = hashPassword(password);
            String insertUserSQL = "INSERT INTO User (user_id, password)VALUES (?, ?)";
                    try (PreparedStatement preparedStatement =
                    connection.prepareStatement(insertUserSQL)) {
                        preparedStatement.setString(1, userId);
                        preparedStatement.setString(2, hashedPassword);
                        System.out.println("Executing query: " +
                                preparedStatement.toString());
                        preparedStatement.executeUpdate();
                    }
        }
        private static boolean validateUser(Connection connection,
                String userId, String password) throws SQLException
        {
            String hashedPassword = hashPassword(password);
            String selectUserSQL = "SELECT password FROM User WHERE user_id = ?";
                            try (PreparedStatement preparedStatement =
                            connection.prepareStatement(selectUserSQL))
                            {
                                preparedStatement.setString(1, userId);
                                try (ResultSet resultSet =
                                        preparedStatement.executeQuery())
                                {
                                    if (resultSet.next())
                                    {
```

UserRegistration.java

```java
        }
        private static boolean validateUser(Connection connection,
                String userId, String password) throws SQLException
        {
            String hashedPassword = hashPassword(password);
            String selectUserSQL = "SELECT password FROM User WHERE user_id = ?";
                            try (PreparedStatement preparedStatement =
                            connection.prepareStatement(selectUserSQL))
                            {
                                preparedStatement.setString(1, userId);
                                try (ResultSet resultSet =
                                        preparedStatement.executeQuery())
                                {
                                    if (resultSet.next())
                                    {
                                        String storedHashedPassword =
                                                resultSet.getString("password");
                                        return
                                        storedHashedPassword.equals(hashedPassword);
                                    } else {return false;
                                    }
                                }
                            }
        }
    }
```

Output :-

```
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, 4:01:2
Enter User ID:
shweta
Enter Password:
shweta
Executing query: com.mysql.cj.jdbc.ClientPreparedStatement: INSERT INTO User (user_id, password)VALUES ('shweta',
'6a85d736a79a0277282ca339dd8e4156dd40b4ecdabffd4a3f5a31e859551f99')
Enter User ID to validate:
shweta
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, 4:07:59
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:
purv
Enter Password to validate:
purva
User access allowed: false
```