Dr. Ambedkar Institute of Technology, Bengaluru

Department of Computer Science & Engg.

1

UG Programme

Subject Name: Java Programming
Subject Code: 18CS52

PRESENTATION BY

DR. SMITHA SHEKAR B
ASSOCIATE PROFESSOR
DEPT. OF C.S.E
DR. AMBEDKAR INSTITUTE OF TECHNOLOGY
BANGALORE-560056

UNIT - 4

Event Handling

- History of user interface toolkit; Displaying the Frames
- Event Handling Mechanisms
- **The Delegation Event Model(DEM)**
- Sources of events
- Adapter classes
- Inner classes

Dr. Smitha Shekar B

Introducing GUI Programming with Swing

- Introducing Swing

JDBC

- The Concept of JDBC
- JDBC Driver Types
- JDBC Packages
- A Brief Overview of the JDBC process
- Database Connection
- Statement Objects; ResultSet; Transaction Processing

Dr. Smitha Shekar B

Agenda

The Delegation Event Model(DEM)

Dr. Smitha Shekar B

DELEGATION EVENT MODEL (DEM)

Delegation Event Model is a specialized kind of Observer

•Description: "An event is propagated from a 'Source' object to a 'Listener' object by invoking a method on the listener and passing in the instance of the event subclass which defines the event type generated.“

•…..called "delegation" event model because the event source "delegates" the processing of an event to a separate object (the event listener).

Dr. Smitha Shekar B

Participants (based on Java's implementation)

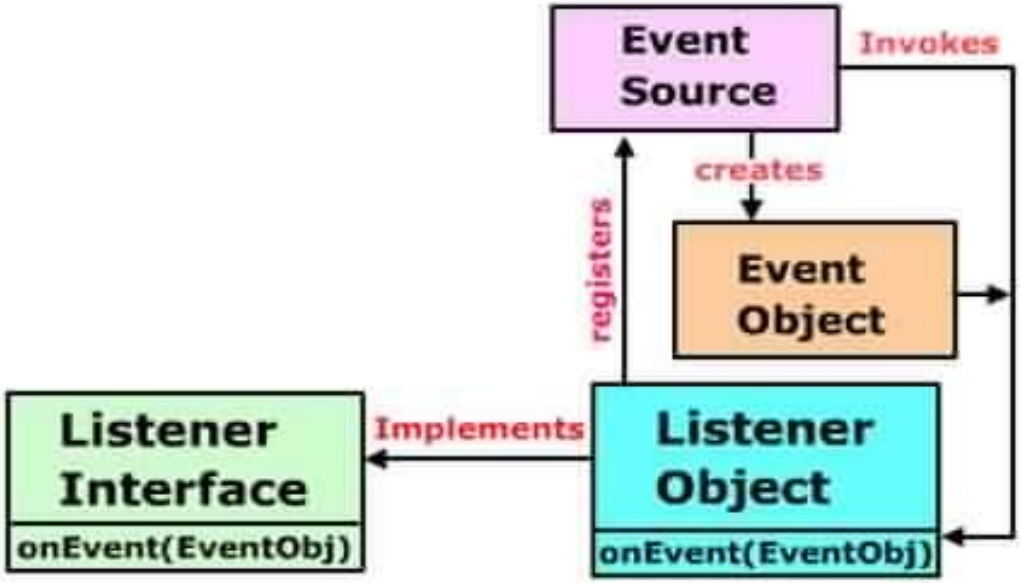
- **event source**
  1. registers/unregisters event listener objects
  2. broadcasts events to listeners (by invoking methods on them)
  3. pushes an "event object" to listeners (by passing it to the event methods)

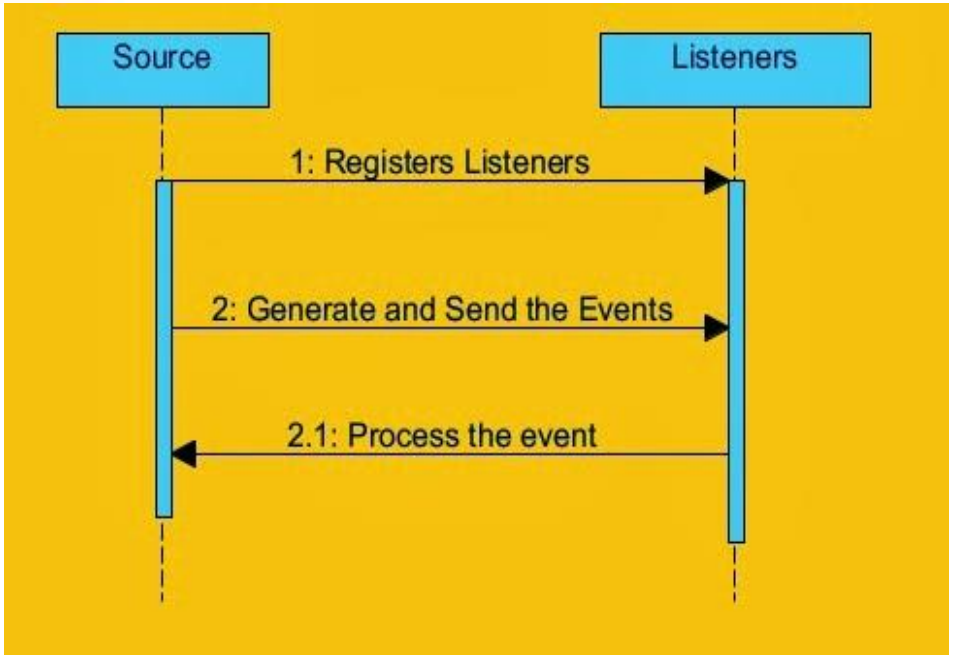
- **event object**
  1. describes the event
  2. provides methods to access information about the event (e.g., *MouseEvent.getX()*)
  3. passed to listener by event source

- **event listener interface**
  1. defines agreed upon event methods (the event source invokes these on the listener object)
  2. is implemented by event listener

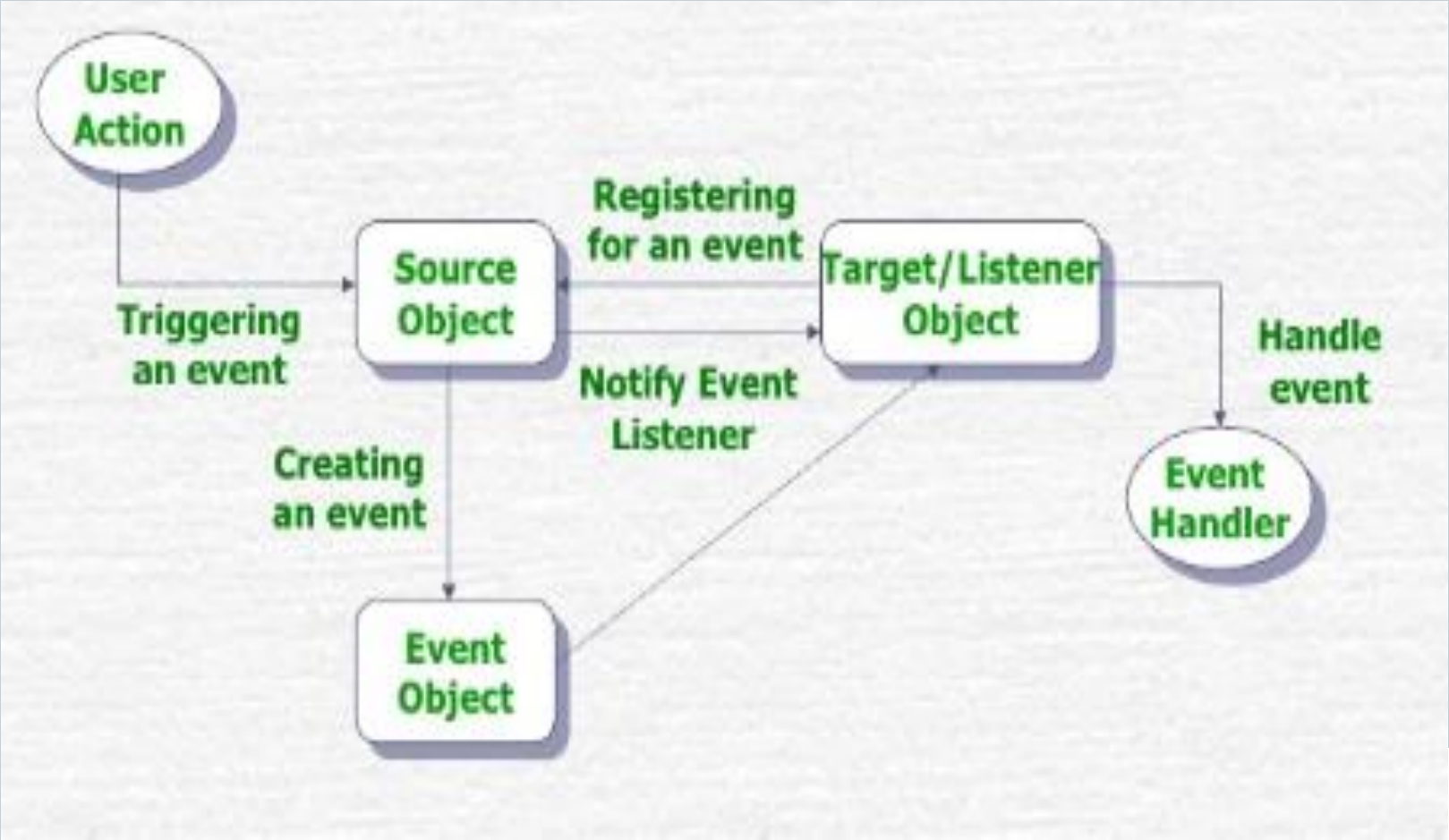
- **event listener object**
  1. registers with event source to receive events
  2. reacts to events broadcast by event source
  3. implements methods in event listener interface

Dr. Smitha Shekar B


Event Source
Invokes
creates
Event Object
registers
Listener Interface
Implements
Listener Object
onEvent(EventObj)
onEvent(EventObj)


Source
Listeners
1: Registers Listeners
2: Generate and Send the Events
2.1: Process the event

EVENT DELEGATION MODEL



Steps in event Handling:

1.The user clicks the button and the event is generated

2.The object of Concerned event class is created automatically and the information about the src and event will get populated with in the same object

3.Event object is forwarded to the method of registered listener class

4.The method now gets executed and returns.

Note: Event handling→ **DEM**

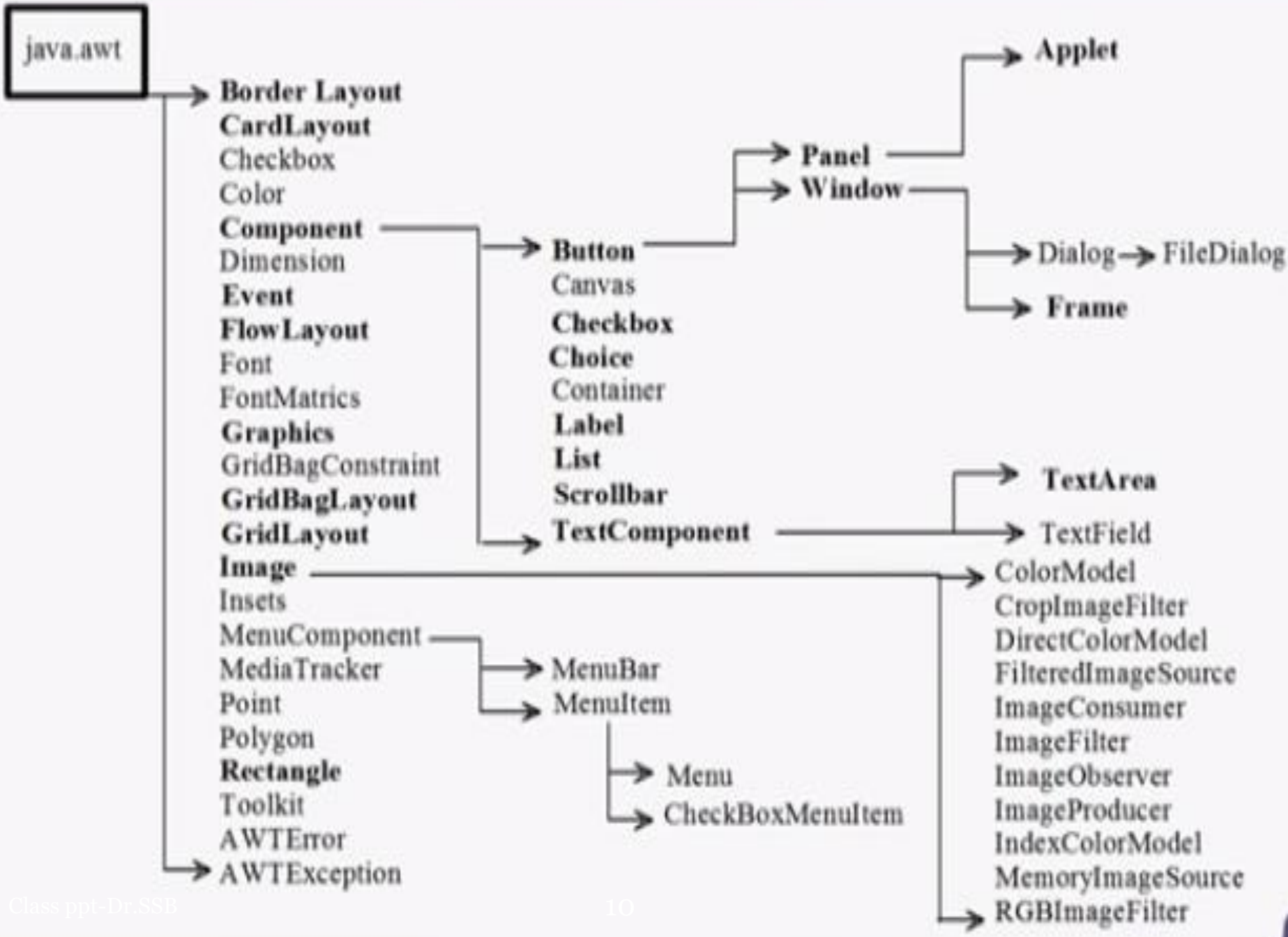
Event packages

AWT Event package

Swing Event Package

-- Event Classes

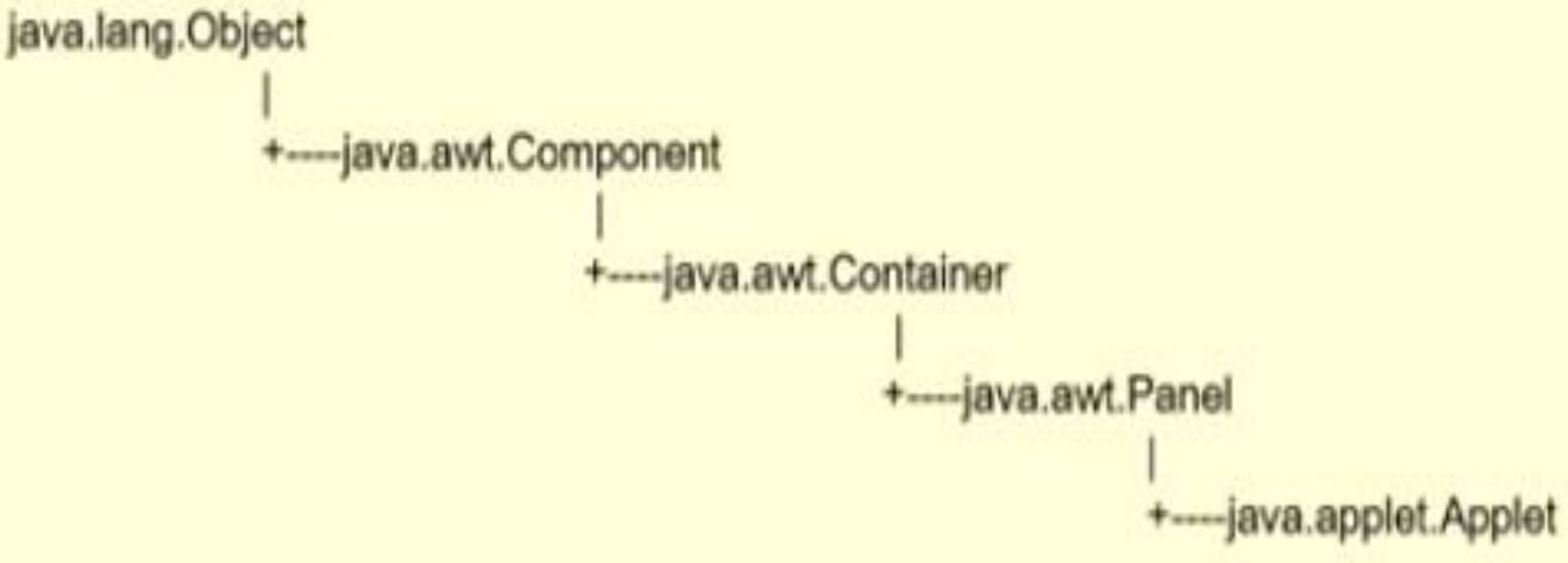
--Adapter Classes


java.awt
Border Layout
CardLayout
Checkbox
Color
Component
Dimension
Event
FlowLayout
Font
FontMatrics
Graphics
GridBagConstraint
GridBagLayout
GridLayout
Image
Insets
MenuComponent
MediaTracker
Point
Polygon
Rectangle
Toolkit
AWTError
AWTException
Button
Canvas
Checkbox
Choice
Container
Label
List
Scrollbar
TextComponent
Panel
Window
Applet
Dialog
FileDialog
Frame
TextArea
TextField
MenuBar
MenuItem
Menu
CheckBoxMenuItem
ColorModel
CropImageFilter
DirectColorModel
FilteredImageSource
ImageConsumer
ImageFilter
ImageObserver
ImageProducer
IndexColorModel
MemoryImageSource
RGBImageFilter

Abstract Window Toolkit (**AWT**) is a set of application program interfaces (APIs) used by **Java** programmers to create graphical user interface (GUI) objects, such as buttons, scroll bars, and windows.

AWT is part of the **Java Developer's Kit** (**JDK**) from Sun Microsystems, the company that originated **Java**.

```
public class Applet extends Panel
```

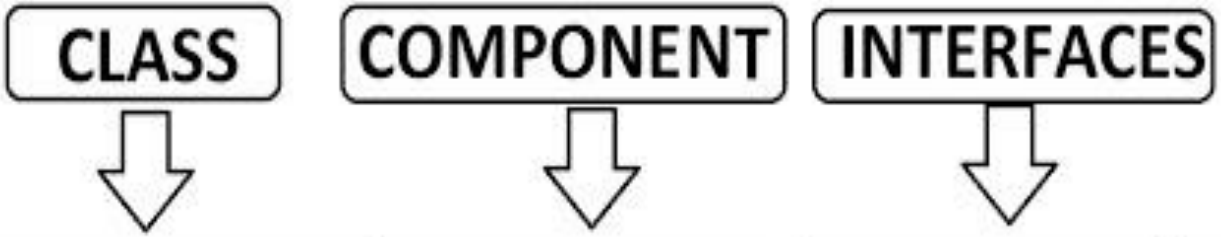


Class	Description
Button	This class creates a labeled button.
Canvas	A Canvas component represents a blank rectangular area of the screen onto which the application can draw or from which the application can trap input events from the user.
CardLayout	A CardLayout object is a layout manager for a container.
Checkbox	A check box is a graphical component that can be in either an "on" (true) or "off" (false) state.
CheckboxGroup	The CheckboxGroup class is used to group together a set of Checkbox buttons.
CheckboxMenuItem	This class represents a check box that can be included in a menu.
Choice	The Choice class presents a pop-up menu of choices.
Color	The Color class is used to encapsulate colors in the default sRGB color space or colors in arbitrary color spaces identified by a ColorSpace.
Component	A *component* is an object having a graphical representation that can be displayed on the screen and that can interact with the user.

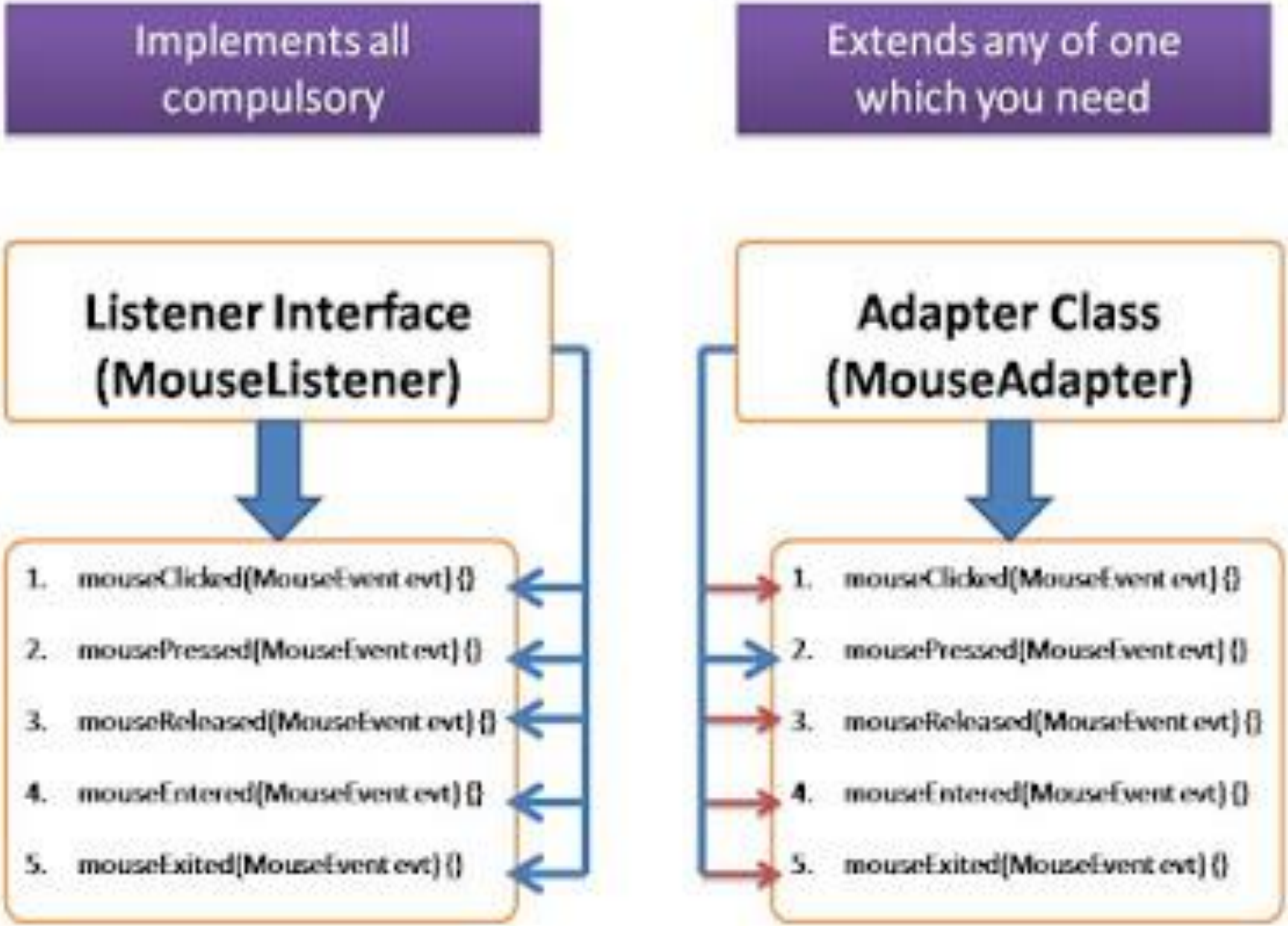
Difference between AWT and Swing

No.	Java AWT	Java Swing
1	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2	AWT components are **heavyweight**.	Swing components are **lightweight**.
3	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4	AWT provides **less components** than Swing.	Swing provides **more powerful components**such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5	AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

ADAPTER CLASSES



EVENTS	SOURCE	LISTENERS	ADAPTER CLASS
Action Event	Button, List, Menultem, Text field	ActionListener	None
Component Event	Component	Component Listener	None
Focus Event	Component	FocusListener	FocusAdapter
Item Event	Checkbox, CheckboxMenultem, Choice, List	ItemListener	None
Key Event	when input is received from keyboard	KeyListener	KeyAdapter
Text Event	Text Component	TextListener	None
Window Event	Window	WindowListener	WindowAdapter
Mouse Event	Mouse related event	MouseListener	MouseAdapter


Implements all compulsory
Extends any of one which you need
Listener Interface (MouseListener)
Adapter Class (MouseAdapter)
1. mouseClicked(MouseEvent evt) {}
2. mousePressed(MouseEvent evt) {}
3. mouseReleased(MouseEvent evt) {}
4. mouseEntered(MouseEvent evt) {}
5. mouseExited(MouseEvent evt) {}
1. mouseClicked(MouseEvent evt) {}
2. mousePressed(MouseEvent evt) {}
3. mouseReleased(MouseEvent evt) {}
4. mouseEntered(MouseEvent evt) {}
5. mouseExited(MouseEvent evt) {}

J2EE

- J2SE -→ API's needed to build Java Applications or Applets
- J2ME-→API's used to create Wireless Java Applications
- **J2EE-→** Enhanced version of J2SE→ API's to build applications for Multi-tier architecture.

Dr. Smitha Shekar B

JDBC

CGI(Common Gateway Interface) Technology

Java Servlet

JSP(JavaServer Pages)

2-Tier Architecture

- Examples

1. Client-Server Architecture

2. Data-base application

   * Software running on the Client captures a request for information from a user, then formats the request into a query that is sent over the network to the Database Server for processing.

   * The Database Server then transmits the requested data to the client where software presents data to the user.

Dr. Smitha Shekar B

JDBC
Java Data Base Connectivity

Java as database front-end

DATABASE CLIENT/SERVER METHODOLOGY

TWO-TIER DATABASE DESIGN

THREE-TIER DATABASE DESIGN

Why J2EE ??

- Collaboration of Industry leaders in JCP, resulted in **J2EE**,
  - As the industry standard Enterprise environment within which all competitive products must operate.
  - Meaning, corporate clients are assured that Server-side products they purchase are supported by J2EE.
    - Meaning, that a corporation is no longer locked into one vendor, instead products from multiple vendors can be mixed and matched based on their cost-effectiveness, while being bonded together with J2EE technology.

Dr. Smitha Shekar B

Definition

- **J2EE** is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online.

- The **J2EE** platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications.

Dr. Smitha Shekar B

- It is a versatile technology, because application components built using J2EE are able to communicate with each other behind the scene using standard communications methods such as,
  - HTTP
  - SSL(Secure Socket Layer)
  - HTML,XML
  - RMI and IIOP(Internet Inter-ORB(Object Request Broker) Protocol)

Dr. Smitha Shekar B

J2EE - standard services/technologies

- Java Servlets
- JavaServer Pages (JSP)
- Enterprise JavaBeans (EJB)
- Java Message Service (JMS)
- Java Naming and Directory Interface (JNDI)
- **Java Database Connectivity (JDBC)**
- JavaMail API
- Java Transaction Service (JTS)
- CORBA
- XML Deployment Descriptors

Dr. Smitha Shekar B

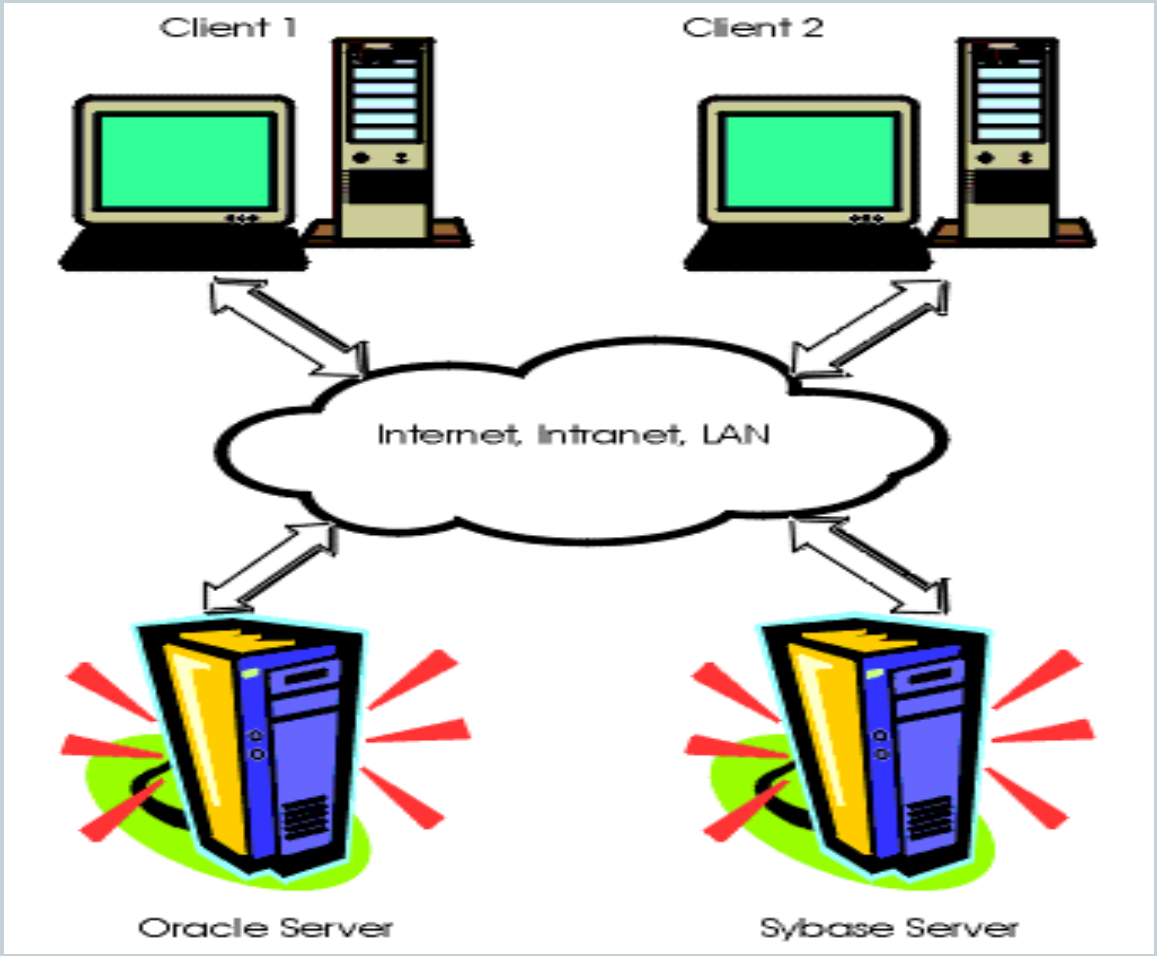
Java as database front-end

- Java offers several benefits to the developer creating a front-end application for a database server.

- Java is 'Write Once Run Everywhere' language.
  - This means that Java programs may be deployed without recompilation on any computer architectures and operating systems that possesses a Java Virtual Machine.

Dr. Smitha Shekar B

Database client-server methodolog



J2EE Multi-Tier Architecture

Software Objects – service

Protocols: 1 . CORBA
2. DCOM

Web Services- SOAP

Tier

Dr. Smitha Shekar B

- Relational databases are the most common DBMS.
- Relational databases allow the definition of relations and integrity rules between data sets.
- A **relational database** consists of a collection of tables that store interrelated data.
- E.F. Codd developed this model at the IBM San Jose Research Lab in the 1970s. A language to handle, define, and control data was also developed at the IBM lab:
  - SQL
  - SQL stands for Structured Query Language.
- SQL is a query language that interacts with a DBMS. It allows data access without supplying physical access plans, data retrieval as sets of records, and the performing of complex computations on the data.

- RDBMS stands for **R**elational **D**atabase **M**anagement **S**ystem.

- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

- A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

Software Architectures

- The first generation of client-server architectures is called two-tiered.
- It contains two active components: the client, which requests data, and the server, which delivers data. Basically, the application's processing is done separately for database queries and updates, and for user interface presentations.
- Usually the network binds the back end to the front end, although both tiers could be present on the same hardware.
- The two tiers are often called as Application layer includes JDBC drivers, business logic and user interfaces whereas second layer i.e. Database layer consists of RDBMS server.

Dr. Smitha Shekar B

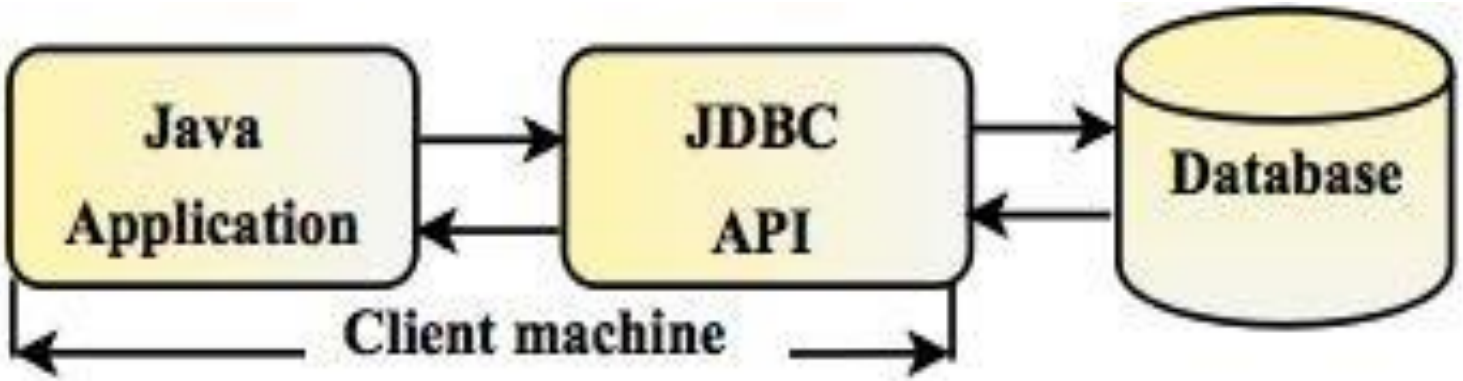


Fig: Two-tier Architecture of JDBC

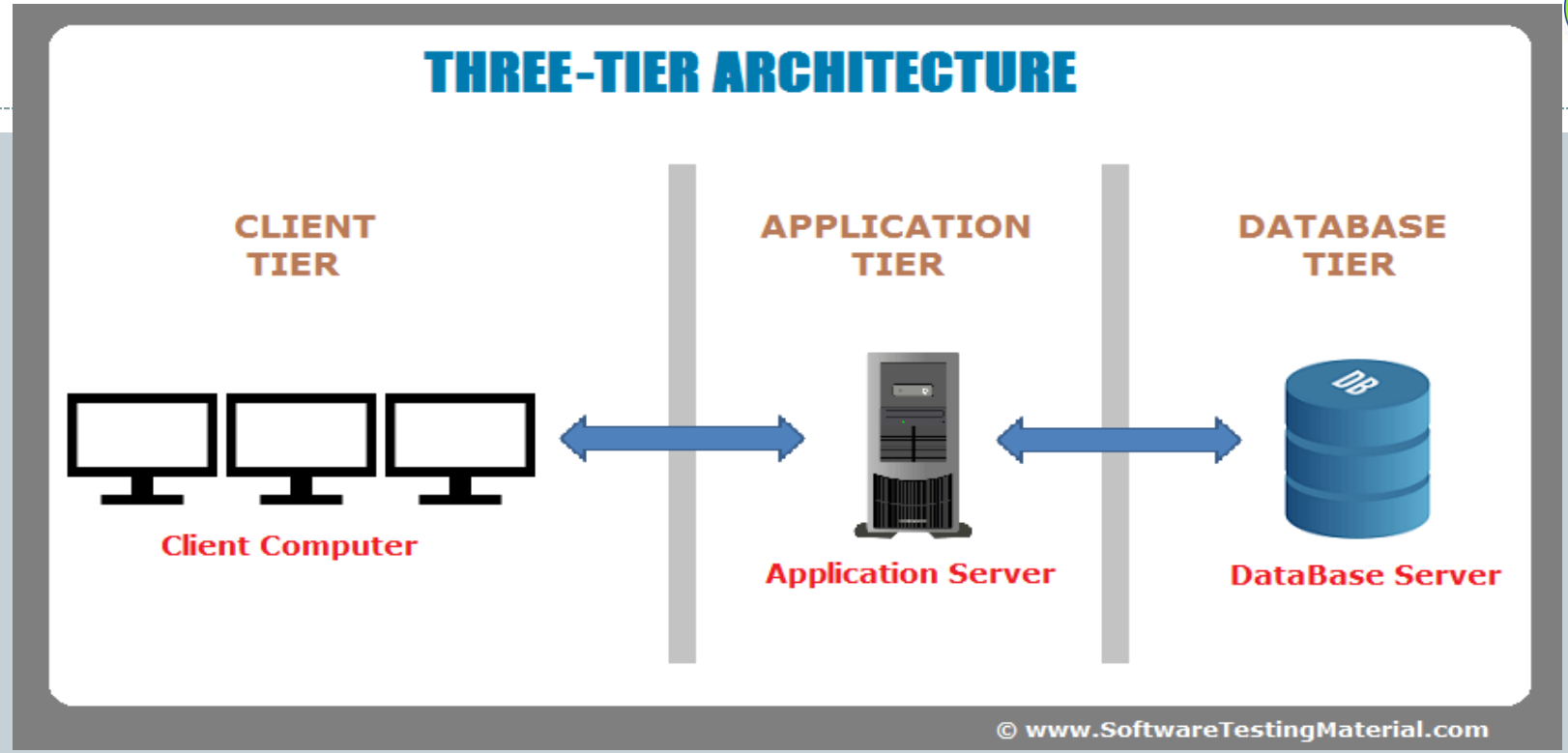
Although the two-tiered architecture is common, another design is starting to appear more frequently.

To avoid embedding the application's logic at both the database side and the client side, a third software tier may be inserted.

- In three-tiered architectures, most of the business logic is frozen in the middle tier.
  - In this architecture, when the business activity or business rules change, only the middleware must be modified.

- Figure below illustrates the three-tier architecture.

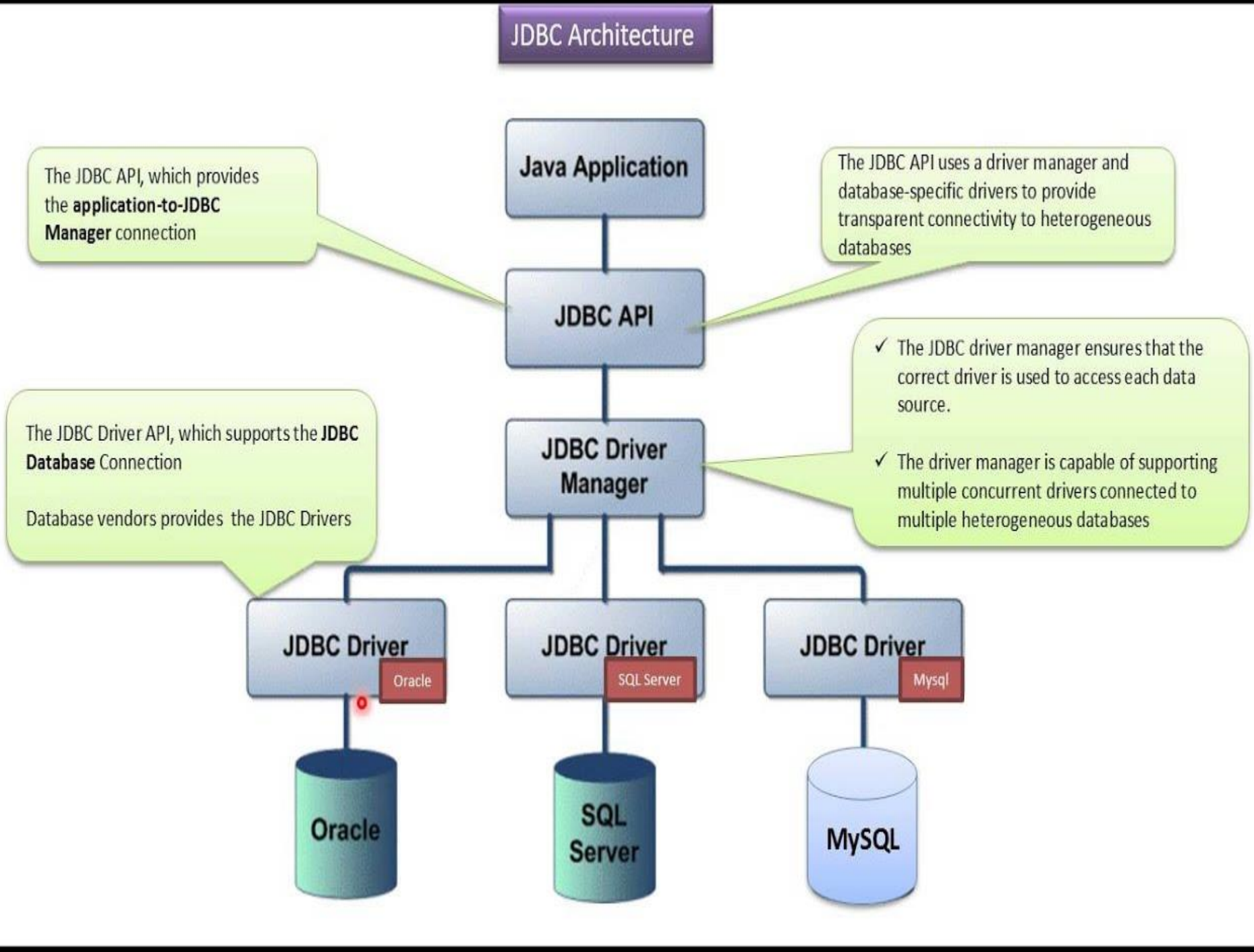
Dr. Smitha Shekar B



Advantages:

- Flexible: It can change one part without affecting others.
- It can connect to different databases without changing code.
- Specialization: presentation / business logic / data management.

Dr. Smitha Shekar B


JDBC Architecture
Java Application
The JDBC API, which provides the application-to-JDBC Manager connection
The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases
JDBC API
✓ The JDBC driver manager ensures that the correct driver is used to access each data source.
✓ The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases
JDBC Driver Manager
The JDBC Driver API, which supports the JDBC Database Connection
Database vendors provides the JDBC Drivers
JDBC Driver
Oracle
JDBC Driver
SQL Server
JDBC Driver
Mysql
Oracle
SQL Server
MySQL

The JDBC Structure

- The JDBC is two-dimensional. The reasoning for the split is to separate the low-level programming from the high-level application interface.
  - The low-level programming is the JDBC driver. The idea is that database vendors and third-party software vendors will supply pre-built drivers for connecting to different databases.
  - JDBC drivers are quite flexible: They can be local data sources or remote database servers. The implementation of the actual connection to the data source/database is left entirely to the JDBC driver.
- **The structure of the JDBC includes these key concepts:**
- **The goal of the JDBC is**
  - **a DBMS independent interface,**
  - **a "generic SQL database access framework," and**
  - **a uniform interface to different data sources.**
- **The programmer writes only one database interface; using JDBC, the program can access any data source without recoding.**

Dr. Smitha Shekar B

Database Driver

- A **database driver** is a computer program that implements a protocol (ODBC or **JDBC**) for a **database** connection.
- The **driver** works like an adaptor which connects a generic interface to a specific **database** vendor implementation. ...
- To connect with individual databases, **JDBC** requires **drivers** for each specific **database** type.

Dr. Smitha Shekar B

Aided By Govt. of Karnataka

Drivers

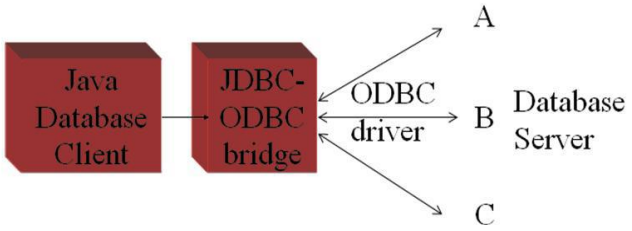
Dr. Smitha Shekar B

JDBC drivers

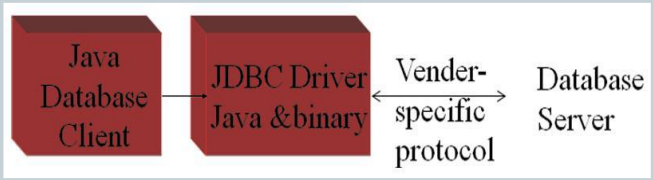
- Sun has defined four categories of JDBC drivers. The categories delineate the differences in architecture for the drivers.
- One difference between architectures lies in whether a given driver is implemented in native code or in Java code.
  - Native code means whatever machine code is supported by a particular hardware configuration.
  - For example, a driver may be written in C and then compiled to run on a specific hardware platform.
- Another difference lies in how the driver makes the actual connection to the database.
- The four driver types are as follows:

Dr. Smitha Shekar B

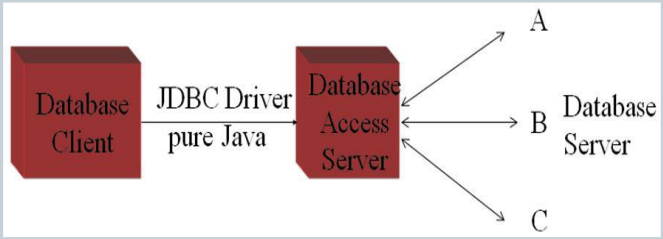
- Type 1
  JDBC-ODBC : by sun.com



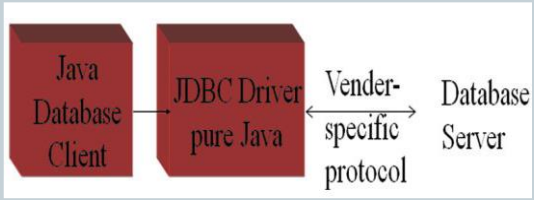
- Type 2
  Native API: partly Java Driver



- Type 3
  Net Protocol: All Java Driver



- Type 4
  Native Protocol: Pure Java Driver



Dr. Smitha Shekar B

Type 1 Driver: JDBC/ODBC Bridge

- This type uses bridge technology to connect a Java client to a third-party API such as Open Data Base Connectivity (ODBC).
- Sun's JDBC-ODBC bridge is an example of a Type 1 driver.
- These drivers are implemented using native code. This driver connects Java to a Microsoft ODBC (Open Database Connectivity) data source.
- JDBC-to-ODBC bridge driver (sun.jdbc.odbc.JdbcOdbcDriver)

Dr. Smitha Shekar B

Type 2 Driver: Native API Driver

- This type of driver wraps a native API with Java classes. The Oracle Call Interface (OCI) driver is an example of a Type 2 driver.

Dr. Smitha Shekar B

Type 3 Driver: Network Protocol, Pure Java Driver

- These drivers take JDBC requests and translate them into a network protocol that is not database specific. These requests are sent to a server, which translates the database requests into a database-specific protocol.
- This type of driver communicates using a network protocol to a middle tier server. The middle tier in turn communicates to the database. Oracle does not provide a Type 3 driver. They do, however, have a program called Connection Manager that, when used in combination with Oracle's Type 4 driver, acts as a Type 3 driver in many respects

Dr. Smitha Shekar B

Type 4 Driver: Native Protocol, Pure Java Driver

- These convert JDBC requests to database-specific network protocols, so that Java programs can connect directly to a database.
- This type of driver, written entirely in Java, communicates directly with the database. No local native code is required.
- Oracle's thin driver is an example of a Type 4 driver.

Dr. Smitha Shekar B

Oracle Thin JDBC Driver

- The JDBC Thin driver is a pure Java, Type IV driver that can be used in applications and applets.
- It is platform-independent and does not require any additional Oracle software on the client-side.
- The JDBC Thin driver communicates with the server using SQL*Net(Oracle's networking software that allows remote data access between programs and the Oracle Database, or among multiple Oracle Databases).
- The JDBC Thin driver allows a direct connection to the database by providing an implementation of SQL*Net on top of Java sockets.
- The driver supports the TCP/IP protocol and requires a TNS (Transparent Network Substrate)listener on the TCP/IP sockets on the database server.

Dr. Smitha Shekar B

What is JDBC driver for SQL Server?

- The **driver** is available at no additional charge and provides Java database connectivity from any Java application, application **server**, or Java-enabled applet.
- This **driver** is a Type 4 **JDBC driver** that provides database connectivity through the standard **JDBC** application program interfaces (APIs).

Dr. Smitha Shekar B

The JDBC API

- The JDBC API is contained in two packages named java.sql and javax.sql.

- The java.sql package contains core Java objects of JDBC API.

  java.sql is an API to access and process the data stored in a database, typically a relational database using Java.

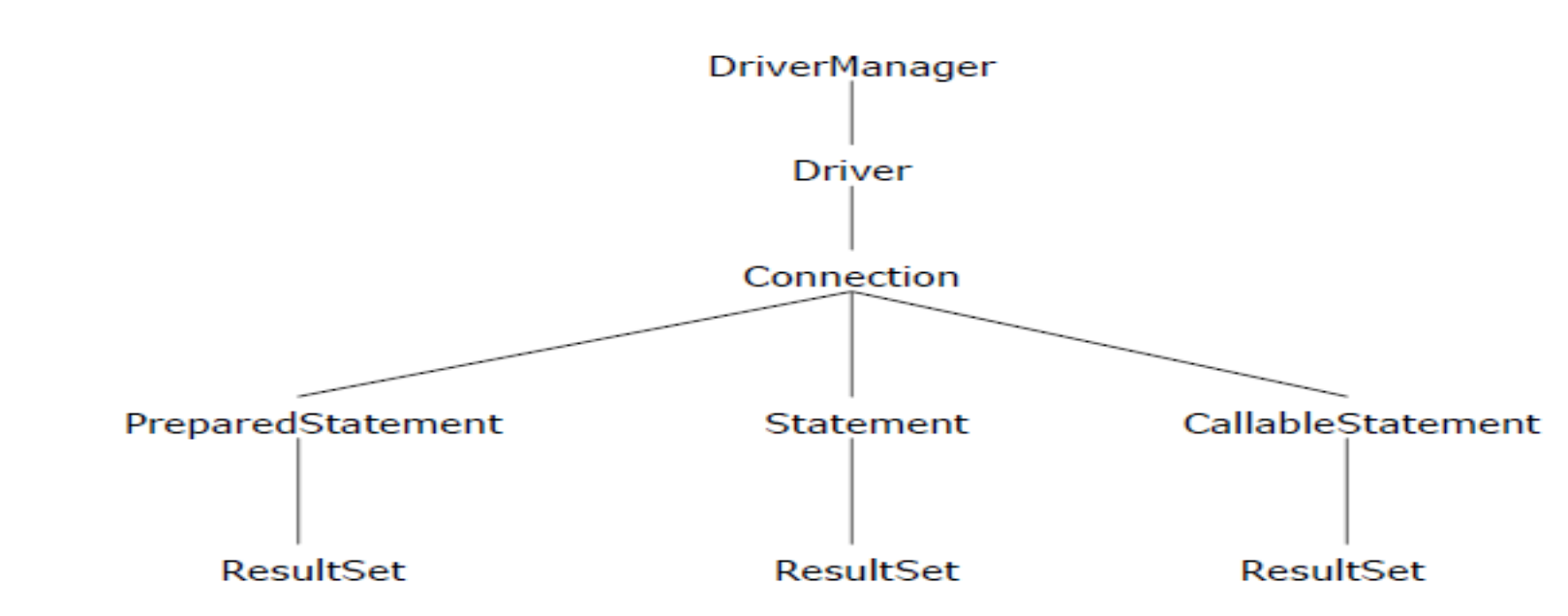
  javax.sql is a JDBC API for the server side for accessing and processing the data from the databases typically a relational database using Java. It is the essential part for J2EE.

- There are two distinct layers within the JDBC API:
  - **application** layer, which database application developers use and
  - **driver** layer which the drivers vendors implement.

Dr. Smitha Shekar B

Figure: The JDBC API

- The connection between application and driver layers is illustrated in figure below.



Dr. Smitha Shekar B

The JDBC process

Dr. Smitha Shekar B

There are six different steps to use JDBC in our Java application program.

Phase	Task	Relevant java.sql classes
Initialisation	Load driver	DriverManager
	Create connection	Connection
Processing	Generate SQL statements	Statement
	Process result data	ResultSet
Termination	Terminate connection	Connection
	Release data structures	Statement

Dr. Smitha Shekar B

There are 7- different steps to use JDBC in our Java application program

1. Import package
2. Load and Register the driver
3. Define and establish the Connection
4. Create a Statement object
5. Execute a query
6. Process the results
7. Close the connection

Dr. Smitha Shekar B

What will Class.forName do while loading drivers?

- When you have loaded a driver, it is available for making a connection with a DBMS.
  - It is used to create an instance of a driver and register it with the DriverManager.
- Before connecting to the database we need to load a driver.
- The database drivers are to be loaded by Java Virtual Machine class loader.
- By using Class.forName() method, the driver class is loaded. The driver class is vendor and driver type specific. The vendors supplies the specific driver classes.

Mysql Driver:

Class.*forName("com.mysql.jdbc.Driver");*

Dr. Smitha Shekar B

Purpose of DriverManager

- DriverManager looks after managing the drivers for a JDBC application.
- When it is instantiated it makes an attempt to load the driver classes.
- When the method getConnection() is invoked, the driver manager attempts to locate the suitable driver.
- The DriverManager obtains the information about the drivers such as registering, locating, finding the drivers loaded, setting the time to wait when it tries to get the connection to a database.

Dr. Smitha Shekar B

To open a database connection using JDBC

- **Opening a database connection**

  The database connection should be established after registering the drivers.
- The getConnection is used to open a database connection.

Connection

con=DriverManager.*getConnection("jdbc:mysql://localhost:3306/DBName","root","root");*

returns the connection object

Dr. Smitha Shekar B

To send SQL statements to databases for execution

- The SQL statements are to be created as objects. These objects are to be assigned to Statement reference.
  Statement stmt=con.createStatement();

Dr. Smitha Shekar B

Execute query and Process Results

- The specific SQL statements are to be executed and the result should be captured

ResultSet rs=stmt.executeQuery("select * from emp");

Dr. Smitha Shekar B

Close Statement and Connection

- con.close();

Dr. Smitha Shekar B

Connecting to MySQL database

- First, you need to import three classes

- SQLException, DriverManager, and Connection from the java.sql.* package.

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

Dr. Smitha Shekar B

- Second, you call the getConnection() method of the DriverManager class to get the Connection object. There are three parameters you need to pass to the getConnection() method:
- url: the database URL in the form jdbc:subprotocol:subname.

- For MySQL,
- use the **URL**
  - jdbc:mysql://localhost:3306/mysqljdbc
  - i.e., you are connecting to the MySQL with server name localhost, port 3306, and database mysqljdbc.
- **user**: the database user that will be used to connect to MySQL.
- **password**: the password of the database user.

Dr. Smitha Shekar B

Statement Objects

- 3 types of Statement objects is used to execute the query.
- These objects are
  - Statement, which executes a query immediately;
  - PreparedStatement, which is used to execute a compiled query
  - CallableStatement, which is used to execute store procedures.

Dr. Smitha Shekar B

Statement

- The Statement object contains ,
- The executeQuery() which is passed the query as an argument. The query is then transmitted to the DBMS for processing.
- It returns one ResultSet object that contains rows, columns, and metadata that represent data requested by the query.
- The execute() method of the Statement object is used when multiple results may be returned.
- The executeUpdate() is used to execute queries that contain INSERT, UPDATE, DELETE, and DDL statements.,which change values in a row and remove a row, respectively. It returns an integer indicating the number of rows that were updated by the query.
- The createStatement() method of the Connection object is called to return a Statement object

Dr. Smitha Shekar B

- Ex: executeQuery

```
String query = "SELECT * FROM Customers";
DataRequest = Db.createStatement();
Results = DataRequest.executeQuery(query);
```

- Ex: executeUpdate

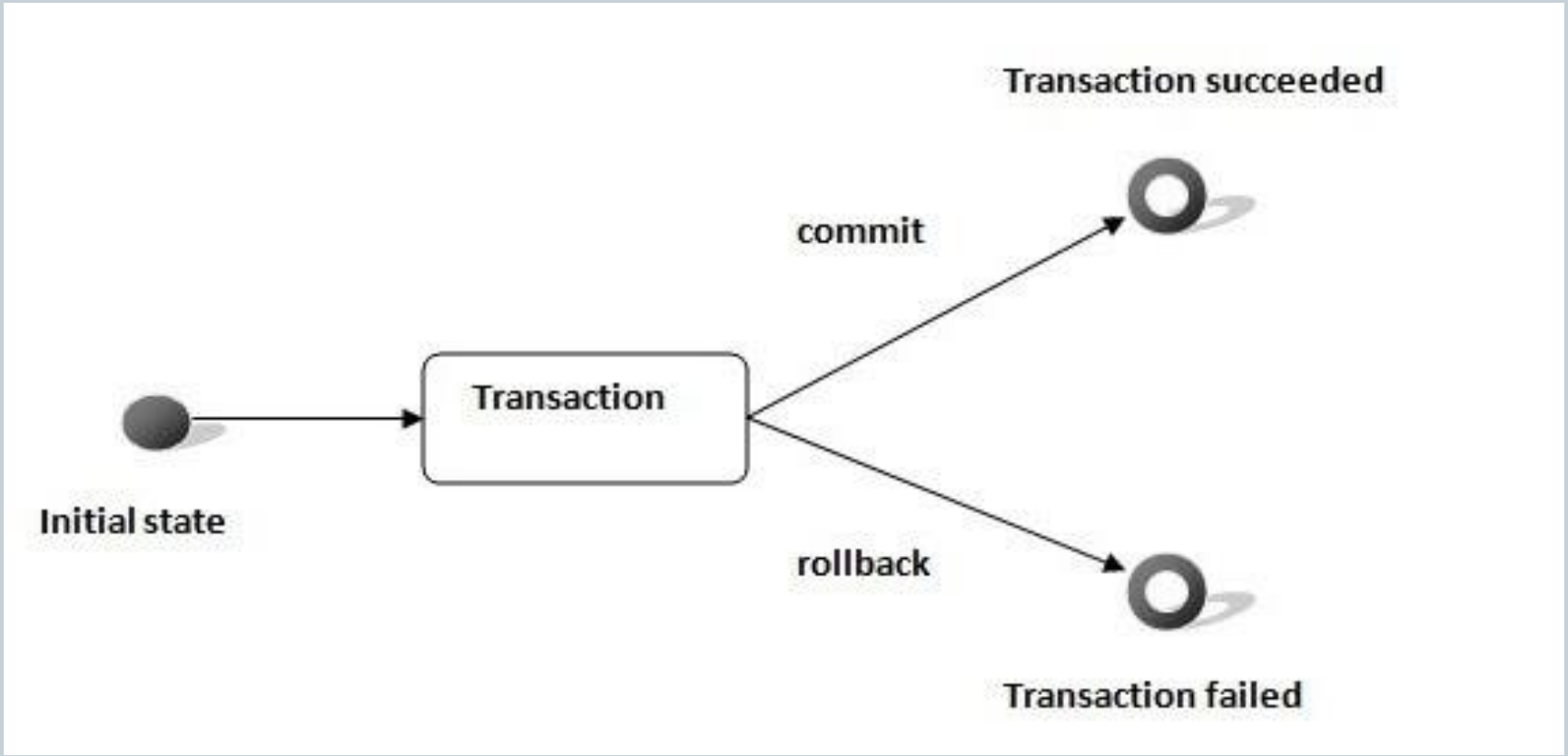
```
SET PAID = 'Y' WHERE BALANCE = '0';
DataRequest = Db.createStatement();
rowsUpdated = DataRequest.executeUpdate(query);
```

Dr. Smitha Shekar B

JDBC Transaction Management

- Transaction represents **a single unit of work.**
- The ACID properties describes the transaction management well.
- ACID stands for Atomicity, Consistency, isolation and durability.
  - **Atomicity** means either all successful or none.
  - **Consistency** ensures bringing the database from one consistent state to another consistent state.
  - **Isolation** ensures that transaction is isolated from other transaction.
  - **Durability** means once a transaction has been committed, it will remain so, even in the event of errors, power loss etc.

Dr. Smitha Shekar B



Dr. Smitha Shekar B

- Advantage of Transaction Management
  - **Fast performance**
    - It makes the performance fast because database is hit at the time of commit.

Dr. Smitha Shekar B

- In JDBC, **Connection interface** provides methods to manage transaction.

Method	Description
void setAutoCommit(boolean status)	It is true bydefault means each transaction is committed bydefault.
void commit()	commits the transaction.
void rollback()	cancels the transaction.

Dr. Smitha Shekar B

Web Links

- https://docs.oracle.com/javase/tutorial/jdbc/basic
- http://www.mysqltutorial.org/connecting-to-mysql-using-jdbc-driver/s/connecting.html

Dr. Smitha Shekar B

THANK YOU