




# Unit X

## Database Connectivity

Tutor: Pema Galey

# Did you study ….?

- GUI
- Event Handling
  - Event Listener Interfaces
  - Adaptor classes

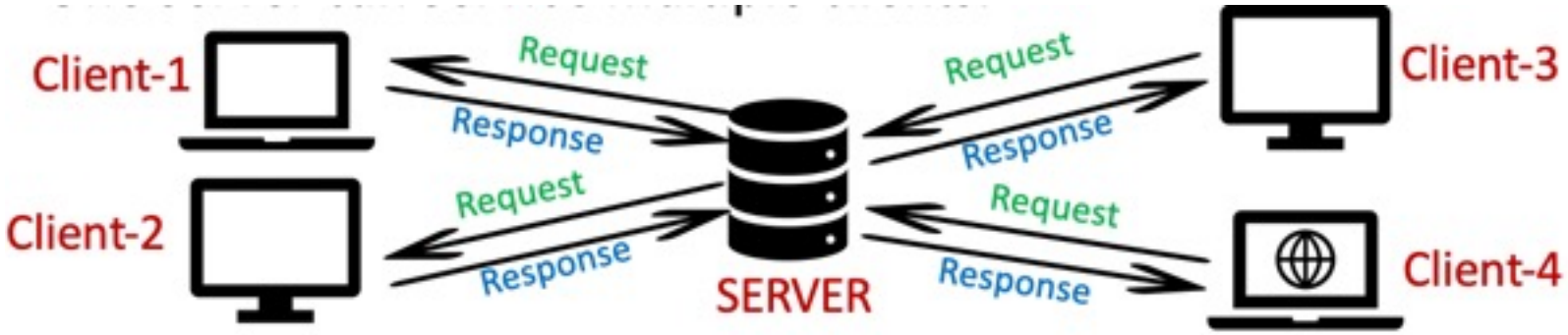
# Learning Outcomes

In this session, you will learn about:

- Introduction to Networking
- Client and Server
- Socket
- IP Addresses and Ports
- Basic Database using SQL
- JDBC Architectures

# Networking : Client/Server Architecture

- The Client/Server model of application development is designed to separate the presentation of data from its internal processing and storage.
- The client requests for services and the server services these requests. The requests are transferred from the client to the server over the network.
- The processing that is done by the server is hidden from the client. One server can service multiple clients.



# Networking : Client/Server Architecture

- Network protocols are a set of rules and conventions followed by systems that communicate over a network.
- Some examples of network protocols are TCP/IP, UDP, Apple Talk, and NetBEUI.
- Java provides a rich library of network-enabled classes that allow applications to readily access network resources.

# Sockets

- Sockets are used to handle the communication links between applications over the network.
- TCP provides a reliable, point-to-point communication channel for Client/Server applications to communicate with each other.
- The advantage of the socket model using TCP is that the server can communicate with any kind of client.
- The client is not restricted to the UNIX, Windows, DOS, or Macintosh platforms.
- Java provides the following socket classes:
  - `Socket`: It is the basic class and supports the TCP protocol. TCP is a stream network connection protocol.
  - `ServerSocket`: It is a class used by the Internet server programs for listening to client requests.

# IP Address and Ports

- An Internet server can be thought of as a set of socket classes that provide additional capabilities that are generally called services.
- Each service is associated with a port, which is a numeric address through which service requests, such as a request for a Web page, are processed.
- The TCP protocol requires two data items, the IP address and the port number.
- IP provides every network device with a logical address called an IP address.
- IPv4: The IP addresses provided by the Internet protocol is a 32-bit number, represented as a series of four 8-bit numbers, which range in value from 0 to 255.
- IPv6: IPv6 is 128 Bit IP Address.

- Identifying the Layers in JDBC Architecture
- How to connect using JDBC/ODBC Driver?

- ***But lets discuss about SQL!***

# Software

- Install **Xampp/Mamp** software for Database Connectivity

# DATABASE

## *Practical Concept on Database using SQL Query*

# Database

- What is **Database**?
  - A comprehensive collection of related data organized for convenient access, generally in a computer.
  - A structured set of data held in a computer, especially one that is accessible in various ways.
  - A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added.

# SQL Query

- **SQL** (**Structured Query Language**) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).
- **SQL** is used to communicate with a database which is the standard language for relational database management systems. SQL is used to *query, insert, update and modify data*.
- SQL is nonprocedural language that provides database access.

# Basic SQL queries

- **Create Database**

```
CREATE DATABASE databasename;
```

```
CREATE DATABASE studentdb;
```

- **Drop Database**

```
DROP DATABASE databasename;
```

```
DROP DATABASE studentdb;
```

- **Create Table**

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

```
CREATE TABLE biotdata (
    StudentID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

# Basic SQL queries

- **Drop Table**

```
DROP TABLE table_name;
```

```
DROP TABLE biodata;
```

- **INSERT Query**

```
INSERT INTO table_name (column1, column2, column3,...columnN) VALUES (value1, value2, value3,...valueN);
```

```
INSERT INTO biodata (StudentID, LastName, FirstName, Address, City) VALUES (123, 'Dorji', 'Tashi', 'CST', 'Phuentsholing');
```

# Example for Table

- Table: biodata

| StudentID | LastName | FirstName | Address | City |
|---|---|---|---|---|
| 20001 | Dorji | Tashi | CNR | Punakha |
| 20002 | Lhamo | Sonam | CST | Phuentsholing |
| 20003 | Penjor | Sonam | CST | Phuentsholing |
| 20004 | Dorji | Tshering | PCE | Paro |
| 20005 | Phuntsho | Sonam | Kanglung | Tashigang |
| 20006 | Tharchen | Jigme | Gyelphozhing | Mongar |
| 20007 | Dorji | Tashi | CST | Phuentsholing |

# Basic SQL queries

- **SELECT Query (Retriving data from table)**

```
SELECT * FROM table_name;
SELECT * FROM biodata;
SELECT StudentID, FirstName,LastName FROM biodata;

SELECT * FROM table_name WHERE  conditions
```

- **AND & OR Operations**

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] AND [condition2]...AND [conditionN];
```

# Basic SQL queries

## ➢UPDATE Query

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

## ➢DELETE Query

```
DELETE FROM table_name
WHERE [condition];
```

# Basic SQL queries

## Other Queries

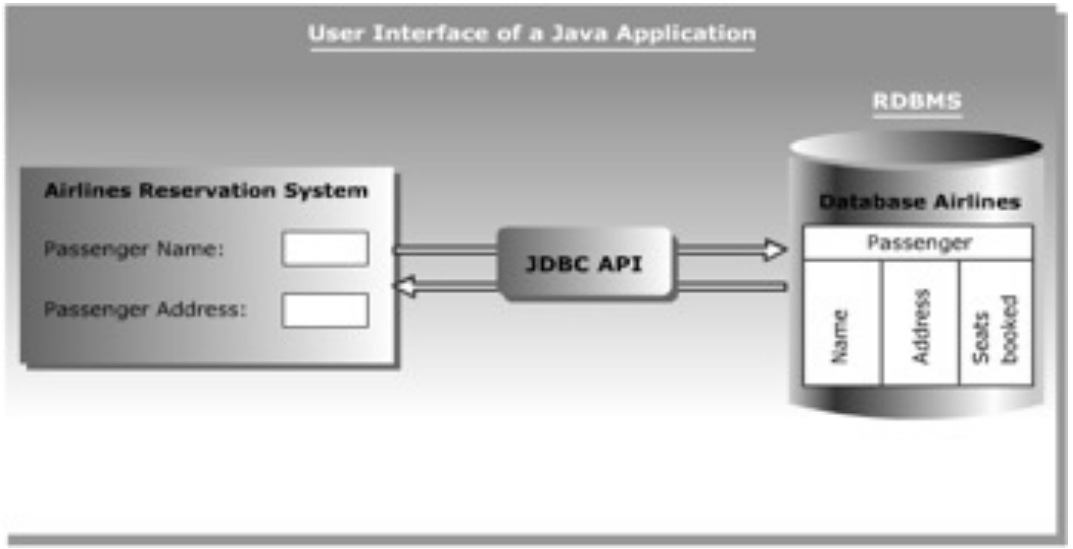
- ✓**LIKE**
- ✓**TOP**
- ✓**ORDER BY**
- ✓**GROUP BY**

# Home Work

- Install Xampp (You can use GUI interface) and make runnable locally.
- Create Database named StudentDB
- Create Table named BioData (StudentID, LastName, FirstName, Address, City)
- Insert data a minimum of 10 (keep common first name with common city)
- Write a SQL query:
  - Display all the data
  - Display only those who are from PCE
  - Display from city *Phuentsholing* whose first name with *Dorji*
  - Display those who are from either CST or CNR
  - Delete the data if any student with last name Tashi

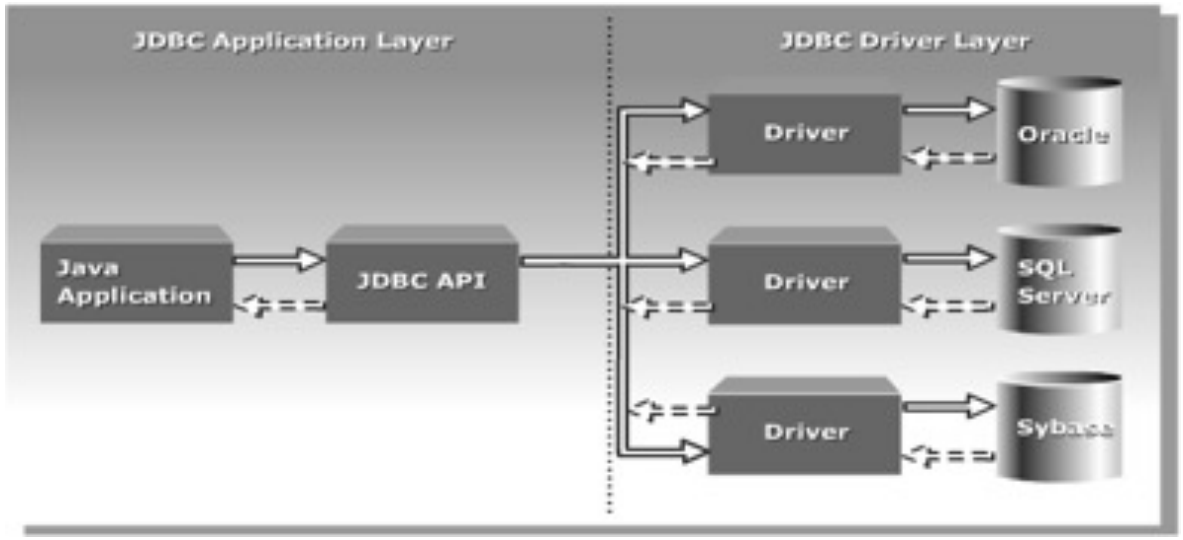
# Identifying the Layers in JDBC Architecture

- Sun Microsystems has included JDBC API as a part of J2SDK to develop Java applications that can communicate with databases.
- The following figure shows the airline reservation system developed in Java interacting with the airlines database using the JDBC API.



# JDBC Architecture

- JDBC architecture provides the mechanism to translate Java statements into SQL statements.
- It can be classified into two layers:
  - JDBC application layer
  - JDBC driver layer
  - The following figure displays the JDBC architecture.

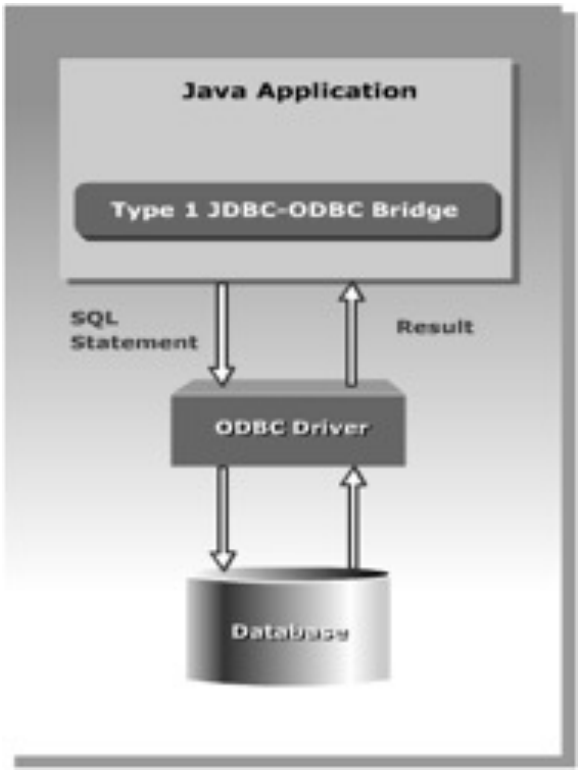


# Types of JDBC Drivers

- JDBC drivers are used to convert queries into a form that a particular database can interpret.
- They retrieve the result of SQL statements and convert the result into equivalent JDBC API class objects.
- They are of four types:
  - JDBC-ODBC Bridge driver
  - Native-API Partly-Java driver
  - JDBC-Net Pure-Java driver
  - Native Protocol Pure-Java driver

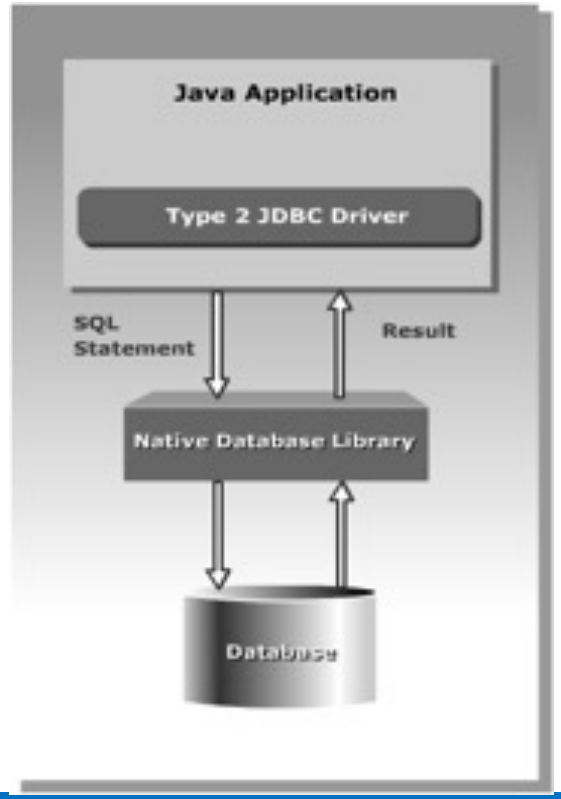
# The JDBC-ODBC Bridge driver

- The JDBC-ODBC Bridge driver is called the Type 1 driver.
- The following figure displays a JDBC-ODBC Bridge driver.



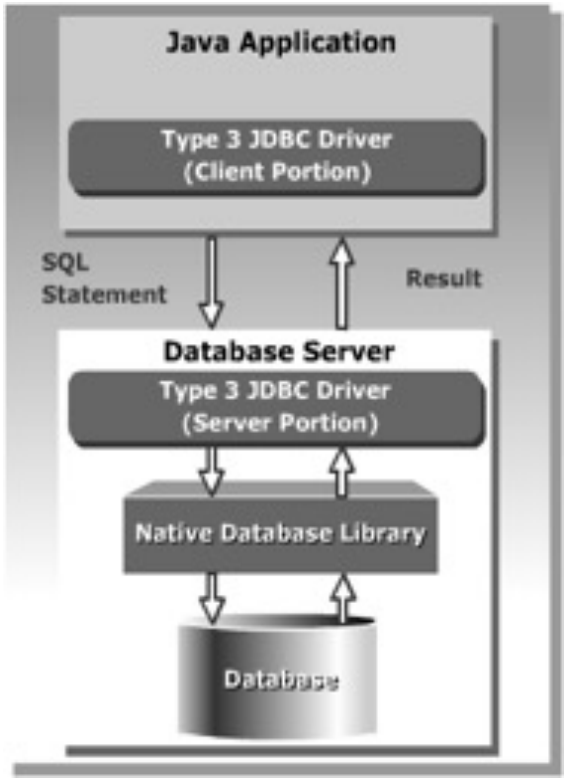
# The Native-API Partly-Java driver

- The Native-API Partly-Java driver is called the Type 2 driver.
- The following figure displays a Native-API Partly-Java driver.



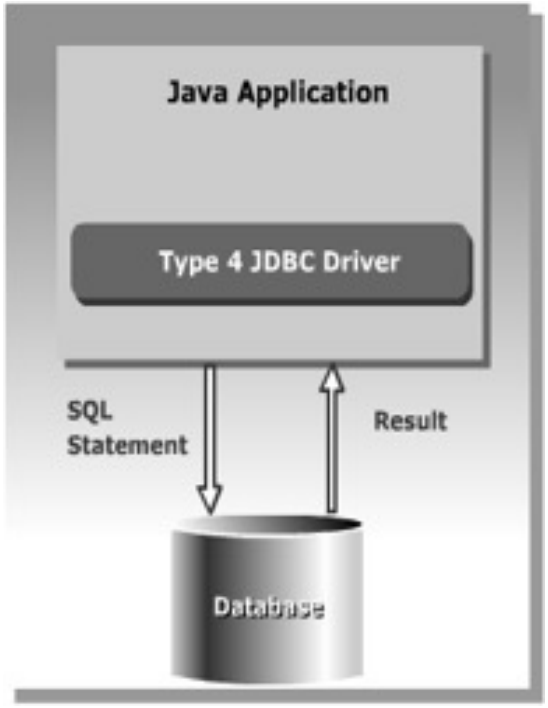
# The JDBC-Net Pure-Java driver

- The JDBC-Net Pure-Java driver is called the Type 3 driver.
- The following figure displays a JDBC-Net Pure-Java driver.



# The Native-Protocol Pure-Java driver

- The Native-Protocol Pure-Java driver is called the Type 4 driver.
- The following figure displays a Native-Protocol Pure-Java driver.



# JDBC API

# JDBC API Classes and Interfaces

- The JDBC API classes and interfaces are available in the `java.sql` and the `javax.sql` packages.
- The commonly used classes and interfaces in the JDBC API are:
  - `DriverManager` class: Loads the driver for a database.
  - `Driver` interface: Represents a database driver. All JDBC driver classes must implement the Driver interface.
  - `Connection` interface: Enables you to establish a connection between a Java application and a database.
  - `Statement` interface: Enables you to execute SQL statements.
  - `ResultSet` interface: Represents the information retrieved from a database.
  - `SQLException` class: Provides information about the exceptions that occur while interacting with databases.

# JDBC API Classes and Interfaces

- To query a database and display the result using Java applications, you need to perform the following steps:
  1. Load a driver.
  2. Connect to a database.
  3. Create and execute JDBC statements.
  4. Handle SQL exceptions.

# 1. Loading a Driver

Driver can be loaded:

1. Programmatically:
   - Using the `forName`() method of the `java.lang.Class` class
   - Using the `registerDriver`()method of the `DriverManager` class
2. Manually:
   - By setting system property

- Using the `forName`() Method:
  - The `forName`() method loads the JDBC driver and registers the driver with the `DriverManager`.
  - The following method call is used for the `forName`() method:

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

# 1. Loading a Driver

- Using the `registerDriver`() Method:
  - You can create an instance of the JDBC driver and pass the reference to the `registerDriver`() method.
  - The following statement is used to create an instance of the Driver class:

```
Driver d = new com.microsoft.sqlserver.jdbc.SQLServerDriver();
```

  - The `registerDriver`() method is called to register the Driver object with the `DriverManager` class.
  - The following method call is used to register the Type 4 driver:

```
DriverManager.registerDriver(d);
```

# 1. Loading a Driver

- Setting System Property:
  - Add the driver name to the `jdbc.drivers` system property to load a JDBC driver.
  - Use the –D command line option to set the system property on the command line.
  - The command to set the system property is:

```
java -Djdbc.drivers=
com.microsoft.sqlserver.jdbc.SQLServerDriver SampleApplication
```

# 2. Connecting to Database

- The `DriverManager` class provides the `getConnection`() method to create a `Connection` object.
- The `getConnection`() method has the following three forms:
  - `public static Connection getConnection(String url)`
  - `public static Connection getConnection(String url, Properties info)`
  - `public static Connection getConnection(String url, String user,String password)`
- The following syntax is used to pass a JDBC URL as a parameter to the `getConnection`():

```
<protocol>:<subprotocol>:<subname>
```

# 3. Creating and Executing JDBC Statements

- The Connection object provides the `createStatement()` method to create a `Statement` object.
- Static SQL statements are used to send requests to a database.
- The `Statement` interface contains the following methods to send static SQL statements to a database:
  - `ResultSet executeQuery(String str)`
  - `int executeUpdate(String str)`
  - `boolean execute(String str)`

# 3. Creating and Executing JDBC Statements

- Various database operations that you can perform using a Java application are:
  - Querying a table
  - Inserting rows in a table
  - Updating rows in a table
  - Deleting rows from a table
  - Creating a table
  - Altering and dropping a table

# 3. Creating and Executing JDBC Statements

- The SELECT statement is executed using the **`executeQuery()`** method and returns the output in the form of a **`ResultSet`** object.

- The following code snippet is used to retrieve data from the *biodata* table:

```
String str = "SELECT * FROM biodata";
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(str);
```

# 3. Creating and Executing JDBC Statements

- The **`executeUpdate`**() method enables you to add rows in a table.
- The following code snippet is used to insert a row in the *biodata* table:
  - ✓
```
String str = "INSERT INTO biodata (StudentID, LastName,
FirstName, Address, City) VALUES (123, 'Dorji', 'Tashi', 'CST',
'Phuentsholing')";
```
  - ✓
```
Statement stmt = con.createStatement();
```
  - ✓
```
int count = stmt.executeUpdate(str);
```

# 3. Creating and Executing JDBC Statements

- The following code snippet is used to modify a row from the authors table:

```
String str = "UPDATE biodata SET address='Khuruthang' where
studentid= '20001'";
Statement stmt = con.createStatement();
int count = stmt.executeUpdate(str);
```

- The following code snippet is used to delete a row from the authors table is:

```
String str = "DELETE FROM biodata WHERE studenid='20003'";
Statement stmt = con.createStatement();
int count = stmt.executeUpdate(str);
```

# 3. Creating and Executing JDBC Statements

- The **`CREATE TABLE`** statement is used to create and define the structure of a table in a database.

- The following code snippet is used to create a table:

```
String str="CREATE TABLE biotdata"
    +"StudentID int,"
    +"LastName varchar(255),"
    +"FirstName varchar(255),"
    +"Address varchar(255),"
    +"City varchar(255))";
Statement stmt=con.createStatement();
stmt.execute(str);
```

# 3. Creating and Executing JDBC Statements

- DDL provides the **ALTER** statement to modify the definition of database object.
- The following code snippet to add a column to the BioData table:

```
String str="ALTER TABLE biodata "+"ADD age INTEGER";
Statement stmt=con.createStatement();
stmt.execute(str);
```

- DDL provides the **DROP TABLE** statement to drop a table from a database.
- The following code snippet is used to drop the BioData table from a database:

```
String str="DROP TABLE biodata";
Statement stmt=con.createStatement();
stmt.execute(str);
```

# Home Work

- Install Xampp (You can use GUI interface) and make runnable locally.
- Create Database named StudentDB
- Create Table named BioData
- Include functionalities for:
    1. Display data (Retreive) → Done
    2. Display data with Condition
    3. Add new data (Insert) → Done
    4. Update existing data
    5. Delete/Remove existing data

# Thank you!