Week 10 : Programming Assignment 5

Due on 2025-04-03, 23:59 IST

Insert a Record into a Table Using JDBC

Once a table is created in a database, we can insert records into it using SQL `INSERT` statements.

In this task, your goal is to insert a single student record into the existing `students` table.

You are provided with the following variables:

- `roll` – an integer roll number
- `name` – a string representing the student's name

Your task is to write one line of code that inserts this data into the table using a `PreparedStatement`.

This is a common and safe way to insert data, as it avoids issues like SQL injection and improper formatting.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	101 Rahul	inserted	inserted\n	Passed

The due date for submitting this assignment has passed.
1 out of 1 tests passed.
You scored 100.0/100.

Assignment submitted on 2025-03-26, 22:36 IST

Your last recorded submission was :

```java
import java.sql.*; // Required for JDBC classes
import java.util.*; // Required for Scanner input

public class W10_P5 {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);

            // Read input values
            int roll = sc.nextInt();
            sc.nextLine(); // consume newline
            String name = sc.nextLine();

            // Set SQLite temp directory to avoid native driver errors in restricted environments
            System.setProperty("org.sqlite.tmpdir", "/tempfs");

            // Connect to SQLite database
            Connection conn = DriverManager.getConnection("jdbc:sqlite:/tempfs/db");

            // Ensure the 'students' table exists (create it if it doesn't)
            Statement stmt = conn.createStatement();
            String createTableQuery = "CREATE TABLE IF NOT EXISTS students (roll INTEGER, name VARCHAR(30))";
            stmt.executeUpdate(createTableQuery);

            // SQL insert statement with placeholders
            String sql = "INSERT INTO students VALUES(?, ?)";

            // Prepare the SQL statement
            PreparedStatement pstmt = conn.prepareStatement(sql);

            // Set values to the placeholders
            pstmt.setInt(1, roll);      // First '?' is replaced with roll number
            pstmt.setString(2, name);   // Second '?' is replaced with name
pstmt.executeUpdate();
// If insert is successful, print a confirmation message
            System.out.println("inserted");

            // Close resources after use
            pstmt.close();
            conn.close();
            sc.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Sample solutions (Provided by instructor)

```java
import java.sql.*; // Required for JDBC classes
import java.util.*; // Required for Scanner input

public class W10_P5 {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);

            // Read input values
            int roll = sc.nextInt();
            sc.nextLine(); // consume newline
            String name = sc.nextLine();

            // Set SQLite temp directory to avoid native driver errors in restricted environments
            System.setProperty("org.sqlite.tmpdir", "/tempfs");

            // Connect to SQLite database
            Connection conn = DriverManager.getConnection("jdbc:sqlite:/tempfs/db");

            // Ensure the 'students' table exists (create it if it doesn't)
            Statement stmt = conn.createStatement();
            String createTableQuery = "CREATE TABLE IF NOT EXISTS students (roll INTEGER, name VARCHAR(30))";
            stmt.executeUpdate(createTableQuery);

            // SQL insert statement with placeholders
            String sql = "INSERT INTO students VALUES(?, ?)";

            // Prepare the SQL statement
            PreparedStatement pstmt = conn.prepareStatement(sql);

            // Set values to the placeholders
            pstmt.setInt(1, roll);      // First '?' is replaced with roll number
            pstmt.setString(2, name);   // Second '?' is replaced with name
// This line executes the SQL INSERT command.
// The executeUpdate() method is used to run INSERT, UPDATE, or DELETE statements.
// It returns the number of rows affected (not needed here).
// If it completes without throwing an exception, the data has been successfully inserted.
pstmt.executeUpdate();
// If insert is successful, print a confirmation message
            System.out.println("inserted");

            // Close resources after use
            pstmt.close();
            conn.close();
            sc.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```