Sanjivani Rural Education Society's College of Engineering,
Kopargaon Department of Electronics and Computer Engineering
TITLE: Experiment Write-up (EW)

Subject: Database Management System Laboratory

EXPERIMENT NO : 06

TITLE: Write a program in a high-level host language such as C/Java and demonstrate to interact with database**.**

LEARNING OBJECTIVES:

1. To understand object-oriented programming concepts in Java.
2. To make students familiar with Java programs for problem-solving.
3. To learn Java programming concepts such as interfaces, exception handling, and packages.

THEORY:

1. Introduction to JDBC:

JDBC (Java Database Connectivity) is a Java API that allows Java applications to interact with relational databases.
It provides a standard interface for accessing databases.

2. JDBC Architecture:

Application Layer: The Java application interacting with the database.
JDBC API: Classes and interfaces provided by JDBC to perform database operations.
Driver Manager: Manages registered database drivers.
JDBC Driver: Implements JDBC interfaces to communicate with specific databases.

3. Key Concepts:

Database Connection: Connection interface represents a connection to the database.
Statement Execution: Statement, PreparedStatement, and CallableStatement are used to execute SQL queries.
Result Sets: ResultSet holds data retrieved from the database.
Transaction Management: JDBC allows commit() and rollback() for transaction handling.
Exception Handling: Use SQLException to handle errors gracefully.
Metadata Access: Retrieve database and table information using DatabaseMetaData and ResultSetMetaData.

JDBC Implementation Steps:

1. **Import Necessary Libraries:**
   ```
   import java.sql.Connection;
   import java.sql.DriverManager;
   import java.sql.SQLException;
   import java.sql.Statement;
   import java.sql.ResultSet;
   ```

2. **Load Driver Class:**
   ```
   Class.forName("com.mysql.cj.jdbc.Driver");
   ```

3. **Establish Database Connection:**
   ```
   Connection conn = DriverManager.getConnection(
      "jdbc:mysql://localhost:3306/your_database", "username", "password");
   ```

4. **Create Statement Object:**
   ```
   Statement stmt = conn.createStatement();
   ```

5. **Execute SQL Query:**
   ```
   String sql = "INSERT INTO backlogs(rn, name, nobacklogs) VALUES (77, 'Rohit', 5)";
   stmt.execute(sql);
   ```

6. **Process Results (if SELECT query):**
   ```
   ResultSet rs = stmt.executeQuery("SELECT * FROM backlogs");
   while(rs.next()){
      System.out.println(rs.getInt("rn") + "\t" + rs.getString("name") + "\t" +
   rs.getInt("nobacklogs"));
   }
   ```

7. **Handle Exceptions:**
   ```
   try {
      // JDBC operations
   } catch(SQLException e) {
      e.printStackTrace();
   }
   ```

8. **Close Resources:**
   ```
   stmt.close();
   conn.close();
   ```

PROGRAM FOR IMPLEMENTATION:-

```
package student_records;

import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class DatabaseInsert {
   public static void main(String[] args) {
      try {
         // Load MySQL JDBC driver
         Class.forName("com.mysql.cj.jdbc.Driver");

         // Establish connection
         Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/harshal", "root", "Rohit@123");

         // Create statement
         Statement stmt = conn.createStatement();

         // Static SQL query to insert data
         String sql = "INSERT INTO backlogs(rn, name, nobacklogs) VALUES (77, 'Rohit', 5)";
         stmt.execute(sql);

         System.out.println("Inserted Successfully");

         // Close resources
         stmt.close();
         conn.close();

      } catch (ClassNotFoundException e) {
         System.out.println("Driver not found: " + e);
      } catch (SQLException e) {
         System.out.println("SQL Error: " + e);
      }
   }
}
```

CONCLUSION:

We successfully implemented JDBC using a static SQL query in Java.
Learned how to establish a connection, execute SQL queries, and handle exceptions.
Gained understanding of JDBC architecture and interfaces like Connection and Statement.
Emphasized the importance of closing resources to avoid memory leaks.
This hands-on experiment builds the foundation for developing real-world Java database applications.