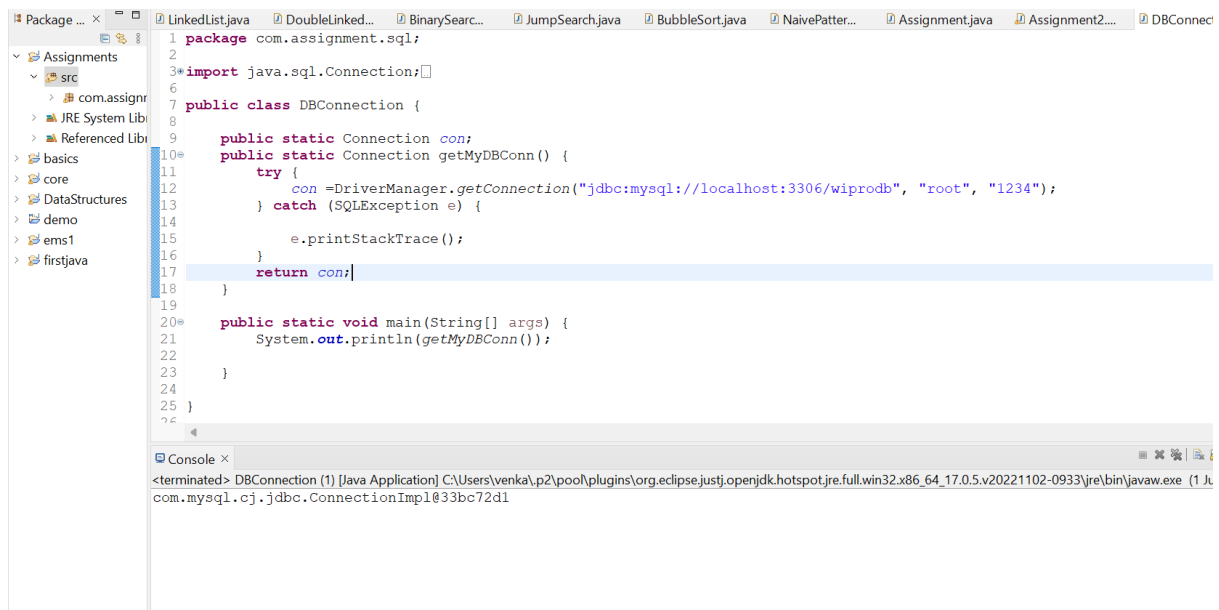


## Task 1: Establishing Database Connections

Write a Java program that connects to a SQLite database and prints out the connection object to confirm successful connection.



```
1 package com.assignment.sql;
2
3 import java.sql.Connection;
4
5 public class DBConnection {
6
7     public static Connection con;
8     public static Connection getMyDBConn() {
9         try {
10             con = DriverManager.getConnection("jdbc:mysql://localhost:3306/wiproddb", "root", "1234");
11         } catch (SQLException e) {
12             e.printStackTrace();
13         }
14         return con;
15     }
16
17     public static void main(String[] args) {
18         System.out.println(getMyDBConn());
19     }
20 }
21
22
23
24
25
```

Console ×

```
<terminated> DBConnection (1) [Java Application] C:\Users\venka\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.17.0.5.v20221102-0933\jre\bin\javaw.exe (1 J
com.mysql.cj.jdbc.ConnectionImpl@33bc72d1
```

## 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
30             String createTableSQL = "CREATE TABLE IF NOT EXISTS User (UserID INT PRIMARY KEY, Password VARCHAR(50))";
31             con.createStatement().executeUpdate(createTableSQL);
32             System.out.println("User table successfully created");
33
34             System.out.println("Enter User ID: ");
35             int userid = scanner.nextInt();
```

```
40
41             System.out.println("Enter Password: ");
42             String password = scanner.next();
43             String hashedPassword = createHashedPassword(password);
44
45             String insertUserSQL = "INSERT INTO User (UserID, Password) VALUES(?, ?)";
46             try (PreparedStatement preparedStatement = con.prepareStatement(insertUserSQL)) {
47                 preparedStatement.setInt(1, userid);
48                 preparedStatement.setString(2, hashedPassword);
49                 preparedStatement.executeUpdate();
50                 System.out.println("User " + userid + " is successfully inserted");
51             }
52
53             System.out.println("For Validation");
54             System.out.println("Enter user id: ");
55             userid = scanner.nextInt();
56
57             System.out.println("Enter password: ");
58             password = scanner.next();
59
60             if (checkValidation(userid, password, con)) {
61                 System.out.println("User allowed");
62             } else {
63                 System.out.println("User not allowed");
64             }
65         } catch (SQLException e) {
66             System.out.println("SQL Exception: " + e.getMessage());
67         }
68     }
69 }
70
```

## Output:

```
User table successfully created
Enter User ID:
1234
Enter Password:
123
User 1234 is successfully inserted
For Validation
Enter user id:
1234
Enter password:
123
User allowed
```

## Task 3: PreparedStatement

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

```
1 package com.assignment.sql;
2
3 import java.sql.Connection;
4
5
6
7
8 public class Assignment2 {
9     public static void main(String[] args) {
10         Scanner scan = new Scanner(System.in);
11
12         String sqlStatement = "INSERT INTO USER (UserID, Password) VALUES (?, ?)";
13
14         System.out.println("Enter User ID:");
15         int userId = scan.nextInt();
16
17         System.out.println("Enter User Password:");
18         String password = scan.next();
19         password = Assignment.createHashedPassword(password);
20
21         try {
22             Connection con = DBConnection.getMyDBConn();
23             PreparedStatement preparedStatement = con.prepareStatement(sqlStatement);
24
25             preparedStatement.setInt(1, userId);
26             preparedStatement.setString(2, password);
27
28             preparedStatement.executeUpdate();
29
30             System.out.println("User " + userId + " is successfully inserted.");
31         } catch (SQLException e) {
32             throw new RuntimeException(e);
33         }
34     }
35 }
36
37
```

## Output:

```
Enter User ID:  
9999  
Enter User Password:  
venkat@123  
User 9999 is successfully inserted.
```

	UserID	Password
▶	1111	1509442
	1234	48690
	9999	-2082698159
*	NULL	NULL