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.

```java
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:

```
Console ×
<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: 6
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.

KMPPatternSearching.java | Graph.java | *Assignment.java

```java
package com.assignment.sql;

import java.sql.Connection;

public class Assignment {

    public static String createHashedPassword(String password) {
        return Integer.toString(password.hashCode());
    }
    public static boolean checkValidation(int userid, String password, Connection con) {
        String hashedPassword = createHashedPassword(password);
        String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";

        try (PreparedStatement preparedStatement = con.prepareStatement(sql)) {
            preparedStatement.setInt(1, userid);
            preparedStatement.setString(2, hashedPassword);
            try (ResultSet resultSet = preparedStatement.executeQuery()) {
                return resultSet.next();
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in);
             Connection con = DBConnection.getMyDBConn()) {
            String createTableSQL = "CREATE TABLE IF NOT EXISTS User (UserID INT PRIMARY KEY, Password VARCHAR(50))";
            con.createStatement().executeUpdate(createTableSQL);
            System.out.println("User table successfully created");
            System.out.println("Enter User ID: ");
            int userid = scanner.nextInt();
            System.out.println("Enter Password: ");
```


```java
             Connection con = DBConnection.getMyDBConn()) {
            String createTableSQL = "CREATE TABLE IF NOT EXISTS User (UserID INT PRIMARY KEY, Password VARCHAR(50))";
            con.createStatement().executeUpdate(createTableSQL);
            System.out.println("User table successfully created");
            System.out.println("Enter User ID: ");
            int userid = scanner.nextInt();
            System.out.println("Enter Password: ");
```


```java
            String password = scanner.next();
            String hashedPassword = createHashedPassword(password);
            String insertUserSQL = "INSERT INTO User (UserID, Password) VALUES(?, ?)";
            try (PreparedStatement preparedStatement = con.prepareStatement(insertUserSQL)) {
                preparedStatement.setInt(1, userid);
                preparedStatement.setString(2, hashedPassword);
                preparedStatement.executeUpdate();
                System.out.println("User " + userid + " is successfully inserted");
            }
            System.out.println("For Validation");
            System.out.println("Enter user id: ");
            userid = scanner.nextInt();
            System.out.println("Enter password: ");
            password = scanner.next();
            if (checkValidation(userid, password, con)) {
                System.out.println("User allowed");
            } else {
                System.out.println("User not allowed");
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
    }
}
```

```
Problems  @ Javadoc  Declaration  Coverage  Console ×
<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.

```java
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
```

Problems | Javadoc | Declaration | Coverage | Console

```
<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