① What is SWING ? define features of swing class over the AWT.

SWING :-

+ Swing is the part of JAVA Foundation classes (JFC) used to create graphical user interfaces (GUIs) in JAVA application.

+ JAVA swing provides platform-independent and lightweight components.

+ It is built on top of AWT (Abstract window toolkit) offering a more flexible and powerful way to create GUI application.

Feature of SWING-

[1] Lightweight Component-
Unlike AWT component, which are heavyweight (dependent on native OS components), swing component are lightweighted meaning they are rendered by JAVA rather than the OS.

[2] More component-
Offers advanced UI elements like tables, lists and trees.

[3] Customizable-
You can change the look and feel of the components easily.

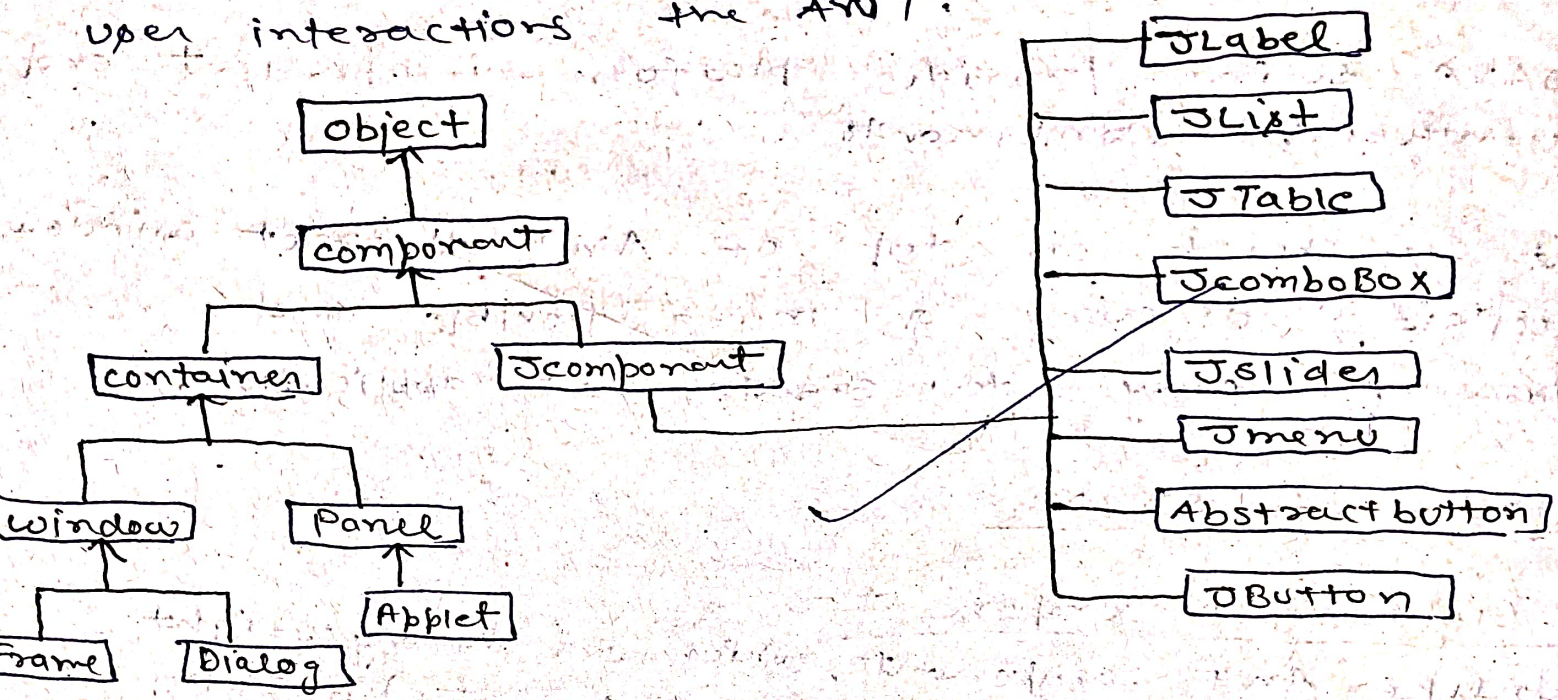
[4] Plateform Independent-
swing provide a consistent user interface across all plateform since it is written entirely in JAVA.

[5] MVC Architecture -

Swing follows the model - view - controller (MVC) design pattern which seperates the data (model), the UI (view) and the behaviour (controller).

[6] Event Handling -

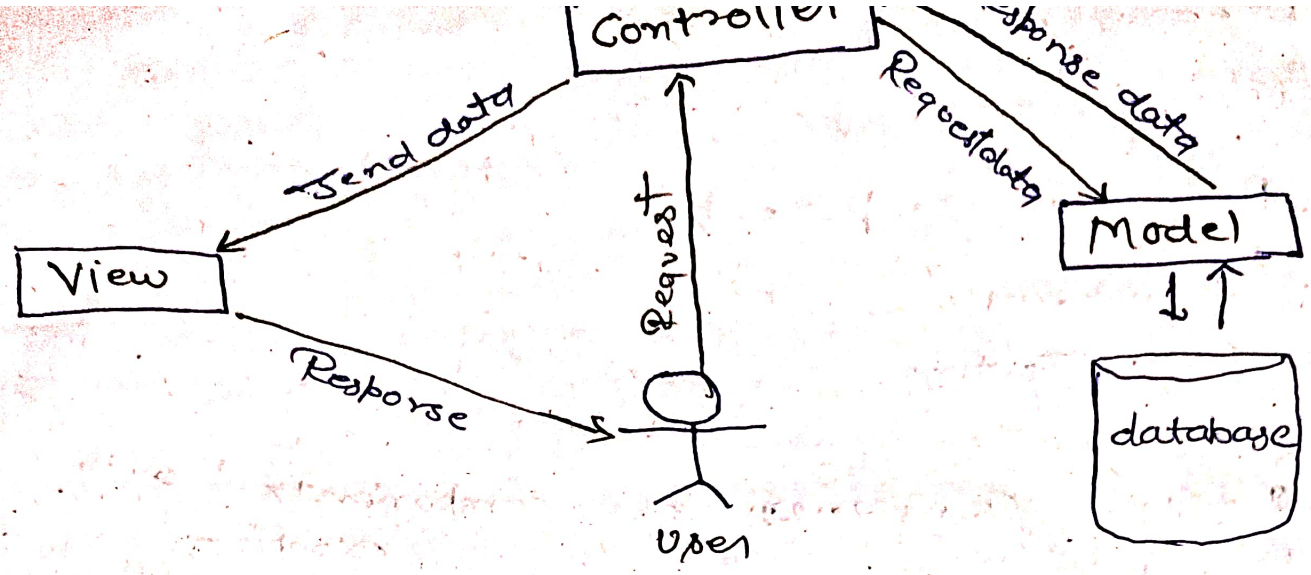
Swing's event handling is more organized and gives developers better control over user interactions than AWT.



Q-2 Explain MVC Architecture with diagram and write advantage of MVC.

MVC Architecture -

* MVC (model - view - controller) is a design pattern used to separate the logic of an application into three interconnected components - Model, view and controller.
* This seperation allow for modular development and easier maintenance of code.



Componants of MVC- [1] Model

- This component represents the data and the business logic of the application.
- It directly manages the data, logic and rules of the application.

[2] View -

- This componant handles the user interface and presentation.
- This is what the user sees and interacts with (the UI).

[3] Controller -

- The controller is the intermediary between the model and the view.
- It take input from the user through the view, tells the model to change data if needed and then updates the view with the new data.

How MVC works -

- User interacts with the view (UI).
- controller recieves this input from the view and processes it.
- Model changes and sends the updated data back.
- View refreshes to show the updated data.

Advantages -

1. Organized code
2. Easy maintenance
3. Reusable components
4. Scalability
5. Better Testing.

Q-3 Define containers and components with example -

1. Containers :-

* Containers are components that hold and organize other components (like button, text fields) on the screen.
* They proride a layout to arrange component within them.

Common container classes - JFrame, JPanel, JDialog and JApplet.

```
import Jarax.swing.*;
public class MyContainer {
Public static void main (string args[])
{
JFrame frame = new JFrame ("container Example");
frame.setsize (300, 200);
frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE
JPanel Panel = new JPanel();
frame.add (Panel);
Panel1.add (new JButton ("Button1");
frame.setVisible (true);
```

componants -

* componants are individual UI elements with a container, like button, labels, text field etc.
* They are responsible for user interactions and displaying information.

common coponants - JButton, JLabel, JTextField.

Example -

```
JButton button = new JButton ("clickme");
JLabel label = new label ("Enter name");
JTextField textfield = new JTextField (10);
```

Q-4. Write the steps of JDBC connectivity model with example.

The JDBC (Java Database connectivity) model allows JAVA applications to interact with databases.

Steps of JDBC connectivity -

1. Load the Driver -
   Load the database driver class.

   ⇒ Class.forName ("com.mysql.cj.jdbc.Driver");

2. Establish connection -
   Connect to the database using DriverManager.

   Connection conn = DriverManager.getConnection ("jdbc:mysql://localhost:3306/mydb", "username", "password");

[3] Create Statement -
Create a statement or prepared statement object to execute queries.

Statement stmt = conn.createStatement();

Use the statement executeQuery() method for select queries or executeUpdate() for INSERT, UPDATE and DELETE operation.

```
ResultSet rs = stmt.executeQuery("select * FROM users");
```

5. Process the ResultSet – Iterate through the ResultSet to fetch data returned by the query.

```
while (rs.next())
{
System.out.println("ID:" + rs.getInt("id") +
                   "Name:" + rs.getString("name");
}
```

6. Close the connection – Close the ResultSet, Statement and connection to free up resources.

```
rs.close();
Stmt.close();
Conn.close();
```

Example – Connect to a MYSQL database and retriere records from a user table.

```
import java.sql.connection.
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.statement;

Public class JDBCExample{
    public static void main(String[] args)
    {
        connection conn = null;
        statement stmt = null;
        ResultSet rs = null;
```

```
try {      // Load
Class.forName("com.mysql.cj.jdbc.Driver");

    // step-2 Establish connection
conn = DriverManager.getConnection("jdbc:
    mysql://localhost:3306/mydatabase", "root",
                                    "Password");

    // step-3 Create a statement
stmt = conn.createstatement();

    // step-4 Execute Query.
String sql = "select * from users";
rs. stmt.executeQuery(sql);
// step-5 Process the Resultset.
while (rs.next())
    {
    int id = rs.getInt("Id");
    String name = rs.getString("name");
System.out.println("ID: " + id + ", Name: " + name);

    }
} catch (Exception e)
    {
    e.PrintStackTrace();
    }
finally {
    try {
        if (rs != null)
            {
            rs.close();
        if (stmt != null)
            {
            stmt.close();
        if (conn != null)
            {
            conn.close();
    }
```

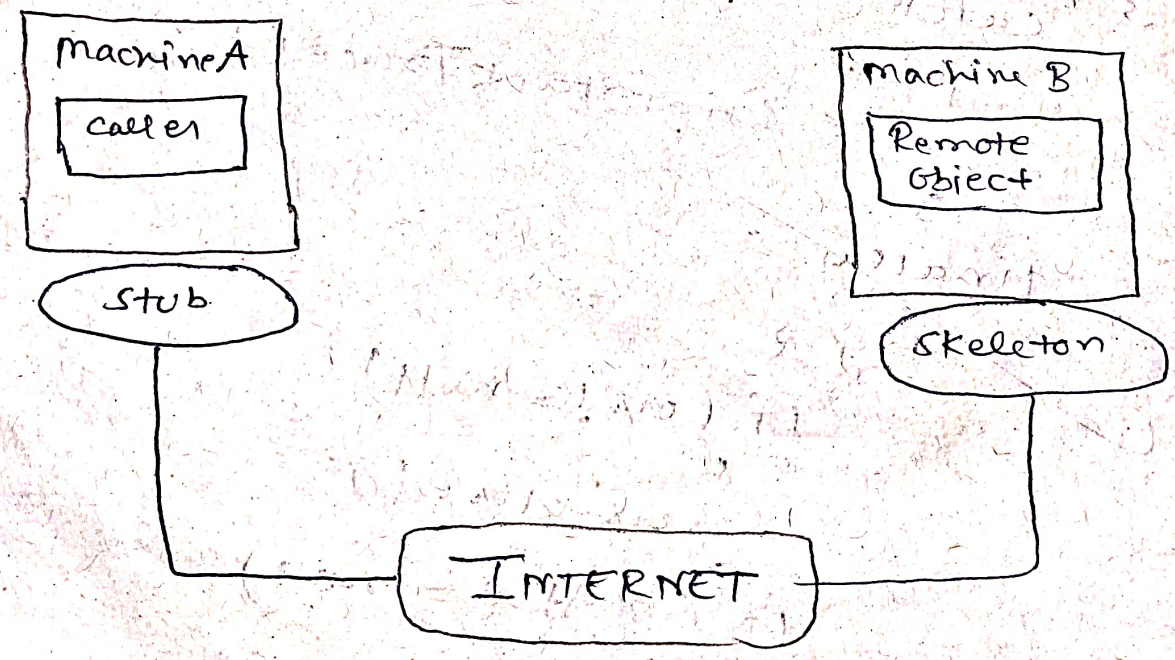
```
        Catch (Exception e)
            {
              e.printStackTrace();
            }
        }
    }
}
```

Ques-s What is RMI ? How to connect remote database ?

RMI stands for Remote method Invocation.

+ RMI allow one JAVA Program on a computer (client) to call methods (like functions) in another JAVA Program on a different computer (server) over the internet.

+ Through RMI, an object running in a JVM present on a computer (client side) can invoke methods on an object present in another JVM (server side).



→ RMI is an API that provide a mechanism to create distributed application in JAVA.

→ The RMI allows an object to invoke methods on an object running in another JVM.

Example- Connection of remote database -

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

public class RemoteDatabase {
    public static void main (String [] args)
    {
        String JdbcUrl = "jdbc:mysql://192.168.1.100:
                          3308/mydatabase";
        String username = "smir-9918";
        String Password = "sriram@123";

        Connection conn = null;
        try {
            conn = DriverManager.getConnection (jdbcurl, username,
                                                password);
            System.out.println ("connection established
                                 successfully");
        }
        catch (Exception e)
        {
            e.PrintStackTrace ();
            System.out.println ("connection failed");
        }
        finally {
            if (conn != null && !conn.isClosed())
            {
                conn.close ();
                System.out.println ("connection closed");
            }
```

```
        catch (SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}
```

Ques-06 What is JDBC/ODBC Bridge.

JDBC Driver is a software component that enables java application to interact with the database.

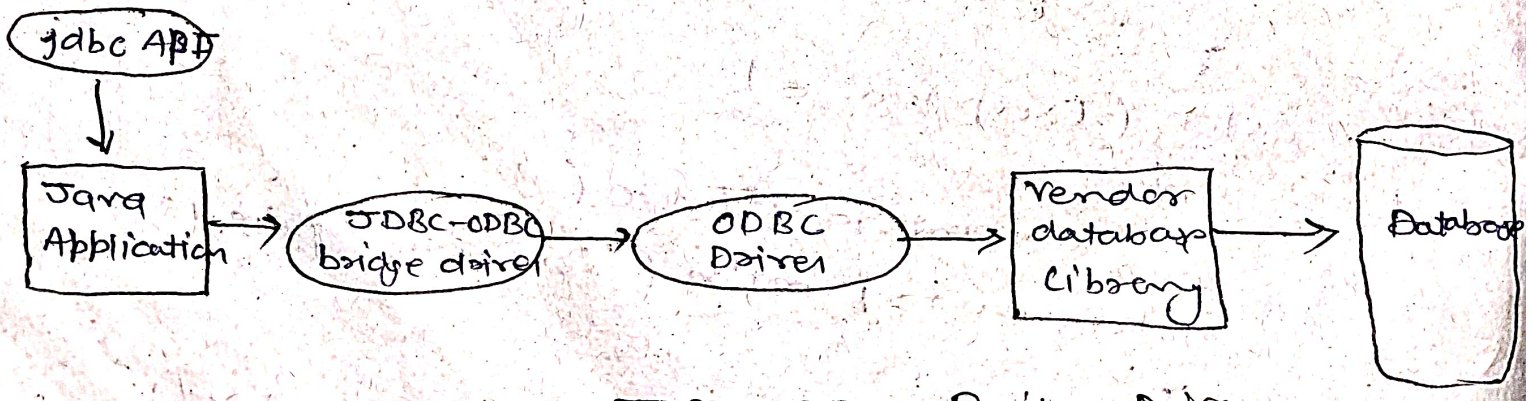
There are four types of JDBC Drivers —

1. JDBC-ODBC bridge driver
2. Native-API Driver (partially Java driver)
3. Network Protocal driver (fully Java driver)
4. Thin driver (fully JAVA driver)

JDBC-ODBC Bridge —

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database.

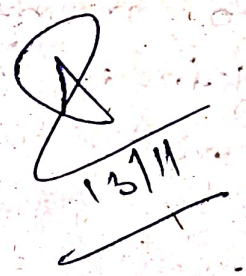
The JDBC-ODBC bridge driver converts JDBC method call into the ODBC function calls.



JDBC-ODBC Bridge Driver.

Advantages - Easy to use.
Can be easily connected to any database.

DisAdvantage -
* Performance reduced because JDBC method call is converted into the ODBC function calls.
* The ODBC driver need to be installed on the client machine.



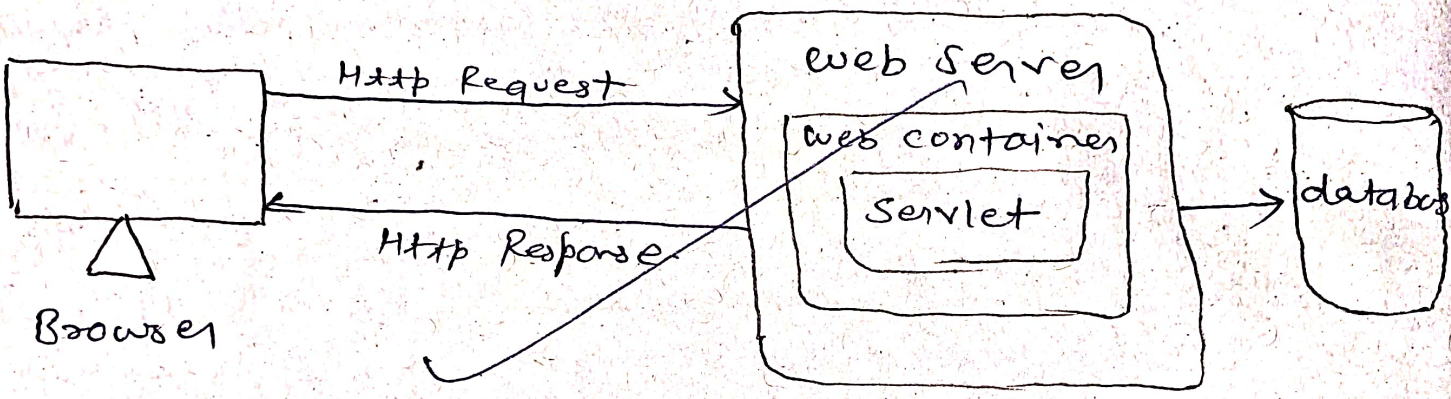
13/11

Q- What is Servlet ? Explain life cycle of servlet.

JAVA Servlets -

Java servlets are software components that run on a web server and be responsible to recieve the request from the web server, process the request and respond back to the server.

Servlets extends the capabilities of a server as they can respond to many type of request, they act like web containers for hosting web applications on web servers.



+ The User sends HTTP requests to the web server.
+ The server has the web container containing servlet, which gathers data from the database and create a response.
+ The response created by servlet is sent through HTTP response to the client browser.

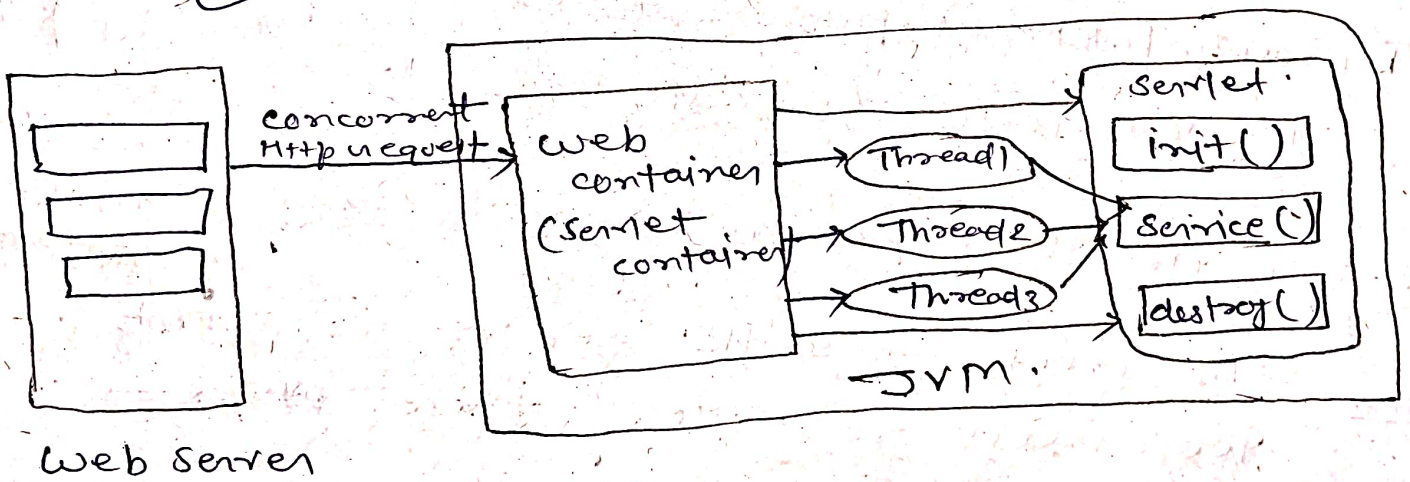
Life cycle of Servlet -

Servlet life cycle consists method that covers entire process from its creation till the destruction. following are the steps —

- **init() -** init() is called only once. It is invoked when servlet is created. So it is used for one - time initialisation.

- **service() -** service() method is the main method that perform the actual task. The web container (servlet container) calls the service() method to handle request coming from the client.

- **destroy() -** destroy() method is invoked only once at the end of the life cycle of a servlet. This method gives the servlet a chance to close database connection and perform other such cleanup tasks.



Q-2 what do you ... functionality in servlets.

Server side include- (SSI)

Server side include means adding a piece of content (like a header or footer) from one part of the website into another part, all done on the server before showing it to the user.

Why it is useful ?

(i) Reuses Common Parts- Instead of writing the same header or footer on every page, you create it once and include it wherever needed.

(ii) Easier to Update- If you change the header in one place, it updates everywhere it's included.

Server Side Include functionality can be implemented using the RequestDispatcher's include() method.

This method allows you to include other servlets or JSP files into the response.

RequestDispatcher dispatcher = request.getRequestDispatcher("/header.jsp");
dispatcher.include(request, response);

Q-3 write the steps to create a servlet with example.

Ans- To create a simple servlet in JAVA, you need to follow these steps —

[1] Set-up environment-

* Make sure you have a JAVA development kit (JDK) installed.
* Download and install a servlet container like

[2] Create a Dynamic web Project-

* Open your IDE, and create a new Dynamic web Project.
* Specify the Project name (e.g. HelloWorldServletApp)
* Set the Target Runtime to Apache Tomcat

[3] Create servlet class-

* Create a new JAVA class in the Src folder.
* Import required packages and extend HttpServer.

[4] Define the servlet-

* Use @WebServlet("/hello") to specify the URL pattern.
* Override doGet();

```
Public class HelloWorldServlet extends HttpServlet
{
Protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
        throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1> Hello, world! </h1>");
    }
}
```

[5] Deploy and Run-

* Deploy the project to tomcat using Run on Server.
* Access it at: http://localhost:8080/YourProjectName/hello.

Q-1- what is Java Beans ? what are the steps to create it and view it in the Bean Box ?

JAVA Bean -

* JAVA Beans are classes that encapsulate many Object into a single object (the bean)
* It is a Java class that should follow the following conventions -
* Must implement Serializable.
* It should have a public no-arg constructor.
* All property in Java bean must be private with public getters and setter methods.

Eaon - Public class TestBean

```
{
Private string name;
Public void setName(string name)
{
this.name = name;
}
Public string getName()
{
return name;
}
}
```

Steps to create and view the BeanBox -

To create a javabeans below steps are to be followed -

1- Implements java.io.Serializable ; interface.
2- Declare private variables.
3- Declare no-argument constructor or default

4- declare getter and

Implementation of Java Bean -

```
Public class Employee implements java.io.Serializable
{
private int id;
Private string name;
Public Employee ()
    {
    }
public void setId (int id)
    {
    this.id = id;
    }
public int getId ()
    {
    return id;
    }
Public void setName (string name)
    {
    this.name = name;
    }
public string getName ()
    {
    return name;
    }
}
```

Program to access JavaBean -

```
Public class Employee1
    {
public static void main (string args[])
    {
Employee s = new Employee ();
s.setName ("shiv");
System.out.println (s.getName ());
} }
```

JDBC connection –

```
Public class Demo {
    Public static void main(String[] args) throws Exception
    {
String sql = "Select Name from product where id=8";
String url = "jdbc:postgresql://localhost:5432/telusko";
String username = "Postgres";
String password = "0000";
Connection con = DriverManager.getConnection(url, username, password);
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(sql);
rs.next();
String name = rs.getString(1);
System.out.println(name);
con.close();
```

STUB and Skeleton –

 # STUB – * The stub is an object, act as a gateway for the client side.

* All the outgoing requests are routed through it.
* It resides at the client side and represent the remote object.

 # Skeleton – * The skeleton is an object, act as a gateway for the server side object.

* All the incoming requests are routed through it.
* When the skeleton receives the incoming request, it does the following tasks –

* It reads the parameter for the remote method.
* It invokes the method on the actual remote object
* It writes and transmits the result to the caller.