Module 2 – Java – RDBMS & Database Programming with JDBC Introduction to JDBC

• Theory:

1. **What is JDBC (Java Database Connectivity)?**

The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a relational database.

JDBC helps you to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query
4. **Importance of JDBC in Java Programming**

JDBC includes four components:

1. **The JDBC API** —  The JDBC™ API provides programmatic access to relational data from the Java™ programming language. Using the JDBC API, applications can execute SQL statements, retrieve results, and propagate changes back to an underlying data source. The JDBC API can also interact with multiple data sources in a distributed, heterogeneous environment.

The JDBC API is part of the Java platform, which includes the *Java™ Standard Edition* (Java™ SE ) and the *Java™ Enterprise Edition* (Java™ EE). The JDBC 4.0 API is divided into two packages: java.sql and javax.sql. Both packages are included in the Java SE and Java EE platforms.

1. **JDBC Driver Manager** —  The JDBC DriverManager class defines objects which can connect Java applications to a JDBC driver. DriverManager has traditionally been the backbone of the JDBC architecture. It is quite small and simple.

The Standard Extension packages javax.naming and javax.sql let you use a DataSource object registered with a *Java Naming and Directory Interface*™ (JNDI) naming service to establish a connection with a data source. You can use either connecting mechanism, but using a DataSource object is recommended whenever possible.

1. **JDBC Test Suite** —  The JDBC driver test suite helps you to determine that JDBC drivers will run your program. These tests are not comprehensive or exhaustive, but they do exercise many of the important features in the JDBC API.
2. **JDBC-ODBC Bridge** —  The Java Software bridge provides JDBC access via ODBC drivers. Note that you need to load ODBC binary code onto each client machine that uses this driver. As a result, the ODBC driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in a three-tier architecture.

This Trail uses the first two of these four JDBC components to connect to a database and then build a java program that uses SQL commands to communicate with a test relational database. The last two components are used in specialized environments to test web applications, or to communicate with ODBC-aware DBMSs.

**3 JDBC Architecture**: **Driver Manager, Driver, Connection, Statement, and ResultSet**

* DriverManager:

Manages the set of JDBC drivers available for an application to use, ensuring the correct driver is used for a specific database.

* Driver:

An interface that controls communication with a specific database server, including establishing connections and executing queries.

* Connection:

Represents a connection to a database, providing methods to create statements and manage the connection.

* Statement:

An interface that represents a SQL statement, allowing you to execute queries and retrieve results.

* ResultSet:

An interface that represents the result set of a query, allowing you to access the data row by row.

• Lab Exercise:

o Write a simple Java program to connect to a MySQL database using JDBC.

o Demonstrate the process of loading a JDBC driver and establishing a connection.

2. JDBC Driver Types

• Theory:

o Overview of JDBC Driver Types:

♣ Type 1: JDBC-ODBC Bridge Driver

♣ Type 2: Native-API Driver

♣ Type 3: Network Protocol Driver

♣ Type 4: Thin Driver o Comparison and Usage of Each Driver Type

• Lab Exercise: o Identify which driver your Java program uses to connect to MySQL. o Research and explain the best JDBC driver for your database and Java environment. 3. Steps for Creating JDBC Connections • Theory: o Step-by-Step Process to Establish a JDBC Connection: 1. Import the JDBC packages 2. Register the JDBC driver 3. Open a connection to the database 4. Create a statement 5. Execute SQL queries 6. Process the result set 7. Close the connection • Lab Exercise: o Write a Java program to establish a connection to a database and print a confirmation message upon successful connection.