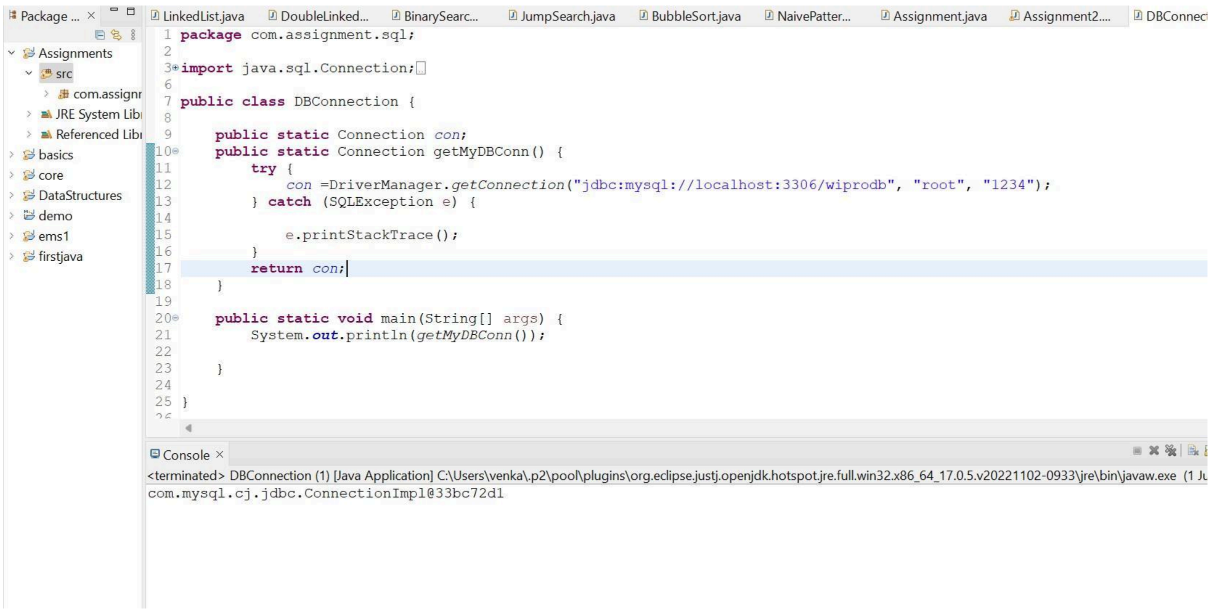
Name- Amit Kumar Parhi

Email- amitkparhi07@gmail.com

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.

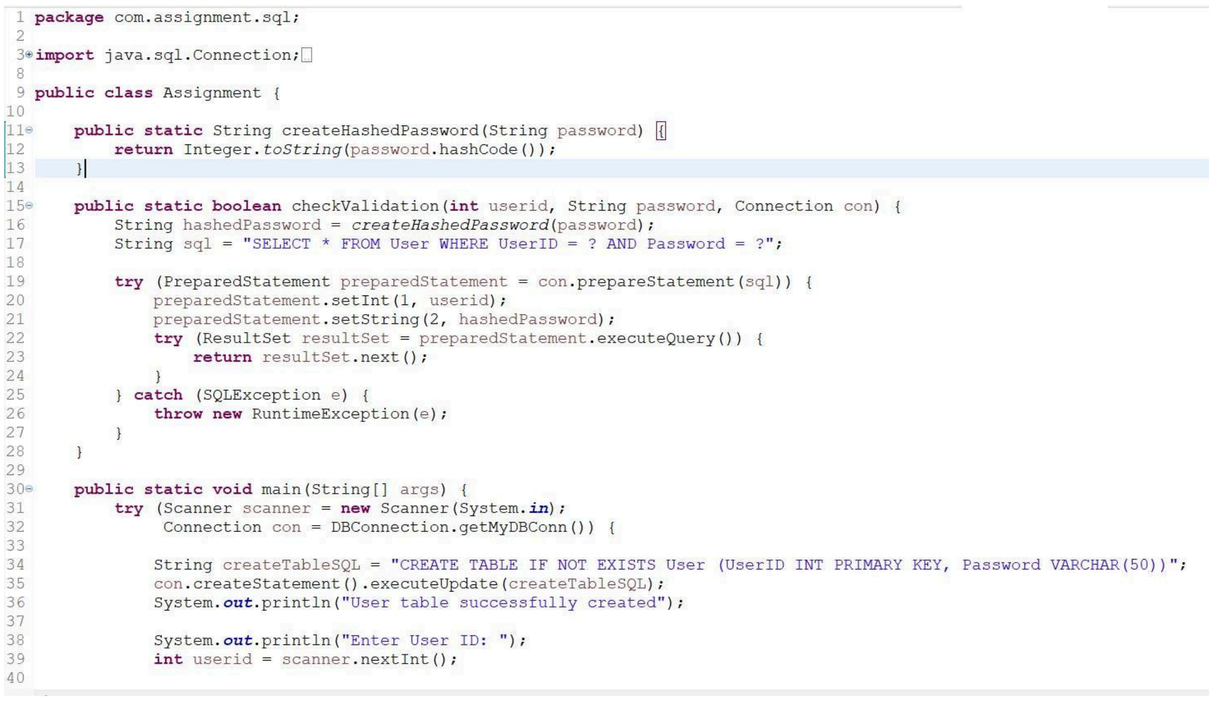


```java
package com.assignment.sql;

import java.sql.Connection;

public class DBConnection {

    public static Connection con;
    public static Connection getMyDBConn() {
        try {
            con =DriverManager.getConnection("jdbc:mysql://localhost:3306/wiprodb", "root", "1234");
        } catch (SQLException e) {

            e.printStackTrace();
        }
        return con;
    }

    public static void main(String[] args) {
        System.out.println(getMyDBConn());

    }

}
```

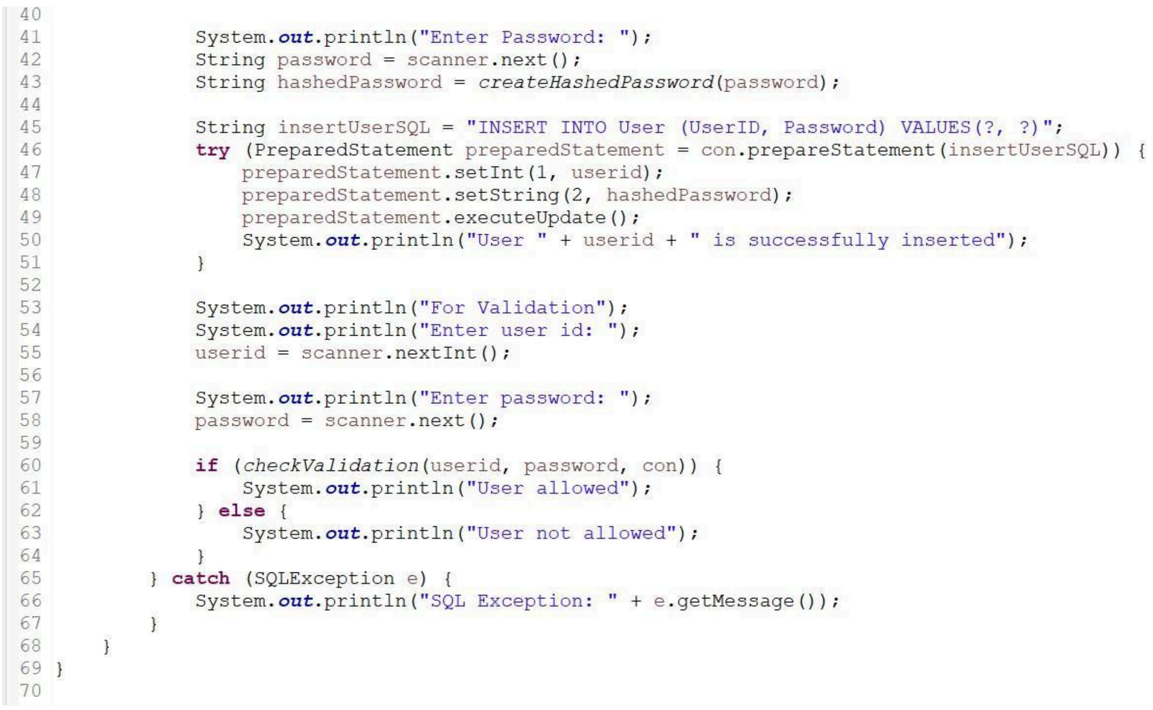
```
<terminated> DBConnection (1) [Java Application] C:\Users\venka\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (1 Ju
com.mysql.cj.jdbc.ConnectionImpl@33bc72d1
```

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.

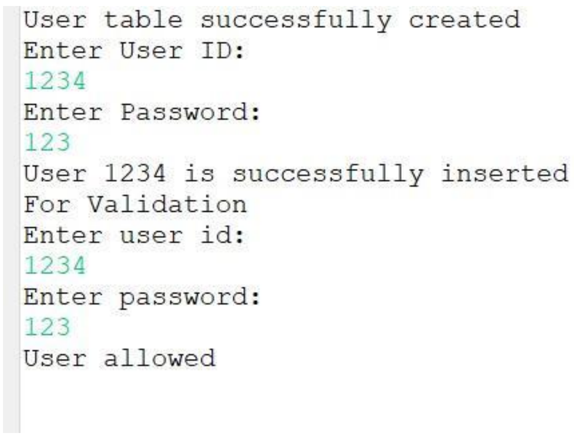


```java
package com.assignment.sql;

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();

```



```java

            System.out.println("Enter Password: ");
            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());
        }
    }
}

```

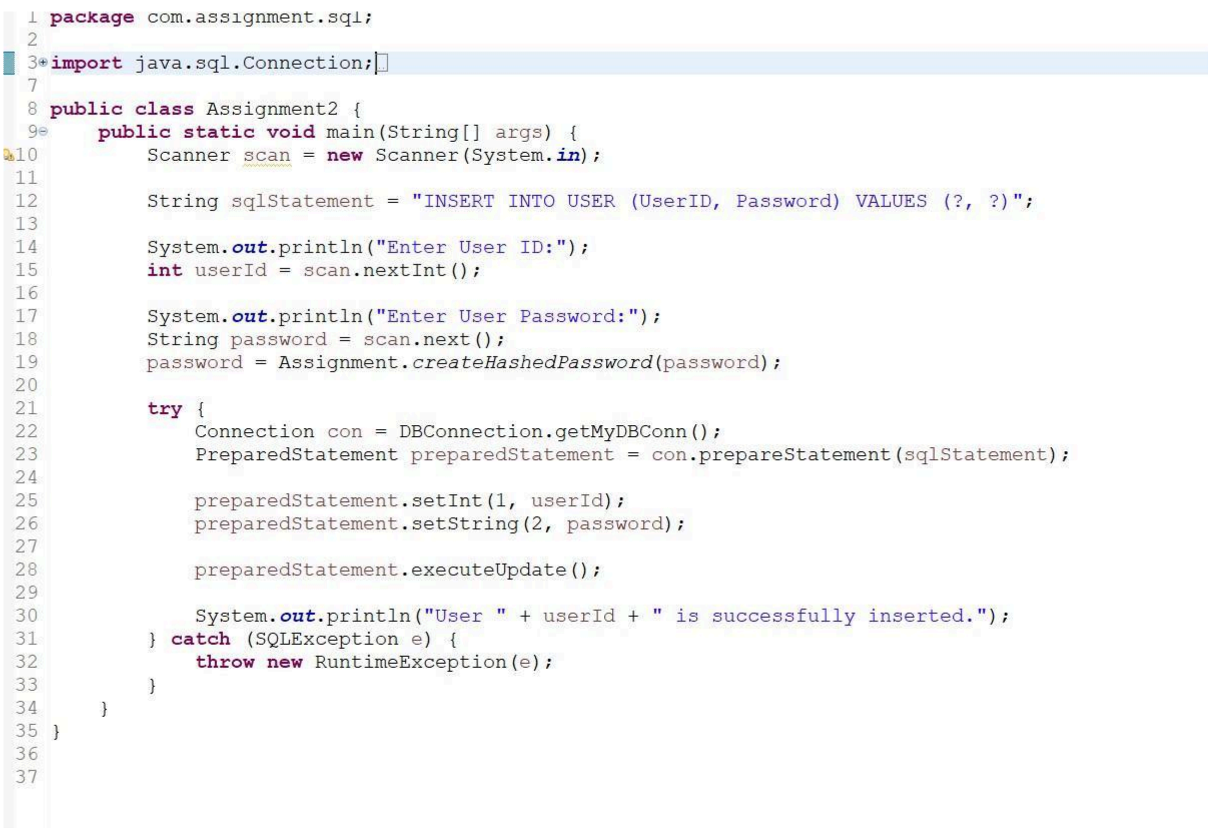
Output:



```
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
```

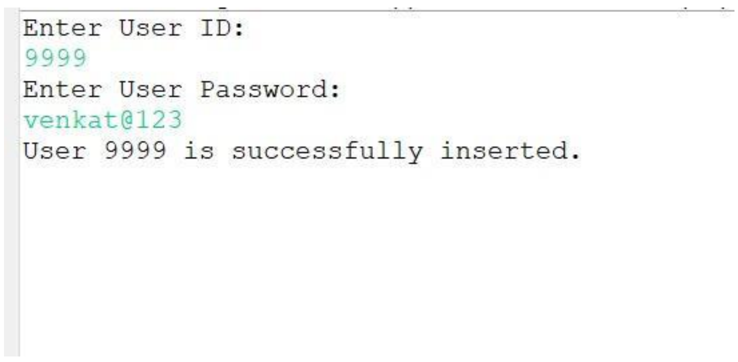
Task 3: PreparedStatement

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

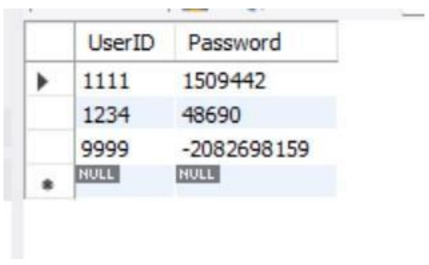


```java
package com.assignment.sql;

import java.sql.Connection;

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:



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



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