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.

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

In conclusion, connecting a Java application with a MySQL database involves several steps:

1. Download and Install MySQL Connector/J:
   - Download the MySQL Connector/J driver from the official MySQL website.
   - Install the driver by adding the JAR file to your Java project's classpath.

2. Create a MySQL Database:
   - Use MySQL Workbench or the MySQL command-line interface to create a new database if one doesn't already exist.
   - Define tables and schema as needed for your application.

3. Establish Connection in Java:
   - Import the necessary classes from the `java.sql` package, including `Connection`, `DriverManager`, and `SQLException`.
   - Use the `DriverManager.getConnection()` method to establish a connection to the MySQL database, passing the connection URL, username, and password as parameters.

4. Handle Exceptions:
   - Wrap the connection code in a try-catch block to handle potential exceptions, such as `ClassNotFoundException` or `SQLException`.

5. Execute SQL Queries:

- Once the connection is established, create `Statement` or `PreparedStatement` objects to execute SQL queries against the database.
- Use methods like `executeQuery()` for SELECT statements or `executeUpdate()` for INSERT, UPDATE, DELETE statements.

6. Process Results:
   - If executing a SELECT query, use the `ResultSet` object to retrieve and process the query results.

7. Close Connection:
   - Always close the connection, statements, and result sets after use to release database resources and prevent memory leaks.
   - Use the `close()` method on the connection, statement, and result set objects within a finally block or try-with-resources block.

By following these steps, you can successfully connect a Java application with a MySQL database, allowing seamless interaction between the application and the database management system. This integration enables data storage, retrieval, and manipulation within your Java application, facilitating robust and efficient data-driven applications.