Unit 4

JavaBean

A JavaBean is a Java class that should follow the following conventions:

- It should have a no-arg constructor.
- It should be Serializable.
- It should provide methods to set and get the values of the properties, known as getter and setter methods.

The **Java Bean components** can exist in one of the following three phases of development:

- **Construction phase**- involves creation of java bean and its user interface.
- **Build phase**- involves placing the java bean into target container.
- **Execution phase**- starts when container application is executed.

Steps for Creating a JavaBeans

Step 1: Put this source code into a file named "SimpleBean.java"

```
import java.awt.*;
import java.io.Serializable;

public class SimpleBean extends Canvas
  implements Serializable{

//Constructor sets inherited properties
  public SimpleBean(){
    setSize(60,40);
    setBackground(Color.red);
  }
}
```

Step 2: Compile the file:

javac SimpleBean.java

Step 3: Create a manifest file, named "manifest.tmp":

```
Name: SimpleBean.class
Java-Bean: True
```

Step 4: Create the JAR file, named "SimpleBean.jar":

```
jar cfm SimpleBean.jar manifest.tmp SimpleBean.class
```

Then verify that the content is correct by the command "jar tf SimpleBean.jar".

Step 5:

1. Start the Bean Box.
   CD to c:\Program Files\BDK1.1\beanbox\.
   Then type "run".
2. Load JAR into Bean Box by selecting "LoadJar..." under the File menu.

Step 6:

1. After the file selection dialog box is closed, change focus to the "ToolBox" window. You'll see "SimpleBean" appear at the bottom of the toolbox window.
2. Select SimpleBean.jar.
3. Cursor will change to a plus. In the middle BeanBox window, you can now click to drop in what will appear to be a colored rectangle.

Step 7:
Try changing the red box's color with the Properties windows.
Step 8:
Chose "Events" under the "Edit" menu in the middle window to see what events SimpleBean can send. These events are inherited from java.awt.Canvas.

JAR Files:

A java archive file is a file format/ archiving tool which contains all the components of an executable Java application. All the predefined libraries are available in this format.

To include any of these (other than rt.jar) in to your project you need to set the class path for this particular JAR file. You can create a JAR file using the command line options or using any IDE's.

Creating a Jar file

You can create a Jar file using the jar command as shown below.

jar cf jar-file input-file(s)

JavaBeans Properties

A JavaBean property is a named attribute that can be accessed by the user of the object. The attribute can be of any Java data type, including the classes that you define.

A JavaBean property may be **read, write, read only**, or **write only**. JavaBean properties are accessed through two methods in the JavaBean's implementation class −

S.No.	Method & Description
1	get**PropertyName**() For example, if property name is *firstName*, your method name would be **getFirstName()** to read that property. This method is called accessor.
2	set**PropertyName**() For example, if property name is *firstName*, your method name would be **setFirstName()** to write that property. This method is called mutator.

Enterprise Java Bean

- EJB is an acronym for *enterprise java bean*.

- It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.
- EJB application is deployed on the server, so it is called server side component also.
- EJB is like COM (*Component Object Model*) provided by Microsoft. But, it is different from Java Bean, RMI and Web Services.

To run EJB application, you need an *application server* (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:

a. life cycle management,
b. security,
c. transaction management, and
d. object pooling.

When use Enterprise Java Bean?

1. **Application needs Remote Access**. In other words, it is distributed.
2. **Application needs to be scalable**. EJB applications supports load balancing, clustering and fail-over.
3. **Application needs encapsulated business logic**. EJB application is separated from presentation and persistent layer.

Types of Enterprise Java Bean

There are 3 types of enterprise bean in java- session, message driven & entity bean.

Session Bean

Session bean contains business logic that can be invoked by local, remote or webservice client.

Message Driven Bean

Like Session Bean, it contains the business logic but it is invoked by passing message.

Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

Advantages of EJB:

- A JavaBean can be registered to receive events from other objects.
- As JavaBeans are built in Java, we can easily port them to any other platform that contains JRE.
- JavaBeans are light weighted, I.e., we don't need to fulfill any special requirement to use it. Also, it is very easy to create them. However, it doesn't need a complex system to register components with JRE.
- JavaBeans include reusability, deployment, and customization that can be archived using JavaBeans.

Disadvantages of EJB

- Requires application server
- Requires only java client. For other language client, you need to go for webservice.
- Complex to understand and develop ejb applications.

Unit 4

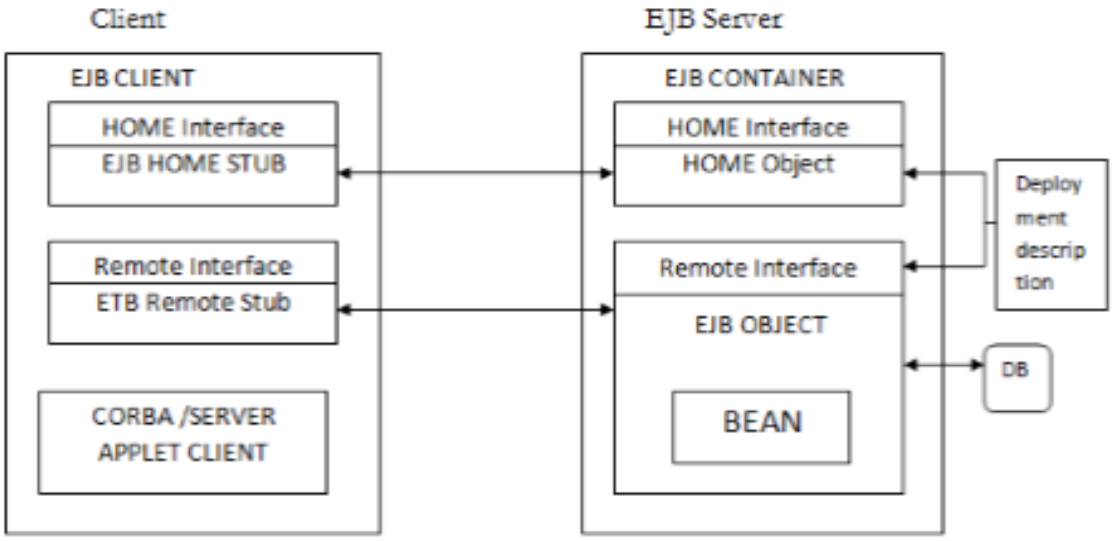
Session Bean

- Session bean encapsulates business logic only, it can be invoked by local, remote and webservice client.
- It can be used for calculations, database access etc.
- The life cycle of session bean is maintained by the application server (EJB Container).

Types of Session Bean

There are 3 types of session bean.
1) Stateless Session Bean: It doesn't maintain state of a client between multiple method calls.
2) Stateful Session Bean: It maintains state of a client across multiple requests.
3) Singleton Session Bean: One instance per application, it is shared between clients and supports concurrent access.

EJB Architecture:



- EJB (Enterprise Java Beans) is a server-side component that executes specific business logic.
- It enables development and deployment of component based, robust, highly scalable & transactional business application.

1) EJB Server :

- It provides execution environment for server components.
- It hosts EJB containers.

2) EJB Container:

- It serves as interface EJB and Outside World.

Container hosts following components:

Unit 4

- EJB Object: A client invokes beans instance through EJB Object. It is called Request Interceptor.
- Remote Interface: It duplicates machines of bean class.
- Home Object: It is the object factory bean where client can create or destroy object.
- Home Interface: It provides machine for creating and destroying, finding bean object.
- Local Interface: It is used instead of EJB objects for faster access.
- The type of container depends on beans they contains i.e either transient or persistent beans.

3) Enterprise Beans:

- They are EJB Components that encapsulates business functionalities in an application.
- Container provides security to enterprise beans.
- Types of Enterprise Beans:
  - Session Beans(Synchronous)
  - Message Driven(Asynchronous)
  - Entity Beans (Data/Records in DB).

4) EJB Client:

- It makes use of EJB beans to perform the operations.
- Client locates the container using JNDI (Java Naming & Directory Interface)
- Then the client invokes JB Beans machines through container.

5) Deployment Descriptor:

- It lists the bean properties and elements like:
  - JNDI Name for Bean.
  - Home and Remote Interface.
  - Bean Implementation Class.
  - Environment Variables.
  - Access Rules or Rights.
  - Beans Management, Etc.

Difference between JavaBeans & EJB:

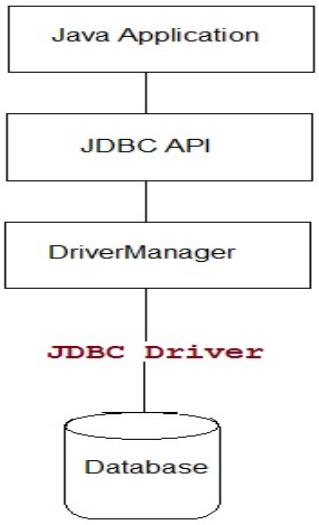
S.No.	JavaBeans	Enterprise JavaBeans
1.	Javabeans is a component technology to create universal Java components.	Even though EJB is a component technology, it neither reconstructs nor enhances the original JavaBean specification.
2.	Beaninfo classes, property editors or customizers can be present in Javabeans.	No perception of Beaninfo classes, property editors or customizers is in Enterprise JavaBeans and no additional information is provided except that described in the deployment descriptor.
3.	An external interface called a properties interface is given in JavaBeans, that allows a builder tool to describes the	A deployment descriptor is given in Enterprise JavaBeans to interpret the functionality to an external builder tool or

	functionality of a bean.	IDE.
4.	There is no further category of Java Beans.	Enterprise JavaBeans are categorized into two types – session beans and entity beans.
5.	JavaBeans have not any definite support exists for transactions.	EJBs may be transactional and transactional support is provided by the EJB servers.
6.	JavaBeans are designed for a single process and are localized	EJBs are remotely executable components or business objects.
7.	JavaBeans may be visible or non-visible components.Visual GUI component (Button, listbox, graphics) are examples of JavaBeans.	An EJB is a non-visual isolated object.
8.	JavaBeans has components bridges.A JavaBeans can also be arranged as an ActiveX control.	ActiveX controls are designed for the desktop so EJB cannot be deployed as an ActiveX control.
9.	JavaBeans are mainly designed to run on the client side whereas one can develop server side JavaBeans.	EJBs are deployed on the server-side only.

Introduction to JDBC

- **Java Database Connectivity(JDBC)** is an **Application Programming Interface(API)** used to connect Java application with Database.
- JDBC is used to interact with various type of Database such as Oracle, MS Access, My SQL and SQL Server.
- JDBC can also be defined as the platform-independent interface between a relational database and Java programming.
- It allows java program to execute SQL statement and retrieve result from database.
- The JDBC API consists of classes and methods that are used to perform various operations like: connect, read, write and store data in the database.

Unit 4



Java introduced **JDBC 4.0**, a new version which is advance specification of JDBC. It provides the following advance features

- Connection Management
- Auto loading of Driver Interface.
- Better exception handling
- Support for large object
- Annotation in SQL query.

JDBC Driver

- JDBC Driver is required to establish connection between application and database.
- It also helps to process SQL requests and generating result.
- Different types of driver available in JDBC are:

Type-1 Driver or **JDBC-ODBC bridge**
Type-2 Driver or **Native API Partly Java Driver**
Type-3 Driver or **Network Protocol Driver**
Type-4 Driver or **Thin Driver**

JDBC-ODBC bridge

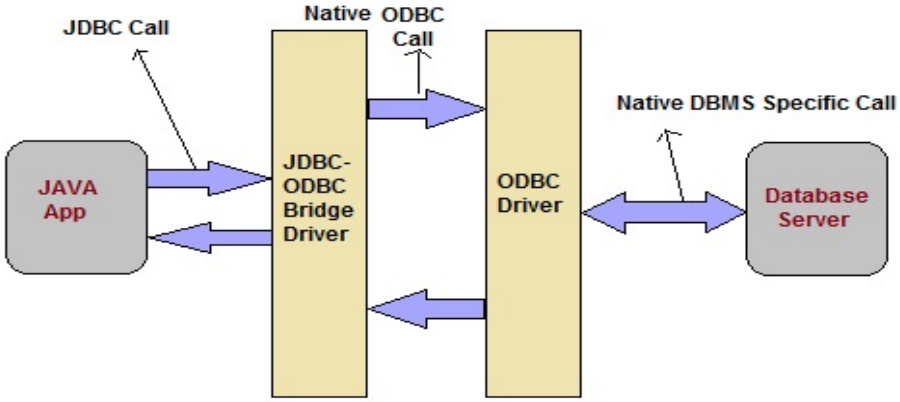
Type-1 Driver act as a bridge between JDBC and other database connectivity mechanism (ODBC). This driver converts JDBC calls into ODBC calls and redirects the request to the ODBC driver.

Advantage

- Easy to use
- Allow easy connectivity to all database supported by the ODBC Driver.

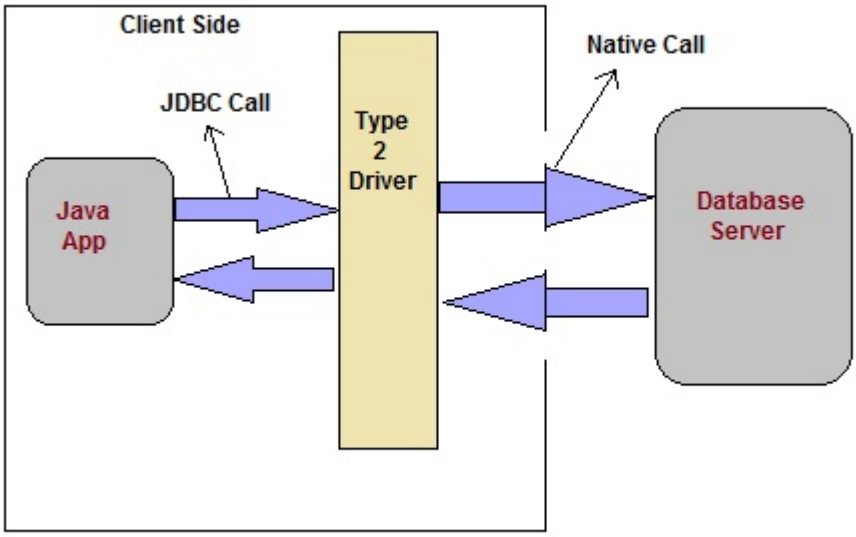
Disadvantage

- Slow execution time
- Dependent on ODBC Driver.
- Uses Java Native Interface(JNI) to make ODBC call.



Native API Driver

This type of driver make use of Java Native Interface(JNI) call on database specific native client API. These native client API are usually written in C and C++.



Advantage

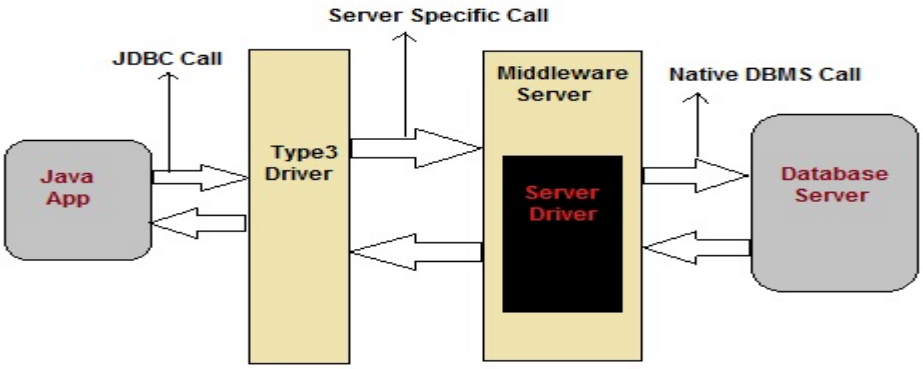
- faster as compared to **Type-1 Driver**
- Contains additional features.

Disadvantage

- Requires native library
- Increased cost of Application

Network Protocol Driver

This driver translate the JDBC calls into a database server independent and Middleware server-specific calls. Middleware server further translate JDBC calls into database specific calls.



Advantage

- Does not require any native library to be installed.
- Database Independency.
- Provide facility to switch over from one database to another database.

Disadvantage

- Slow due to increase number of network call.

Thin Driver

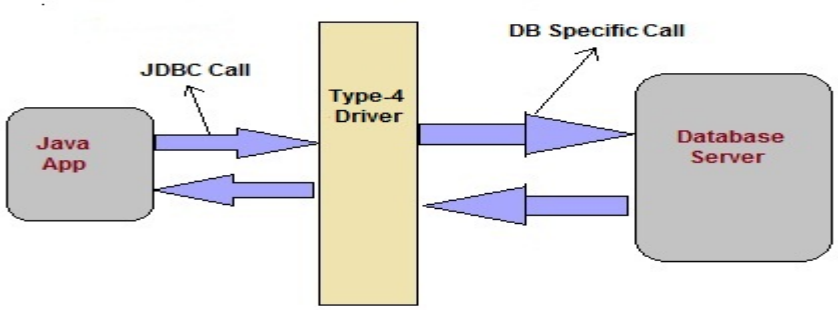
This is Driver called Pure Java Driver because. This driver interact directly with database. It does not require any native database library, which is why it is also known as Thin Driver.

Advantage

- Does not require any native library.
- Does not require any Middleware server.
- Better Performance than other driver.

Disadvantage

- Slow due to increase number of network call



Steps for database connection:

Unit 4

Step 1) load the JDBC driver
Step 2) Specify the name and location of the database being used
Step 3) Connect to the database with a Connection object
Step 4) Execute a SQL query using a Statement object
Step 5) Get the results in a ResultSet object
Step 6) Finish by closing the ResultSet, Statement and Connection objects

Set up a database server (Oracle , MySQL, pointbase)
Get a JDBC driver
- set CLASSPATH for driver lib
  - Set classpath in windows, control panel->system->advanced->environment variable
  - Set classpath in Solaris, set CLASSPATH to driver jar file

Import the library
- import java.sql.*;

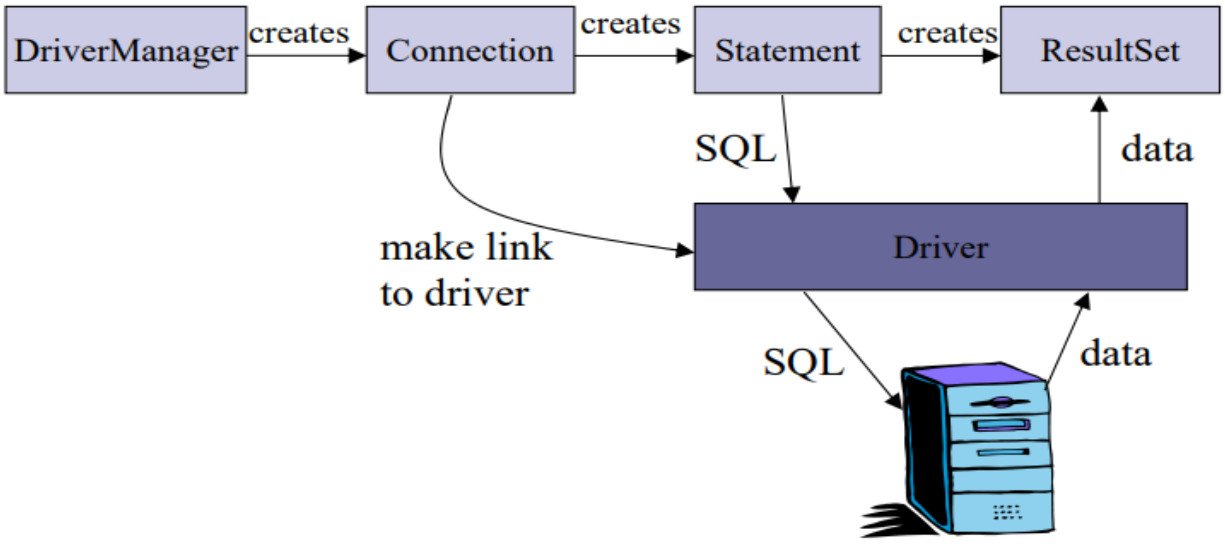
Specify the URL to database server
- String **url = "jdbc:pointbase://127.0.0.1/test"**

Load the JDBC driver
- Class.forName("**com.pointbase.jdbc.jdbcUniversalDriver**");

Connect to database server
- Connection con = DriverManager.getConnection(url, "dbUser", "dbPass");

Create SQL Statement
- stmt = con.createStatement();

Execute SQL
- stmt.executeUpdate("insert into COFFEES " + "values('Colombian', 00101, 7.99, 0, 0)");
- ResultSet rs = stmt.executeQuery(query);



Unit 4

Difference between Statement, Prepared Statement, Callable statement:

Statement	Prepared Statement	Callable Statement
Super interface for Prepared and Callable Statement	extends Statement (sub-interface)	extends PreparedStatement (sub-interface)
Used for executing simple SQL statements like CRUD (create, retrieve, update and delete	Used for executing dynamic and pre-compiled SQL statements	Used for executing stored procedures
The Statement interface cannot accept parameters.	The PreparedStatement interface accepts input parameters at runtime.	The CallableStatement interface can also accept runtime input parameters.
stmt = conn.createStatement();	PreparedStatement ps=con.prepareStatement ("insert into studentDiet values(?,?,?)");	CallableStatement cs=conn.prepareCall("{call getbranch(?,?)}");
java.sql.Statement is slower as compared to Prepared Statement in java JDBC.	PreparedStatement is faster because it is used for executing precompiled SQL statement in java JDBC.	None
java.sql.Statement is suitable for executing DDL commands - CREATE, drop, alter and truncate in java JDBC.	java.sql.PreparedStatement is suitable for executing DML commands - SELECT, INSERT, UPDATE and DELETE in java JDBC.	java.sql.CallableStatement is suitable for executing stored procedure.

Write a program to insert student records to database using prepared statement

```
1. import java.sql.*;
2. public class PreparedInsert {
3. public static void main(String[] args) {
4. try {
5. Class.forName("com.mysql.jdbc.Driver");
6. Connection conn= DriverManager.getConnection
7. ("jdbc:mysql://localhost:3306/DIET", "root","pwd");

8. String query="insert into dietstudent values(?,?,?,?)";
9. PreparedStatement ps=conn.prepareStatement(query);
10.     ps.setString(1, "14092"); //Enr_no

11.     ps.setString(2, "abc_comp"); //Name
12.     ps.setString(3, "computer"); //Branch
13.     ps.setString(4, "cx"); //Division
14.     int i=ps.executeUpdate();

15.     System.out.println("no. of rows updated ="+i);
16.     ps.close();
17.     conn.close();
18.     }catch(Exception e){System.out.println(e.toString());} }//PSVM
  }//class
```

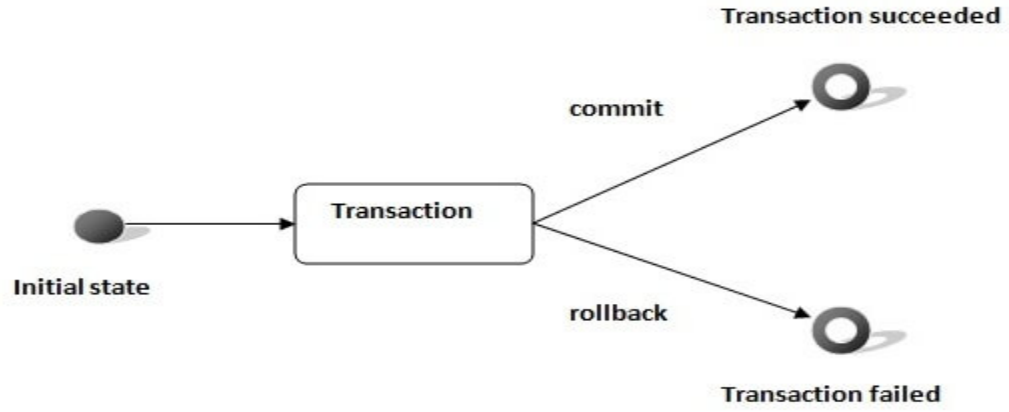
Unit 4

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.

Advantage of Transaction Mangaement

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



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.

How to retrieve contents of a table using JDBC connection?

```
import java.sql.*;
public class jdbcResultSet {
  public static void main(String[] args) {
    try {
      Class.forName("org.apache.derby.jdbc.ClientDriver");
```

```
    } catch(ClassNotFoundException e) {
      System.out.println("Class not found "+ e);
    }
    try {
      Connection con = DriverManager.getConnection(
        "jdbc:derby://localhost:1527/testDb","username", "password");

      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT * FROM employee");
      System.out.println("id  name    job");

      while (rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("name");
        String job = rs.getString("job");
        System.out.println(id+"   "+name+"    "+job);
      }
    } catch(SQLException e) {
      System.out.println("SQL exception occured" + e);
    }
  }
}
```

Result:

```
id  name   job
1   alok   trainee
2   ravi   trainee
```

ASSIGNEMENT

Q1. Discuss EJB. Explain EJB architecture. What are its various types? Write steps for creating Java Beans.

Q2. Explain Java Beans. Why they are used? Discuss setter and getter method. Differentiate between Java Bean & Enterprise Java Bean.

Q3. Explain JDBC application architecture. List various types of JDBC Drivers. Discuss the steps to connect database with the web application using JDBC. Write a program to demonstrate how to retrieve the data from a table using JDBC API.

Q4. Write a JDBC program for insert and display the record of employees using prepared statement.

Q5. What are jar files? Explain its advantage? Write the command for creating jar file?