Experiment No.9
Demonstrate Database connectivity
Date of Performance:
Date of Submission:

Aim :- Write a java program to connect Java application with the MySQL database

Objective :- To learn database connectivity

Theory:

Database used : MySql

1. Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver. 2. Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/loan management where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, can also use IP address, 3306 is the port number and loan management is the database name. 3. Username: The default username for the mysql database is Hiren.
4. Password: It is the password given by the user at the time of installing the mysql database. Password used is " ".

To connect a Java application with the MySQL database, follow the following steps.

● First create a database and then create a table in the mysql database. ● To connect java application with the mysql database, mysqlconnector.jar file is required to be loaded.

● download the jar file mysql-connector.jar
● add the jar file to the same folder as the java program.
● Compile and run the java program to retrieve data from the database.

Implementation:

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

public class MySQLConnectionExample {
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/mydatabase";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "root";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            // Register MySQL JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Open a connection
            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);

            // Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();
```

```
26          System.out.println("Creating statement...");
27          stmt = conn.createStatement();
28          String sql = "SELECT id, name, age FROM employees";
29          rs = stmt.executeQuery(sql);
30
31          // Process the result set
32          while (rs.next()) {
33              // Retrieve by column name
34              int id = rs.getInt("id");
35              String name = rs.getString("name");
36              int age = rs.getInt("age");
37
38              // Display values
39              System.out.print("ID: " + id);
40              System.out.print(", Name: " + name);
41              System.out.println(", Age: " + age);
42          }
43      } catch (SQLException | ClassNotFoundException e) {
44          e.printStackTrace();
45      } finally {
46          // Close resources
47          try {
48              if (rs != null) rs.close();
49              if (stmt != null) stmt.close();
50              if (conn != null) conn.close();
51          } catch (SQLException e) {
```

JDBCDemo.java

```
 7  public class MySQLConnectionExample {
12      public static void main(String[] args) {
17          try {
32              while (rs.next()) {
34                  int id = rs.getInt("id");
35                  String name = rs.getString("name");
36                  int age = rs.getInt("age");
37
38                  // Display values
39                  System.out.print("ID: " + id);
40                  System.out.print(", Name: " + name);
41                  System.out.println(", Age: " + age);
42              }
43          } catch (SQLException | ClassNotFoundException e) {
44              e.printStackTrace();
45          } finally {
46              // Close resources
47              try {
48                  if (rs != null) rs.close();
49                  if (stmt != null) stmt.close();
50                  if (conn != null) conn.close();
51              } catch (SQLException e) {
52                  e.printStackTrace();
53              }
54          }
55      }
56  }
57
```

Conclusion: Data has been retrieved successfully from a table by establishing database connectivity of java program with mysql database.

1. Explain steps to connect a java application with the MySQL database

Ans. To connect a Java application with a MySQL database, you can follow these steps:

Download MySQL Connector/J:

- First, you need to download the MySQL Connector/J driver, which allows Java applications to connect to a MySQL database. You can download it from the official MySQL website or include it as a dependency in your project using a build tool like Maven or Gradle.

Include the Connector/J Driver in your Java Project:

- Once you have downloaded the MySQL Connector/J JAR file, include it in your Java project's classpath. If you are using a build tool like Maven or Gradle, you can specify the dependency in your project configuration file (pom.xml for Maven or build.gradle for Gradle).

2. **Establish a Connection:**

   Use the **DriverManager.getConnection()** method to establish a connection to your MySQL database. You need to provide the JDBC URL, username, and password as parameters to this method. For example:

   String url = "jdbc:mysql://localhost:3306/mydatabase";

   String username = "username";

   String password = "password";

   Connection connection = DriverManager.getConnection(url, username, password);

1. **Create a Statement:** Once the connection is established, create a **Statement** object using the **Connection.createStatement()** method. This statement will be used to execute SQL queries against the database.
2. **Execute SQL Queries:** Use the **Statement.executeQuery()** method to execute SELECT queries that retrieve data from the database, or use the **Statement.executeUpdate()** method to execute INSERT, UPDATE, DELETE, or DDL (Data Definition Language) queries that modify the database.

   Statement statement = connection.createStatement();

   ResultSet resultSet = statement.executeQuery("SELECT * FROM mytable");

3. **Close the Connection and Resources:**

   resultSet.close();

   statement.close();

   connection.close();

   4. Handle Exceptions:

   Handle any potential exceptions that may occur during database connectivity and query execution. This includes **SQLExceptions** that may be thrown when interacting with the database.