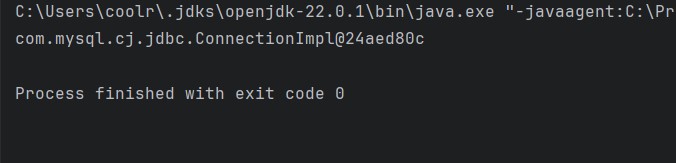
Day 13 Assignment

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

|  |
| --- |
| import java.sql.\*; |
| import java.sql.DriverManager; |
| import java.sql.SQLException; |
| /\* |
| Write a Java program that connects to a SQLite database and |
| prints out the connection object to confirm successful connection. |
| \*/ |
| public class Assignment\_1 { |
| private static final String *url* = "jdbc:mysql://localhost:3306/wiprodb"; |
| private static final String *user* = "root"; |
| private static final String *password* = "Sayan$1999"; |
|  |
| private static Connection *con*; |
|  |
| public static Connection createConnection() { |
|  |
| try { |
| Class.*forName*("com.mysql.cj.jdbc.Driver"); |
| *con* = DriverManager.*getConnection*(*url*, *user*, *password*); |
| } catch (SQLException e) { |
| throw new RuntimeException(e); |
| } catch (ClassNotFoundException e) { |
| throw new RuntimeException(e); |
| } |
| return *con*; |
| } |
|  |
| public static void main(String[] args) { |
| System.*out*.println(*createConnection*()); |
| } |
| } |

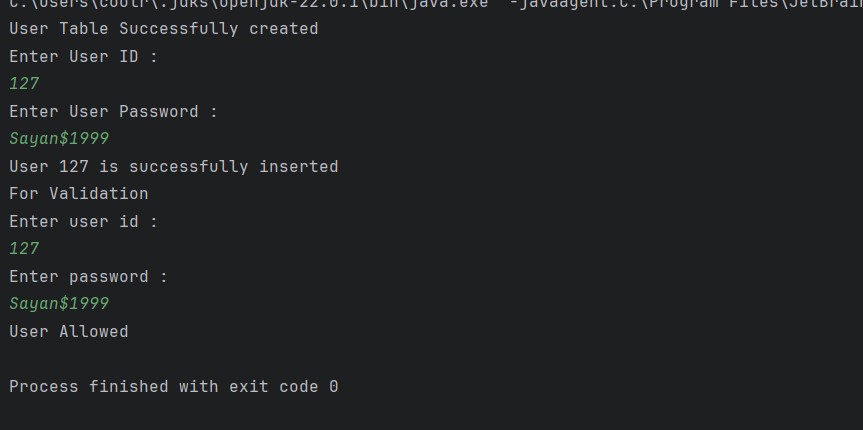
Output



1. 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.

|  |
| --- |
| package m5\_core\_java\_programming.day\_13; |
|  |
| /\* |
| 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.  \*/ |
|  |
| import java.sql.\*; |
| import java.util.Scanner; |
|  |
| public class Assignment\_2 { |
|  |
| public static String createHash(String password) { |
| return Integer.*toString*(password.hashCode()); |
| } |
|  |
| public static boolean checkValidation(int userid, String password, |
| Connection con) { |
| password = *createHash*(password); |
| try { |
| String sqlStat = "SELECT \* FROM User WHERE UserID = ? AND Password = |
| ?"; |
| PreparedStatement preparedStatement = con.prepareStatement(sqlStat); |
| preparedStatement.setInt(1, userid); |
| preparedStatement.setString(2, password); |
| ResultSet resultSet = preparedStatement.executeQuery(); |
| return resultSet.next(); |
| } catch (SQLException e) { |
| throw new RuntimeException(e); |
| } |
| } |
|  |
| public static void main(String[] args) { |
| Scanner scan = new Scanner(System.*in*); |
| try { |
| Connection con = Assignment\_1.*createConnection*(); |
| String sqlStat = "CREATE TABLE User (" + |
| " UserID INT PRIMARY KEY," + |
| " Password VARCHAR(50)" + |
| ");"; |
|  |
| Statement statement = con.createStatement(); |
| statement.executeUpdate(sqlStat); |
| System.*out*.println("User Table Successfully created"); |
| sqlStat = "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 = *createHash*(password); |
| PreparedStatement preparedStatement = con.prepareStatement(sqlStat); |
| preparedStatement.setInt(1, userid); |
| preparedStatement.setString(2, password); |
| preparedStatement.executeUpdate(); |
| System.*out*.println("User " + userid + " is successfully inserted"); |
| System.*out*.println("For Validation "); |
| System.*out*.println("Enter user id : "); |
| userid = scan.nextInt();  System.*out*.println("Enter password : "); |
| password = scan.next(); |
|  |
| if (*checkValidation*(userid, password, con)) { |
| System.*out*.println("User Allowed"); |
| } else { |
| System.*out*.println("User Not Allowed"); |
| } |
|  |
| con.close(); |
| } catch (Exception e) { |
| System.*out*.println(e); |
| } |
| } |
| } |

Output



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

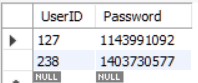
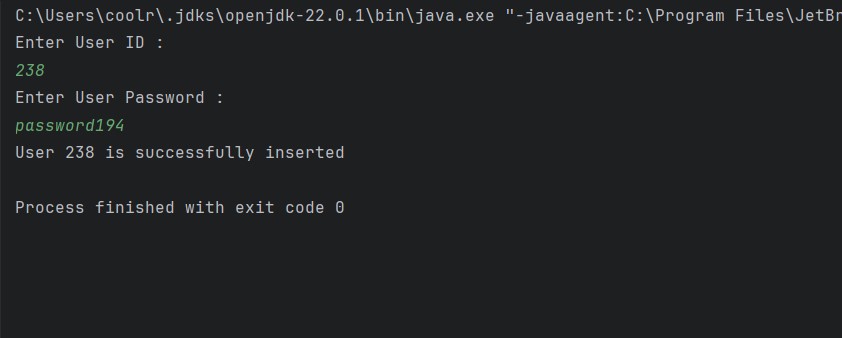
|  |
| --- |
| package m5\_core\_java\_programming.day\_13; |
|  |
| /\* |
| Modify the SELECT query program to use PreparedStatement to parameterize the |
| query and |
| prevent SQL injection. |
| \*/ |
|  |
| import java.sql.Connection; import java.sql.PreparedStatement; |
| import java.sql.SQLException; |
| import java.util.Scanner; |
|  |
| public class Assignment\_3 { |
| public static void main(String[] args) { |
| Scanner scan = new Scanner(System.*in*); |
| String sqlStat = "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\_2.*createHash*(password); |
| try { |
| Connection con = Assignment\_1.*createConnection*(); |
| PreparedStatement preparedStatement = con.prepareStatement(sqlStat); |
| 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

Tools

Used

:



IntelliJ IDE java version "1.8.0\_411"

Java(TM) SE Runtime Environment (build 1.8.0\_411-b09)

Java HotSpot(TM) Client VM (build 25.411-b09, mixed mode, sharing)

MySQL Workbench 8.0 CE