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 APPLICATION:-

```
import java.sql.*;
 public class JDBCDemo{
   public static void main(String args[]){
     try{
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection(
             "jdbc:mysql://localhost:3306/Hotel_Management","root","Mangesh@123");
//here sonoo is database name, root is username and password
        Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery("select * from customer");
```

```
        while(rs.next())
            System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
        con.close();
      }catch(Exception e){ System.out.println(e);}
    }
}
```

```
C:\Users\ts395\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.1\lib\idea_rt.jar=60829:C:\Program Files\JetBra
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via
1  Piyush  Pradip
2  Sara  samir
3  Priya  Sandesh

Process finished with exit code 0
```

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**

Sure, here are the steps to connect a Java application with a MySQL database:

1. Import JDBC Library:
   - Ensure that you have the MySQL JDBC driver JAR file included in your Java project's build path. You can download the MySQL Connector/J driver from the MySQL website or use a dependency management tool like Maven or Gradle.

2. **Load the JDBC Driver:
   - Use the `Class.forName()` method to dynamically load the MySQL JDBC driver class. For example:

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

3. Establish Connection:
   - Use the `DriverManager.getConnection()` method to establish a connection to the MySQL database. Provide the database URL, username, and password as parameters. For example:

```
String url = "jdbc:mysql://localhost:3306/mydatabase";
String username = "root";
String password = "password";
Connection connection = DriverManager.getConnection(url, username, password);
```

4. Create Statement:
   - Create a Statement or PreparedStatement object to execute SQL queries against the database. For example:

```
Statement statement = connection.createStatement();
```

5. Execute SQL Queries:
- Use the `executeQuery()` method to execute SELECT queries or `executeUpdate()` method to execute INSERT, UPDATE, DELETE queries. For example:

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

6. Process Results:
- Iterate through the ResultSet object to retrieve and process the query results. For example:

```
while (resultSet.next()) {
    String name = resultSet.getString("name");
    int age = resultSet.getInt("age");
    // Process data...
}
```

7. Close Resources:
- Close the ResultSet, Statement, and Connection objects when done to release database resources and prevent memory leaks. For example:

```
resultSet.close();
statement.close();
connection.close();
```