JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

JECRC Campus, Shri Ram Ki Nangal, Via-Vatika, Jaipur

LAB MANUAL

Lab Name	: ADVANCE J AVA LAB
Lab Code	: 5CS4-24
Branch	: COMPUTER SCIENCE & ENGINEERING
Year	: 2021-2022

JAIPUR ENGINEERING COLLEGE
AND RESEARCH CENTRE

Department of Computer Science Engineering
Jaipur Engineering College and Research Centre, Jaipur
(Affiliated to RTU, Kota)

INDEX

Experiment 10	Write a JSP program to display the grade of a student by accepting the marks of five subjects	
Content Beyond Syllabus		
Experiment 12	Case study on Java Beans	
Experiment 13	Case Study On Enterprise Java Bean	

Vision of the Institute

To become a renowned centre of outcome based learning, and work towards academic, professional, cultural and social enrichment of the lives of individuals and communities.

Mission of the Institute

M1: Focus on evaluation of learning outcomes and motivate students to inculcate research aptitude by project based learning.

M2: Identify, based on informed perception of Indian, regional and global needs, areas of focus and provide platform to gain knowledge and solutions.

M3: Offer opportunities for interaction between academia and industry.

M4: Develop human potential to its fullest extent so that intellectually capable and imaginatively gifted leaders can emerge in a range of professions.

Vision of the Department

To become renowned centre of excellence in Computer Science and Engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

.

Mission of the Department

M1: To impart outcome based education for emerging technologies in the field of Computer Science and Engineering.

M2: To provide opportunities for interaction between academia and industry.

M3: To provide platform for lifelong learning by accepting the change in technologies.

M4: To develop aptitude of fulfilling social responsibilities.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: To provide students with the fundamentals of engineering sciences with more emphasis in Computer Science and Engineering by way of analyzing and exploiting engineering challenges.

PEO2: To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real life problems in Computer Science and Engineering

PEO3: To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues for Computer Science and Engineering.

PEO4: To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career in Computer Science and Engineering.

PEO5: To prepare students to excel in Industry and Higher education by Educating Students along with high moral values and knowledge in Computer Science and Engineering.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1. Ability to Interpret and analyze network specific and cyber security issues in real world environment.

PSO2. Ability to design and develop mobile and web based applications under realistic constraints.

PROGRAM OUTCOMES (POs)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

RTU Syllabus with List of Experiments

5CS4-24: ADVANCE JAVA LAB

Class: 5th Sem. B. Tech. 3rd year	**Evaluation**
Branch: CS **Credits: 1** **Schedule per week: 2Hrs (Practical)**	**Examination Time=TWO (2) Hours** **Maximum Marks = 50** **[Internal Assessment/Sessional(30) & End-term Exam(20)]**

S. No.	**NAME OF PROGRAM**
1	Introduction To Swing, MVC Architecture, Applets, Applications and Pluggable Look and Feel, Basic swing components : Text Fields, Buttons, Toggle Buttons, Checkboxes, and Radio Buttons.
2	Java database Programming, java.sql Package, JDBC driver, Network Programming With java.net Package, Client and Server Programs, Content And Protocol Handlers
3	RMI architecture, RMI registry, Writing distributed application with RMI, Naming services, Naming And Directory Services, Overview of JNDI, Object serialization and Internationalization
4	J2EE architecture, Enterprise application concepts, n-tier application concepts, J2EE platform, HTTP protocol, web application, Web containers and Application servers
5	Server side programming with Java Servlet, HTTP and Servlet, Servlet API, life cycle, configuration and context, Request and Response objects, Session handling and event handling, Introduction to filters with writing simple filter application
6	Server side programming with Java Servlet, HTTP and Servlet, Servlet API, life cycle, configuration and context, Request and Response objects, Session handling and event handling, Introduction to filters with writing simple filter application
Perform the following experiment using virtual lab http://vlabs.iitb.ac.in/vlabs-dev/labs/java-iitd/experiments/java-intro-iitd/simulation.html	
7	To simulate any core java program
8	Case study on Introduction to Java Programming Language

Perform the following experiment using virtual lab http://vlabs.iitb.ac.in/vlabs-dev/vlab_bootcamp/bootcamp/bots_with_dots/labs/exp1/index.html	
9	To demonstrate how to perform database operation with mysql using advanced java technology.
10	CRUD Operations Using Java Database Connectivity

Course Outcomes

CO 1: Develop an in depth understanding of swings programming by writing user interfaceprograms in swings.

CO 2: Develop client side application involving JDBC, RMI and Socket Programming in Java.

CO 3: Develop server side applications involving J2EE, Servlets and JSP.

Mapping of Experiments with COs & BT Level

Experiment 1	Write an AWT program to create check boxes for different courses of a university such that the courses selected will be displayed..	CO1	BT6
Experiment 2	Write an applet to create list of department of a college if one of the department is clicked then it will be displayed on the text field	CO1	BT6
Experiment 3	Write a swing program to create components like text fields, buttons, toggle buttons, checkboxes, and radio buttons	CO1	BT6
Experiment 4	Develop a JDBC application to perform the database driven operation like insert, delete, update and selection using statement, prepared statement and result set	CO1	BT5
Experiment 5	Write a socket program in java to implement client server communication using socket and server socket class	CO2	BT6
Experiment 6	Develop RMI application in which client sends operator and operands for computation and server sends back the result. It behaves just like a distributed calculator app	C02	BT5

Experiment 7	Write a servlet program to display “Hello World” on browser.	CO3	BT6
Experiment 8	Write a servlet program for user registration &amp; then control will be transfer into second page.	CO3	BT6
Experiment 9	Write a JSP program for user login form with static and dynamic database	CO3	BT6
Experiment 10	Write a JSP program to display the grade of a student by accepting the marks of five subjects	CO3	BT6

* BT - Bloom's Taxonomy

Mapping of Course Outcomes & POs/PSOs

	Engineering Knowledge	Problem analysis	Design/Development of Solution	Conduct Invest. of complex problems	Modern Tool Usage	The engineer and society	Environment and Sustainability	Ethics	Individual and Team Work	Communication	Project Management and Finance	Life-long Learning	Interpret and analyze network specific and cyber security issues in real world	design and develop mobile and web based applications under realistic
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO 1**	**PSO 2**
CO-1	3	3	1	3	2	1	1	2	2	3	1	3	2	2
CO-2	3	3	3	2	2	2	1	1	1	2	1	3	2	2
CO-3	3	3	3	1	3	1	2	2	1	1	2	3	3	2

INTRODUCTION ABOUT LABORATORY& APPLICATIONS

If technology touches life, chances are, so does Java technology. Invented by Sun Microsystems in 1995, Java technology has become the essential ingredient of the digital experience for hundreds of millions of people in all walks of life, all over the planet.

Java software powers the onboard computers in toys, cars, planes, rockets, and even the NASA Mars Rover. It brings interactivity to the Internet, real- time graphics to television, instant imaging to cameras, and multi-player games to mobile phones and desktop PCs. It connects the largest enterprises and smallest businesses to their employees, customers, and data. And it secures the vast majority of electronic transactions in retail, finance, government, science, and medicine. In short, Java technology goes everywhere you go.

It's no wonder that Java technology has become the most powerful force in software and the most prevalent software in technology. In fact, it is the software of choice for more software engineers than any other brand of software.
Java is the most efficient, fast, secure, animated, compatible, and reliable software.

Java Technology Facts

- Java technology was invented by Sun Microsystems and celebrated its 10- year Birthday in March 2005
- The Java platform specifications and compatibility standards are controlled by the independent industry members of the Java Community Process
- Java software runs on more types of consumer and embedded devices, smart cards, ATMs, thin clients, PCs, servers, and mainframes than any other software
- Java applications are more resistant to viruses than any other popular programming language
- Today's five million Java developers are the largest community of software developers

After completing Advanced Java Lab students will be able to develop program in Java for following applications
Java Data Base Connectivity: A Data Base can be accessed from program

Servlets: For development of web basd components

Java Beans: Java Beans are, quite simply, reusable controls written in Java, for Java application development. They let you visually assemble components and dynamically change properties
Java Server Pages: to dynamically generate HTML, XML or other types of documents in response to a Web client request.
Remote Method Invocation: RMI provides the mechanism by which the server and the client communicate and pass information back and forth.
As we all know there is great demand of Java in industry so definitely student will benefited by advanced java Lab. It can help them to blossom their future

INSTRUCTIONS SHEET

We need your full support and cooperation for smooth functioning of the lab.

DO's

1. **Please switch off the Mobile / Cell phone before entering in the lab.**
2. **Enter in the Lab with complete preparation for programming.**
3. **Check whether all peripheral are available at your desktop before proceeding for the program.**
4. **Intimate the lab in-charge whenever you are incompatible in using the system or in case software get corrupted/infected by virus.**
5. **Arrange all the devices / peripheral and seats before leaving the lab.**
6. **Properly shutdown the system before leaving the lab.**
7. **Keep the bag outside from the lab / put bag in the given racks.**
8. **Enter in the lab on schedule time and leave at proper time after permission.**
9. **Maintain the decorum of the lab.**
10. **Utilize lab hours in the corresponding experiment.**
11. **Get your external storage device checked by lab in-charge before using it in the lab.**

Don'ts

1. **No one is allowed to bring storage devices like external hard disk in the lab without permission.**
2. **Don't mishandle the system.**
3. **Don't leave the system on standing for long time (more than 15 min.)**
4. **Don't bring any external material in the lab.**
5. **Don't make noise in the lab.**
6. **Don't bring the mobile in the lab. If extremely necessary then keep ringers off.**
7. **Don't enter in the lab with out permission of lab in-charge.**
8. **Don't litter in the lab.**
9. **Don't delete or make any modification in system files.**

10. **Don’t carry any lab equipments outside the lab.**
11. **Don’t eat and drink inside the lab.**
12. **Avoid stepping on electric wires or any other computer cables.**

BEFORE ENTERING IN THE LAB

1. All the students are supposed to prepare the theory regarding the next experiment.
2. Students are supposed to bring the practical file and the lab copy.
3. Previous practical should be written in the practical file.
4. Any student not following these instructions will be denied entry in the lab.

WHILE WORKING IN THE LAB

1. Adhere to experimental schedule as instructed by the lab in-charge.
2. Get the previously executed experiment signed by the instructor.
3. Get the output of the current experiment checked by the instructor in the lab copy.
4. Take responsibility of valuable accessories.
5. If anyone is caught carrying any equipment of the lab outside without permission, they will face strict disciplinary action.

Program No. 1

AIM: Write an AWT program to create check boxes for different courses of a university such that the courses selected will be displayed..

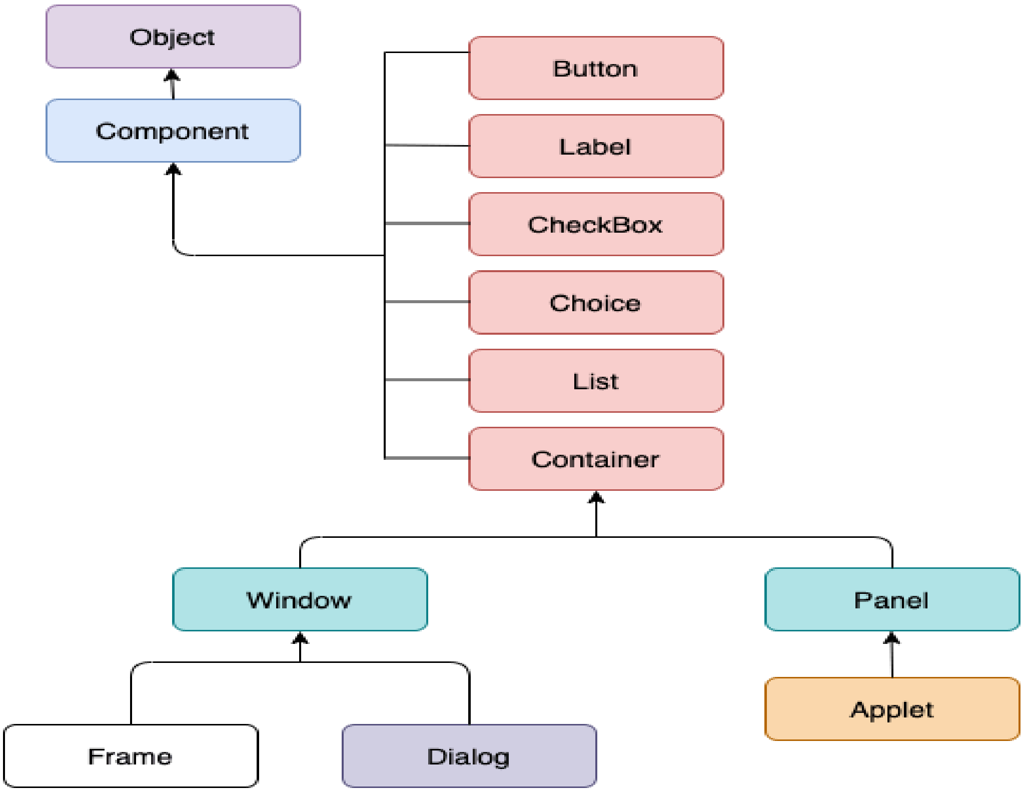
THEORY

AWT Program in Java

AWT stands for Abstract window toolkit is an Application programming interface (API) for creating Graphical User Interface (GUI) in Java. It allows Java programmers to develop window-based applications.

AWT provides various components like button, label, checkbox, etc. used as objects inside a Java Program. AWT components use the resources of the operating system, i.e., they are platform-dependent, which means, component's view can be changed according to the view of the operating system. The classes for AWT are provided by the Java.awt package for various AWT components.

The following image represents the hierarchy for Java AWT.



Component Class

The component class stands at the top of the AWT hierarchy, is an abstract class that contains all the properties of the component visible on the screen. The Component object contains information about the currently selected foreground and background color. It also has information about the currently selected text color.

Container

The container is a component that contains other components like button, textfield, label, etc. However, it is a subclass of the Component class.

Panel

The panel can be defined as a container that can be used to hold other components. However, it doesn't contain the title bar, menu bar, or border.

Window

A window can be defined as a container that doesn't contain any border or menu bar. It creates a top-level view. However, we must have a frame, dialog, or another window for creating a window.

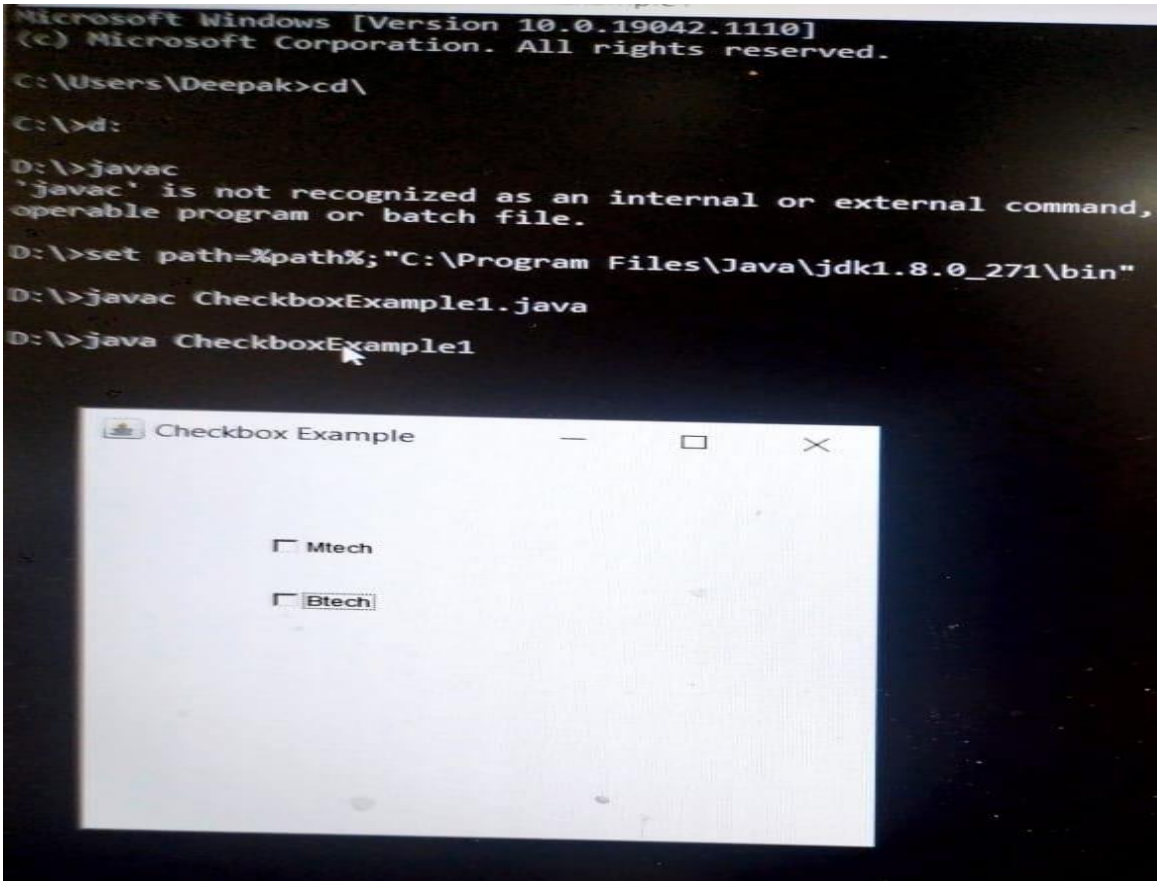
Frame

The frame is a subclass of Window. It can be defined as a container with components like button, textfield, label, etc. In other words, AWT applications are mostly created using frame container.

PROGRAM CODE

```
import java.awt.*;
import java.awt.event.*;
public class CheckboxExample1
{

    CheckboxExample1()
 {
      Frame f = new Frame("Checkbox Example");
       Checkbox checkbox1 = new Checkbox("Mtech");
       checkbox1.setBounds(100, 100,  50, 50);
       Checkbox checkbox2 = new Checkbox("Btech", true);
checkbox2.setBounds(100, 150,  50, 50);
        f.add(checkbox1);
        f.add(checkbox2);

        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
     }
public static void main (String args[])
{
    new CheckboxExample1();
}
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Deepak>cd\

C:\>d:

D:\>javac
'javac' is not recognized as an internal or external command,
operable program or batch file.

D:\>set path=%path%;"C:\Program Files\Java\jdk1.8.0_271\bin"

D:\>javac CheckboxExample1.java

D:\>java CheckboxExample1
```

APPLICATIONS:

1.Used in Creating Various Online Forms.
2.Used when there are lists of options and the user may select any numbers of choices.

VIVA VOCE

1. What is Awt?
2. Explain Hierarchy of Awt?
3. What is Windows Application?
4. Define Component of Awt?
5. What is Container?
6. What is the difference between Awt and Application?

Program No . 2

AIM: Write an applet to create list of department of a college if one of the department is clicked then it will be displayed on the text field.

THEORY

Java Applet

Applet is a special type of program that is embedded in the web page to generate the dynamic content. It runs inside the browser and works at client side. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

Advantages of Applet

It works at client side so less response time.

Secured

It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

Life Cycle Methods of Applet:

There are four methods of java.applet.Applet class:-

init − This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.

start − This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.

stop − This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.

destroy − This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.

paint method:

The paint( ) method is called each time an AWT-based applet's output must be redrawn. paint( ) is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, paint( ) is called.

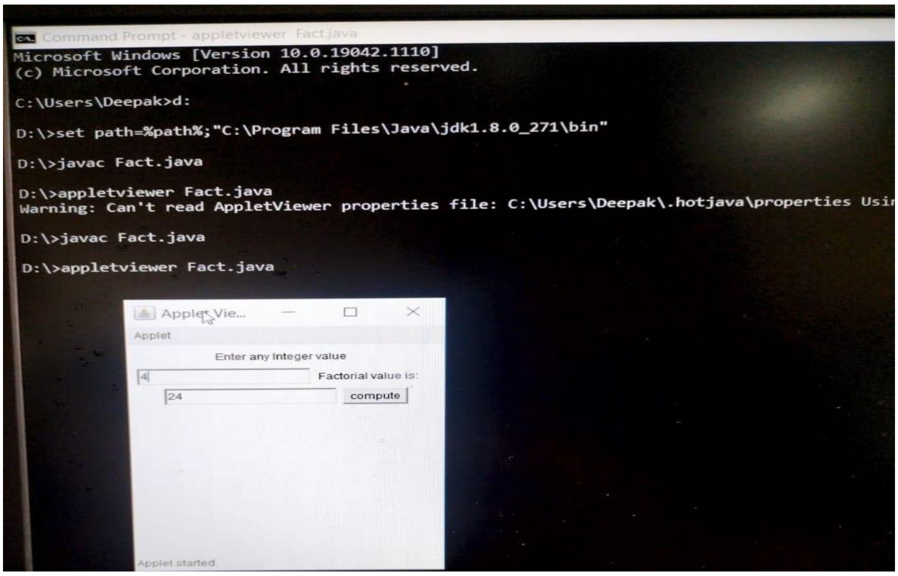
The paint( ) method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

PROGRAM CODE

```
import java.awt.*;
 import java.lang.String;
import java.awt.event.*;
import java.applet.Applet;
```

```
public class Fact extends Applet implements ActionListener
 { String str;
Button b0;
TextField t1,t2;
Label l1;
 public void init()
{ Panel p=new Panel();
 p.setLayout(new GridLayout());
add(new Label("Enter any Integer value"));
add(t1=new TextField(20));
add(new Label("Factorial value is: "));
 add(t2=new TextField(20));
add(b0=new Button("compute"));
 b0.addActionListener(this);
 }
 public void actionPerformed(ActionEvent e)
 { int i,n,f=1;
 n=Integer.parseInt(t1.getText());


 for(i=1;i<=n;i++)
 f=f*i;
t2.setText(String.valueOf(f));
 repaint();

} }
```

OUTPUT



```
Command Prompt - appletviewer  Fact.java
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Deepak>d:

D:\>set path=%path%;"C:\Program Files\Java\jdk1.8.0_271\bin"

D:\>javac Fact.java

D:\>appletviewer Fact.java
Warning: Can't read AppletViewer properties file: C:\Users\Deepak\.hotjava\properties Usi

D:\>javac Fact.java

D:\>appletviewer Fact.java
```

APPLICATIONS:

1.Used for Security Purpose
2.Basically Used in GUI Applications

VIVA QUESTION.

1. Difference Between AWT and APPLET?
2. State Life cycle of Applet?
3. Why Applet does not have main method?
4. Which Header file is Required for graphics in applet?
5. Mention the Difference Between paint() and repaint() method?

Program No. 3

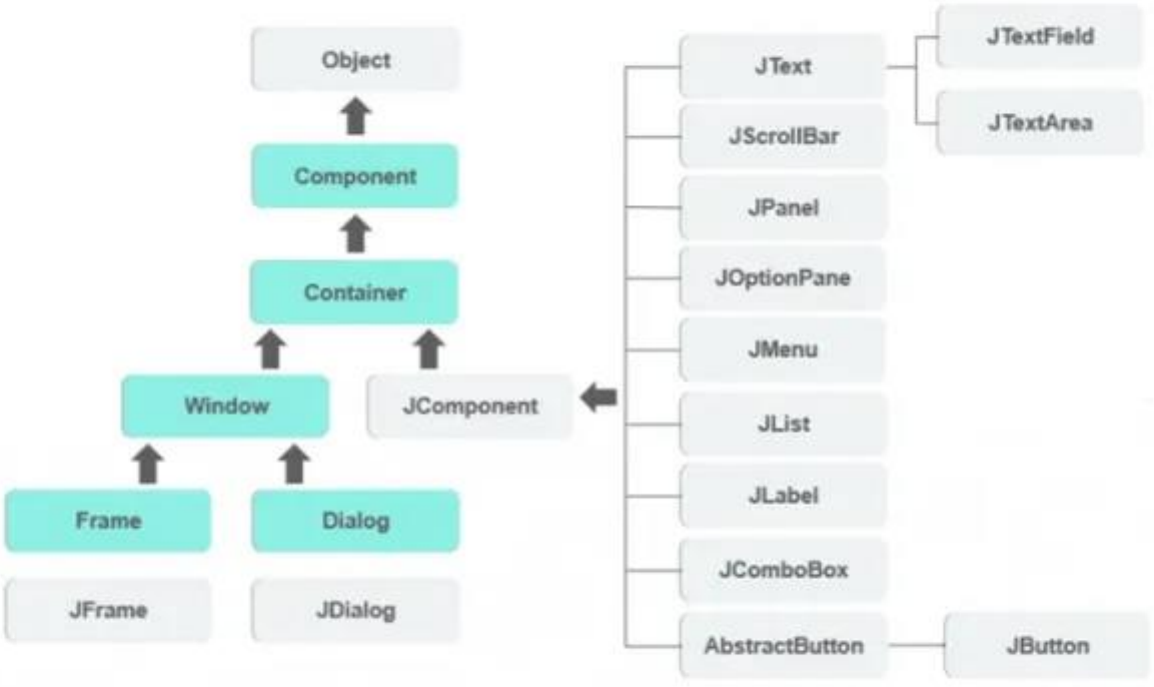
AIM: Write a swing program to create components like text fields, buttons, toggle buttons, checkboxes, and radio buttons.

THEORY

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC(Java Foundation Classes). It is build on top of the AWT API and entirely written in Java. It is platform independent unlike AWT and has lightweight components.

Container Class
Any class which has other components in it is called as a container class. For building GUI applications at least one container class is necessary.
Following are the three types of container classes:
Panel – It is used to organize components on to a window
Frame – A fully functioning window with icons and titles
Dialog – It is like a pop up window but not fully functional like the frame

Hierarchy of Java Swing Classes



Commonly used Methods of Component class

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(intwidth,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false

JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

Commonly used Constructors

Constructor	Description
JJCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied

Commonly used Methods

Methods	Description
AccessibleContextgetAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a string representation of this JCheckBox.

JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz. It should be added in ButtonGroup to select one radio button only.
Commonly used Constructors

Constructor	Description
JRadioButton()	Creates an unselected radio button with no text.
JRadioButton(String s)	Creates an unselected radio button with specified text.
JRadioButton(String s, boolean selected)	Creates a radio button with the specified text and selected status

Commonly used Methods

Methods	Description
void setText(String s)	It is used to set specified text on button.
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.

Methods	Description
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

JToggleButton

JToggleButton is used to create toggle button, it is two-states button to switch on or off
Commonly Used Constructors

Constructor	Description
JToggleButton()	It creates an initially unselected toggle button without setting the text or image.
JToggleButton(Action a)	It creates a toggle button where properties are taken from the Action supplied.
JToggleButton(Icon icon)	It creates an initially unselected toggle button with the specified image but no text.
JToggleButton(Icon icon, boolean selected)	It creates a toggle button with the specified image and selection state, but no text.

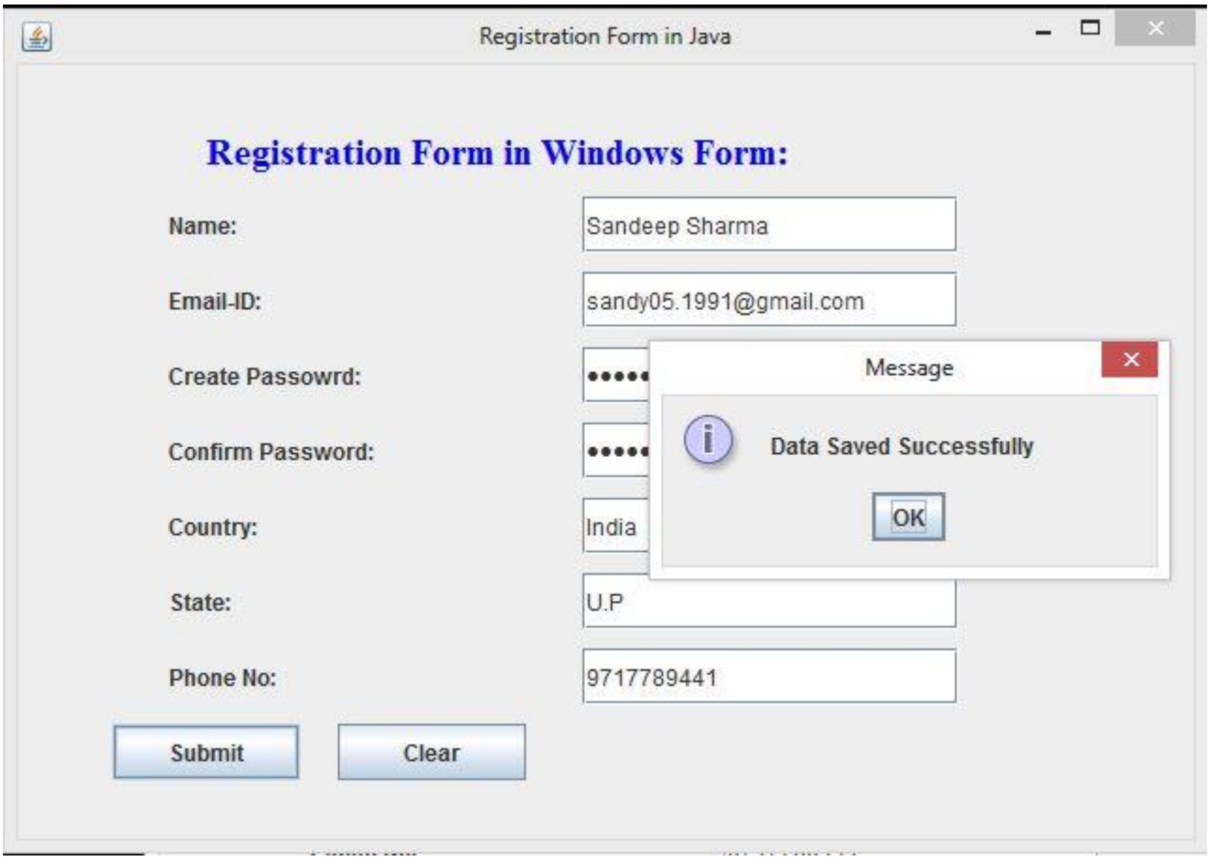
Commonly Used Methods

Modifier and Type	Method	Description
AccessibleContext	getAccessibleContext()	It gets the AccessibleContext associated with this JToggleButton.
String	getUIClassID()	It returns a string that specifies the name of the l&f class that renders this component.
protected String	paramString()	It returns a string representation of this JToggleButton.
void	updateUI()	It resets the UI property to a value from the current look and feel.

PROGRAM CODE

```
importjavax.swing.*;
import java.awt.event.*;
public class ButtonExample

importjava.awt.FlowLayout;
importjava.awt.event.ItemEvent;
importjava.awt.event.ItemListener;
importjavax.swing.JFrame;
importjavax.swing.JToggleButton;



public class Demo
{
public static void main(String[] args)
{
  JFrame f=new JFrame();


final JTextField tf=new JTextField();
   tf.setBounds(50,50, 150,20);
   JButton b=new JButton("Click Here");
   b.setBounds(50,100,95,30);
   b.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
         tf.setText("Welcome to Javatpoint.");
      }

JCheckBox checkBox1 = new JCheckBox("C++");
  checkBox1.setBounds(100,100, 50,50);
  JCheckBox checkBox2 = new JCheckBox("Java");
  checkBox2.setBounds(100,150, 100,50);
  JCheckBox checkBox3 = new JCheckBox("Machine Learning");

 f.add(checkBox1);
      f.add(checkBox2);
      f.add(checkBox3);


JRadioButton r1=new JRadioButton("A) C++");
JRadioButton r2=new JRadioButton("B) Java");
JRadioButton r3=new JRadioButton("C) Machine Learning");
r1.setBounds(75,50,100,30);
r2.setBounds(75,100,100,30);
r3.setBounds(75,150,150,30);

f.add(r1);
f.add(r2);
f.add(r3);


  JButton b=new JButton("Welcome to Goeduhub Technologies");
  b.setBounds(130,100,300,40);
  f.add(b);
  f.setSize(600,500);//600 width and 500 height
  f.setLayout(null);
  f.setVisible(true);//


}
} public class JToggleButtonExample extends JFrame implements ItemListener {
   public static void main(String[] args) {
      newJToggleButtonExample();
   }
   privateJToggleButton button;
   JToggleButtonExample() {
      setTitle("JToggleButton with ItemListener Example");
      setLayout(new FlowLayout());
      setJToggleButton();
      setAction();
      setSize(300, 300);
      setVisible(true);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
   private void setJToggleButton() {
      button = new JToggleButton("ON");
      add(button);
   }
   private void setAction() {
      button.addItemListener(this);
   }
   public void itemStateChanged(ItemEvent eve) {
      if (button.isSelected())
         button.setText("OFF");
      else
         button.setText("ON");
   }
}
```

OUTPUT



Registration Form in Java

Registration Form in Windows Form:

Name: Sandeep Sharma

Email-ID: sandy05.1991@gmail.com

Create Passowrd: •••••

Confirm Password: •••••

Country: India

State: U.P

Phone No: 9717789441

Message

Data Saved Successfully

APPLICATION

1. Swing is heavily used in business specific (vertical)/internal application development.
2. Swing applications are used in most cases where a Java app runs on the desktop

VIVA-VOCE

1. Difference between Awt and Swing?
2. Why swing is considered as low weight component?
3. Define various componets of Swing?
4. Difference between button and toggle button?

Program No. 4

AIM: Develop a JDBC application to perform the database driven operation like insert, delete, update and selection using statement, preparedstatement and result set

THEORY

Network programming

Java Networking is a concept of connecting two or more computing devices together so that we can share resources. Java socket programming provides facility to share data between different computing devices. It is best used for sharing resources and for centralized software management .

Network terminologies :

Names	**Description**
IP address	IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 . It is composed of octets that range from 0 to 255.It is a logical address that can be changed.
Protocol	A protocol is a set of rules basically that is followed for communication. For example:TCP , FTP , Telnet , SMTP ,POP etc.
Port Number	The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.The port number is associated with the IP address for communication between two applications.
MAC Address	MAC (Media Access Control) Address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC.
Socket	A socket is an endpoint between two way communication.
Connection oriented and Connection less	In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP. In connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

Java.Net Package

The java.net package provides many classes to deal with networking applications in Java. Some of classes are given below :

DatagramPacket
DatagramSocket
DatagramSocketImpl
InterfaceAddress
JarURLConnection
MulticastSocket
InetSocketAddress
InetAddress

Protocol and Content handlers : Handlers are classes that extend the capabilities of the standard URL class. A protocol handler provides a reference to a java.io.InputStream object (and a java.io.OutputStream object, where appropriate) that retrieves the content of a URL. Content handlers take an InputStream for a given MIME type and convert it into a Java object of the appropriate type.

JDBC Drivers

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

JDBC-ODBC Bridge Driver: The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.

Advantages	**Disadvantages**
Type-1 Driver is database independent driver. it is very easy to use. Not require to install this driver separately(By default in windows)	It is the slowest driver. Type-1 driver internally depends upon ODBC driver so ODBC driver concept application only on window machine i.e. platform dependent driver.

Native Driver : The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

Advantages	Disadvantages
Good performance as compared to the type-1 driver. No ODBC Driver require. Type-2 Drivers are operating system specific and compiled.	It is a database dependent driver. It is a platform-dependent driver. Only Oracle provides type-2 Driver

Network Protocol Driver : The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

Advantages	Disadvantages
This driver does not directly communicate with the database. So it is database independent driver. For any database, this driver is the same. It is fully written in Java. So it is platform independent driver. No client-side libraries are required.	Network support is required on the client machine. When we change the database, we need to change the middle-wear code. Maintenance of Network Protocol driver becomes costly.

Thin Driver : The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

Advantages	Disadvantages
Better performance than all other drivers. Platform independent Driver. No software is required at the client side or server side.	Database dependent driver, because it is directly communicating with the database directly.

java.sql package

The java.sql package contains the entire JDBC API that sends SQL (Structured Query Language) statements to relational databases and retrieves the results of executing those SQL statements.contains classes and interfaces for JDBC API. A list of popular interfaces of JDBC API are given below:

Driver interface Connection interface	ResultSet interface ResultSetMetaData interface

Statement interface PreparedStatement interface CallableStatement interface	DatabaseMetaData interface RowSet interface

In this, we will learn how to perform basic database operations like creating a table, inserting the records into a table, updating, deleting and retrieving the records from a table using **JDBC API**. In all examples the following five steps have been followed. In these examples, we have used **Oracle 10g** database. You can use any database as your wish. But the steps to interact with the database remains the same. See this post for more on how to connect to any database using JDBC API.

Step 1 : Registering The Driver Class
First step in establishing the connection with any database is registering the JDBC driver class of that database with DriverManager. As we are using Oracle database in our examples, we register '**oracle.jdbc.driver.OracleDriver**' class, which is the JDBC driver class of the Oracle, with the DriverManager. As this step has to be performed only once for the whole execution, it is better to keep this step in Static Initialization Block. Don't forget to update your classpath with JDBC driver of the Oracle database. Otherwise you will get ClassNotFoundException at run time.

Step 2 : Creating The Connection Object
In the second step, we create java.sql.Connection object using **DriverManager.getConnection()** method by passing URL(jdbc:oracle:thin:@localhost:1521:XE), username and password of the database.

Step 3 : Creating The Statement Object
In the third step, we create java.sql.Statement object using **con.createStatement()** method where **'con'** is the reference to Connection object created in the second step.

Step 4 : Executing The Queries
In the fourth step, we send the queries to the database. While sending we use following methods of Satement object depending upon the type of queries we are sending to the database.

ResultSet executeQuery(String sql) throws SQLException : This method is generally used for SQL query statements which retrieve some data from the database. For example **SELECT** statement. This method returns **java.sql.ResultSet** object which contains the results returned by the SELECT query.

int executeUpdate(String sql) throws SQLException : This method is generally used for SQL statements which update the database. For example **INSERT**, **UPDATE** and **DELETE**. This method is also used for SQL statements which return nothing. For example **CREATE** and **ALTER** statements. This method returns an int value that represents the number of rows affected by the query. This value will be 0 for the statements which return nothing.

boolean execute(String sql) throws SQLException : This method can be used to execute any kind of SQL statements. If you don't know which method to use for your SQL query, then this method is the best option. This method returns a boolean value. **TRUE** indicates that query returned ResultSet object and **FALSE** indicates that query returned an int value.

Step 5 : Closing The DB Resources

In the last step, we close all the DB resources – Connection, Statement and ResultSet objects.

JDBC – SQL CREATE Table Example

```
importjava.sql.*;
publicclassCreateTableExample
{
   static
   {
      //STEP 1 : Registering The Driver Class

      try
      {
         Class.forName("oracle.jdbc.driver.OracleDriver");
      }
      catch(ClassNotFoundException e)
      {
         System.out.println("Unable To Load The Driver class");
      }
   }

   publicstaticvoidmain(String[] args)
   {
      Connection con = null;

      Statement stmt = null;

      try
      {
         //Database Credentials

         String URL = "jdbc:oracle:thin:@localhost:1521:XE";

         String username = "username";

         String password = "password";

         //STEP 2 : Creating The Connection Object

         con = DriverManager.getConnection(URL, username, password);

         //STEP 3 : Creating The Statement Object

         stmt = con.createStatement();
         //Constructing The SQL Query

         String sql = "CREATE TABLE EMPLOYEE("+
               "ID NUMBER NOT NULL, "+
               "FIRST_NAME VARCHAR2(200), "+
               "LAST_NAME VARCHAR2(200), "+
               "DISIGNATION VARCHAR2(200))";

         //Step 4 : Executing The Query

         //We are using executeUpdate() method as we are executing CREATE statement

         inti = stmt.executeUpdate(sql);

         if(i == 0)
         {
            System.out.println("Table is created");
         }
         else
         {
            System.out.println("Table is not created");
         }
      }
      catch(SQLException e)
      {
         e.printStackTrace();
      }
      finally
      {
         //STEP 5 : Closing The DB Resources

         //Closing the Statement object
         try
         {
            if(stmt!=null)
            {
               stmt.close();
               stmt=null;
            }
         }
         catch(SQLException e)
         {
            e.printStackTrace();
         }
```

```
            //Closing the Connection object

            try
            {
                if(con!=null)
                {
                    con.close();
                    con=null;
                }
            }
            catch(SQLException e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

JDBC – SQL INSERT Statement Example

```
importjava.sql.*;

publicclassInsertStatementExample
{
    static
    {
        //STEP 1 : Registering The Driver Class

        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch(ClassNotFoundException e)
        {
            System.out.println("Unable To Load The Driver class");
        }
    }

    publicstaticvoidmain(String[] args)
    {
        Connection con = null;

        Statement stmt = null;

        try
        {
            //Database Credentials
    String URL = "jdbc:oracle:thin:@localhost:1521:XE";

    String username = "username";

    String password = "password";

    //STEP 2 : Creating The Connection Object

    con = DriverManager.getConnection(URL, username, password);

    //STEP 3 : Creating The Statement Object

    stmt = con.createStatement();

    //Constructing The SQL Query

    String sql = "INSERT INTO EMPLOYEE VALUES"+
            "(111, 'Navin', 'Sharma', 'CEO')";

    //Step 4 : Executing The Query

    //We are using executeUpdate() method as we are executing INSERT statement

    inti = stmt.executeUpdate(sql);

    if(i != 0)
    {
        System.out.println("Row is created");
    }
    else
    {
        System.out.println("Row is not created");
    }
}
catch(SQLException e)
{
    e.printStackTrace();
}
finally
{
    //STEP 5 : Closing The DB Resources

    //Closing the Statement object
    try
```

```
                {
                    if(stmt!=null)
                    {
                        stmt.close();
                        stmt=null;
                    }
                }
                catch(SQLException e)
                {
                    e.printStackTrace();
                }

                //Closing the Connection object

                try
                {
                    if(con!=null)
                    {
                        con.close();
                        con=null;
                    }
                }
                catch(SQLException e)
                {
                    e.printStackTrace();
                }
            }
        }
    }
```

JDBC – SQL SELECT Statement Example

```
importjava.sql.*;

publicclassSelectStatementExample
{
    static
    {
        //STEP 1 : Registering The Driver Class

        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch(ClassNotFoundException e)
        {
            System.out.println("Unable To Load The Driver class");
    }
}

publicstaticvoidmain(String[] args)
{
    Connection con = null;

    Statement stmt = null;

    ResultSet rs = null;

    try
    {
        //Database Credentials

        String URL = "jdbc:oracle:thin:@localhost:1521:XE";

        String username = "username";

        String password = "password";

        //STEP 2 : Creating The Connection Object

        con = DriverManager.getConnection(URL, username, password);

        //STEP 3 : Creating The Statement Object

        stmt = con.createStatement();

        //Constructing The SQL Query

        String sql = "SELECT * FROM EMPLOYEE";

        //Step 4 : Executing The Query

        //We are using executeQuery() method as we are executing SELECT statement

        rs = stmt.executeQuery(sql);

        //Processing the ResultSet object

        while(rs.next())
        {
            System.out.println("ID :"+rs.getInt(1));
        System.out.println("First Name : "+rs.getString(2));

        System.out.println("Last Name :"+rs.getString(3));

        System.out.println("Designation :"+rs.getString(4));

        System.out.println("-------------------");
    }
}
catch(SQLException e)
{
    e.printStackTrace();
}
finally
{
    //STEP 5 : Closing The DB Resources

    //Closing the ResultSet object

    try
    {
        if(rs!=null)
        {
            rs.close();
            rs=null;
        }
    }
    catch(SQLException e)
    {
        e.printStackTrace();
    }

    //Closing the Statement object

    try
    {
        if(stmt!=null)
        {
            stmt.close();
            stmt=null;
        }
    }
    catch(SQLException e)
    {
        e.printStackTrace();
```

```
            }

            //Closing the Connection object

            try
            {
                if(con!=null)
                {
                    con.close();
                    con=null;
                }
            }
            catch(SQLException e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

4) JDBC – SQL UPDATE Statement Example

```
importjava.sql.*;

publicclassDeleteStatementExample
{
    static
    {
        //STEP 1 : Registering The Driver Class

        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch(ClassNotFoundException e)
        {
            System.out.println("Unable To Load The Driver class");
        }
    }

    publicstaticvoidmain(String[] args)
    {
        Connection con = null;

        Statement stmt = null;

        try
```

```
{
   //Database Credentials

   String URL = "jdbc:oracle:thin:@localhost:1521:XE";

   String username = "username";

   String password = "password";

   //STEP 2 : Creating The Connection Object

   con = DriverManager.getConnection(URL, username, password);

   //STEP 3 : Creating The Statement Object

   stmt = con.createStatement();

   //Constructing The SQL Query

   String sql = "DELETE FROM EMPLOYEE WHERE ID=111";

   //Step 4 : Executing The Query

   //We are using executeUpdate() method as we are executing DELETE statement

   inti = stmt.executeUpdate(sql);

   if(i != 0)
   {
      System.out.println("Record is deleted");
   }
   else
   {
      System.out.println("Record is not deleted");
   }
}
catch(SQLException e)
{
   e.printStackTrace();
}
finally
{
   //STEP 5 : Closing The DB Resources

   //Closing the Statement object
```

```
                try
                {
                   if(stmt!=null)
                   {
                      stmt.close();
                      stmt=null;
                   }
                }
                catch(SQLException e)
                {
                   e.printStackTrace();
                }

                //Closing the Connection object

                try
                {
                   if(con!=null)
                   {
                      con.close();
                      con=null;
                   }
                }
                catch(SQLException e)
                {
                   e.printStackTrace();
                }
             }
          }
       }
```

APPLICATION

1. JDBC is an API(Application programming interface) which is used in java programming to interact with databases.
2. It also provides a standard to connect a database to a client application.
3. Enterprise applications are created using the JAVA EE technology

VIVA-VOCE

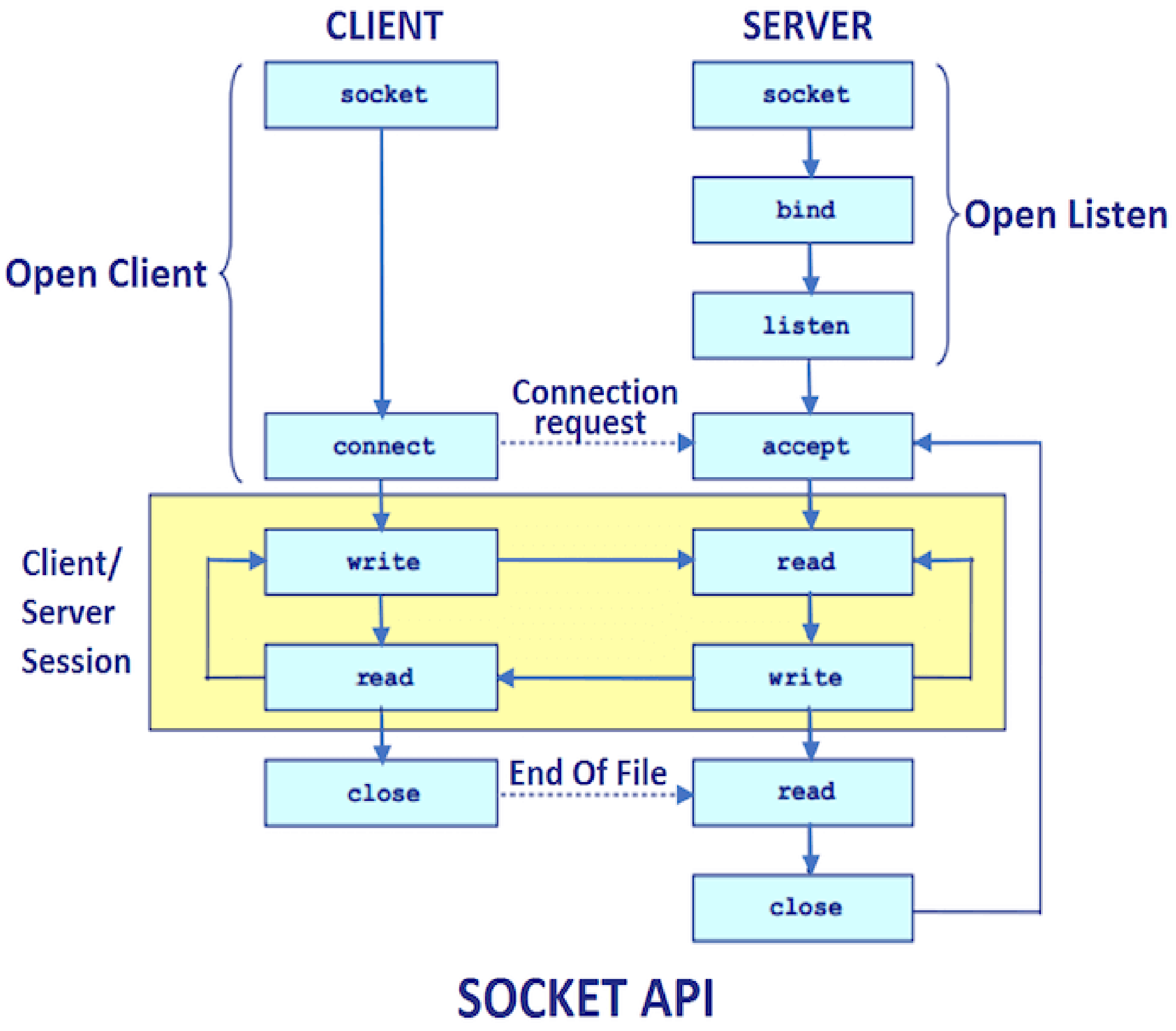
1. What is the executeupdate method in JDBC PreparedStatement?
2. Which interface is responsible for transaction management in JDBC?
3. How can we set null value in JDBC PreparedStatement?
4. What is the return type of Class.forName() method?
5. What are the JDBC statements and JDBC Driver?

ProgramNo. 5

AIM: Write a socket program in java to implement client server communication using socket and server socket class.

THEORY

Java Socket Programming
Java Socket programming is used for communication between the applications running on different JRE.
Java Socket programming can be connection-oriented or connection-less.
Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.
The client in socket programming must know two information:
IP Address of Server, and
Port number.
Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.



Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods

Method	Description
1) public InputStream getInputStream()	returns the InputStream attached with this socket.
2) public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods

Method	Description
1) public Socket accept()	returns the socket and establish a connection between server and client.
2) public synchronized void close()	closes the server socket.

Example of Java Socket Programming

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Let's see a simple of Java socket programming where client sends a text and server receives and prints it.

File: MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}catch(Exception e){System.out.println(e);}
}
}
```

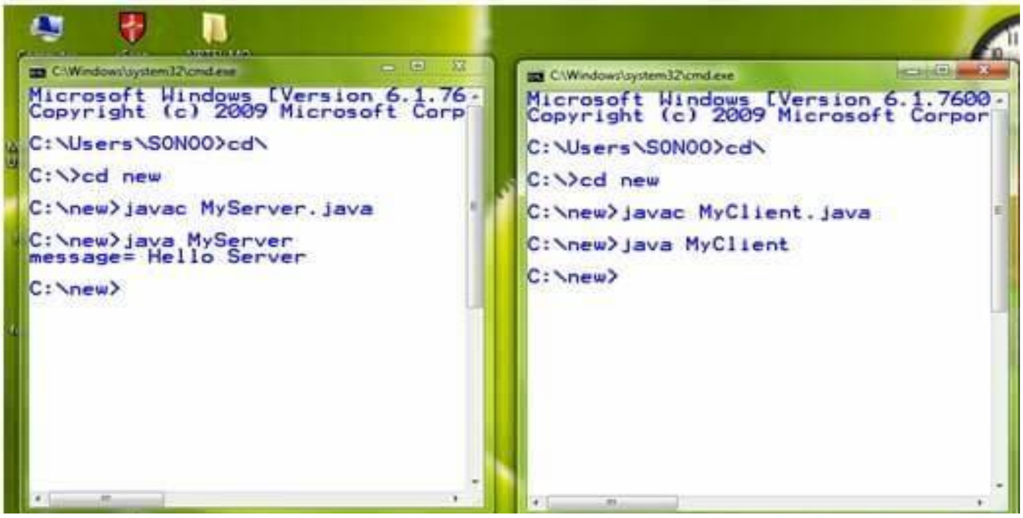
File: MyClient.java

```
import java.io.*;
import java.net.*;
public class MyClient {
```

```
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
dout.writeUTF("Hello Server");
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
}
```

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figure.After running the client application, a message will be displayed on the server console.

OUTPUT



```
Microsoft Windows [Version 6.1.76
Copyright (c) 2009 Microsoft Corp

C:\Users\SONOO>cd\

C:\>cd new

C:\new>javac MyServer.java

C:\new>java MyServer
message= Hello Server

C:\new>
```

```
Microsoft Windows [Version 6.1.7600
Copyright (c) 2009 Microsoft Corpor

C:\Users\SONOO>cd\

C:\>cd new

C:\new>javac MyClient.java

C:\new>java MyClient

C:\new>
```

APPLICATION

1. Socket programming is used with instant messaging, Internet browsers, **file** sharing programs, and anything that forces the computer to connect to a system.
2. Sockets are used to connect software either between different computers or within the same computer so the programs can share data.

VIVA-VOCE

1. How to run a socket program in Java?
2. What does java.net.Socket mean in Java?
3. What does Socket consist of?
4. How does race condition occur?
5. What are the seven layers of Networking?

Program No. 6

AIM: Develop RMI application in which client sends operator and operands for computation and server sends back the result. It behaves just like a distributed calculator app.

THEORY

Calculator.java

```
public interface Calculator extends java.rmi.Remote {
    public long add(long a, long b)
        throws java.rmi.RemoteException;
    public long sub(long a, long b)
        throws java.rmi.RemoteException;
    public long mul(long a, long b)
        throws java.rmi.RemoteException;
    public long div(long a, long b)
        throws java.rmi.RemoteException;
}
```

CalculatorClient.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
public class CalculatorClient {
    public static void main(String[] args) {
        try {
```

```
    Calculator c = (Calculator)
                   Naming.lookup("rmi://localhost/CalculatorService");
    System.out.println( c.sub(4, 3) );
    System.out.println( c.add(4, 5) );
    System.out.println( c.mul(3, 6) );
    System.out.println( c.div(9, 3) );
}
catch (MalformedURLException murle) {
    System.out.println();
    System.out.println("MalformedURLException");
    System.out.println(murle);
}
catch (RemoteException re) {
    System.out.println();
    System.out.println("RemoteException");
    System.out.println(re);
}
catch (NotBoundException nbe) {
    System.out.println();
    System.out.println("NotBoundException");
    System.out.println(nbe);
}
catch (java.lang.ArithmeticException ae) {
```

```
            System.out.println();
            System.out.println("java.lang.ArithmeticException");
            System.out.println(ae);
        }
    }
}
```

CalculatorImpl.java

```
public class CalculatorImpl extends  java.rmi.server.UnicastRemoteObject
    implements Calculator {

    // Implementations must have an
    //explicit constructor
    // in order to declare the
    //RemoteException exception
    public CalculatorImpl() throws java.rmi.RemoteException {
        super();
    }
    public long add(long a, long b) throws java.rmi.RemoteException {
        System.out.println ("Doing addition");
        return a + b;
    }
    public long sub(long a, long b) throws java.rmi.RemoteException {
        System.out.println ("Doing subtraction");
        return a - b;
    }
    public long mul(long a, long b) throws java.rmi.RemoteException {
        System.out.println ("Doing multiplication");
        return a * b;
    }

    public long div(long a, long b) throws java.rmi.RemoteException {
        System.out.println ("Doing division");
        return a / b;
    }
```

CalculatorServer.java

```
import java.rmi.Naming;
public class CalculatorServer {
   public CalculatorServer() {
     try {
       Calculator c = new CalculatorImpl();
       Naming.rebind("rmi://localhost:1099/CalculatorService", c);
     } catch (Exception e) {
       System.out.println("Trouble: " + e);
```

```
    }
  }
  public static void main(String args[]) {
    new CalculatorServer();
  }
}
```

APPLICATION

1. Socket programming is used with instant messaging, Internet browsers, **file** sharing programs, and anything that forces the computer to connect to a system.
2. Sockets are used to connect software either between different computers or within the same computer so the programs can share data.

VIVA-VOCE

1. How to run a socket program in Java?
2. What does java.net.Socket mean in Java?
3. What does Socket consist of?
4. How does race condition occur?
5. What are the seven layers of Networking?

Program No. 7

AIM:Write a Servlet Program to display “Hello World” on browser.

THEORY

Servlets are Java classes which service HTTP requests and implement the **javax.servlet.Servlet** interface. Web application developers typically write servlets that extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.

Code

Following is the sample source code structure of a servlet example to show Hello World −

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
publicclassHelloWorldextendsHttpServlet{

privateString message;

publicvoid init()throwsServletException{
// Do required initialization
      message ="Hello World";
}

publicvoid doGet(HttpServletRequest request,HttpServletResponse response)
throwsServletException,IOException{

// Set response content type
      response.setContentType("text/html");

// Actual logic goes here.
PrintWriterout= response.getWriter();
out.println("<h1>"+ message +"</h1>");
}

publicvoid destroy(){
// do nothing.
}
}
```

Compiling a Servlet

Let us create a file with name HelloWorld.java with the code shown above. Place this file at C:\ServletDevel (in Windows) or at /usr/ServletDevel (in Unix). This path location must be added to CLASSPATH before proceeding further.

Assuming your environment is setup properly, go in ServletDevel directory and compile HelloWorld.java as follows −

```
$ javac HelloWorld.java
```

If the servlet depends on any other libraries, you have to include those JAR files on your CLASSPATH as well. I have included only servlet-api.jar JAR file because I'm not using any other library in Hello World program.

This command line uses the built-in javac compiler that comes with the Sun Microsystems Java Software Development Kit (JDK). For this command to work properly, you have to include the location of the Java SDK that you are using in the PATH environment variable.

If everything goes fine, above compilation would produce HelloWorld.class file in the same directory. Next section would explain how a compiled servlet would be deployed in production.

Servlet Deployment

By default, a servlet application is located at the path <Tomcat-installationdirectory>/webapps/ROOT and the class file would reside in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.

If you have a fully qualified class name of com.myorg.MyServlet, then this servlet class must be located in WEB-INF/classes/com/myorg/MyServlet.class.

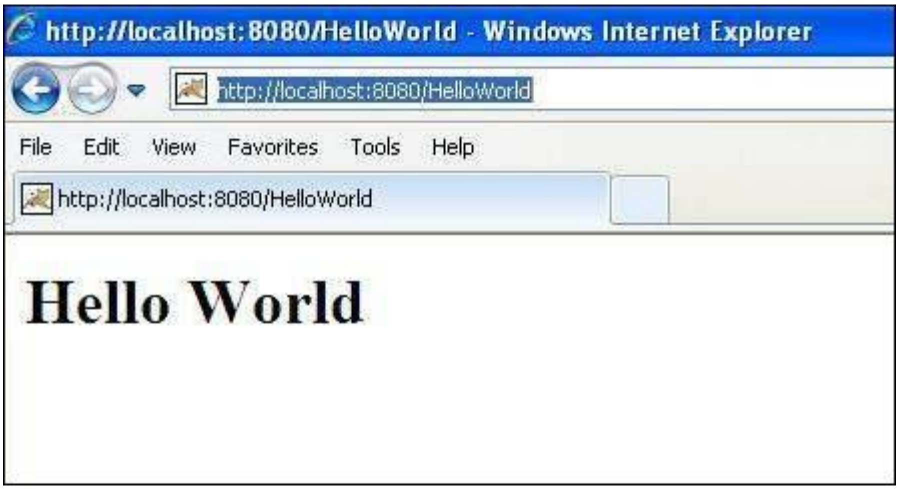
For now, let us copy HelloWorld.class into <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in web.xml file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
<servlet-name>HelloWorld</servlet-name>
<servlet-class>HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HelloWorld</servlet-name>
<url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

Above entries to be created inside <web-app>...</web-app> tags available in web.xml file. There could be various entries in this table already available, but never mind.

You are almost done, now let us start tomcat server using <Tomcat-installationdirectory>\bin\startup.bat (on Windows) or <Tomcat-installationdirectory>/bin/startup.sh (on Linux/Solaris etc.) and finally type http://localhost:8080/HelloWorld in the browser's address box. If everything goes fine, you would get the following result

OUTPUT


http://localhost:8080/HelloWorld - Windows Internet Explorer
http://localhost:8080/HelloWorld
File
Edit
View
Favorites
Tools
Help
http://localhost:8080/HelloWorld

Hello World

.

APPLICATION

1. Servlet is a technology which is used to create a web application.
2. Servlet is an API that provides many interfaces and classes including documentation.
3. Servlet is an interface that must be implemented for creating any Servlet.
4. Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
5. Servlet is a web component that is deployed on the server to create a dynamic web page.

VIVA VOCE

1. Why servlet is mostly used?
2. Explain servlet mapping and servlet context?
3. What is life cycle of Servlet?
4. What is the difference between Servlet Request and Servlet Context when calling a Request Dispatcher?
5. Which interface should be implemented by all servlets?

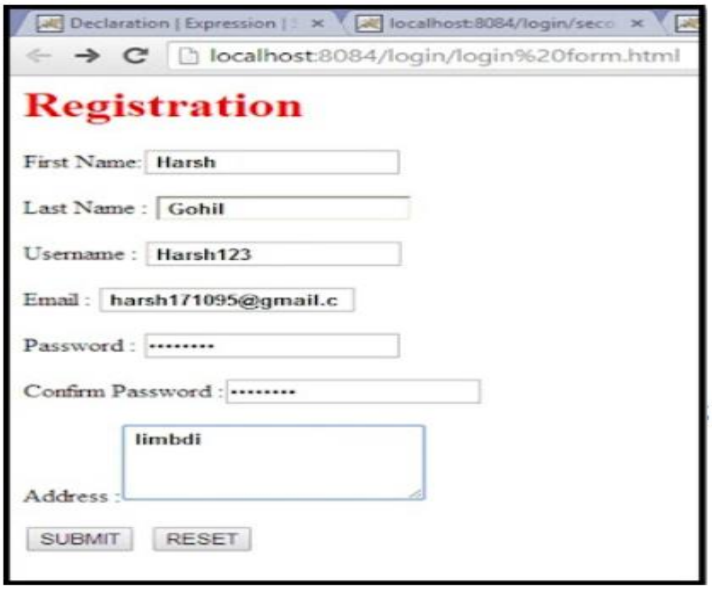
Program No. 8

AIM: Write a servlet program for user registration &amp; then control will be transfer into second page

PROGRAM CODE

```
<html>
<head>
<title></title>
</head>
<body>
<p>
<%
String fname=request.getParameter("fname");
String lname=request.getParameter("lname");
String uname=request.getParameter("uname");
String email=request.getParameter("email");
String pwd=request.getParameter("pwd");
String cpwd=request.getParameter("cpwd");
String add=request.getParameter("add");
out.print("First name is : "+fname+"<br>");
out.print("Last name is : "+lname+"<br>");
out.print("User name is : "+uname+"<br>");
out.print("Email is : "+email+"<br>");
out.print("Password is : "+pwd+"<br>");
out.print("Confrim Password is : "+cpwd+"<br>");
out.print("Address is : "+add+"<br>");
%>
</p>
</body>
</html>
```

OUTPUT


Declaration | Expression |
localhost:8084/login/seco
localhost:8084/login/login%20form.html

Registration

First Name: Harsh

Last Name : Gohil

Username : Harsh123

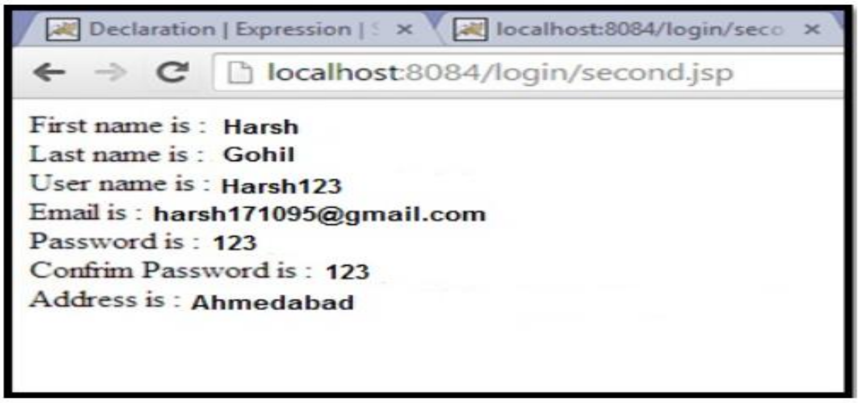
Email : harsh171095@gmail.c

Password : ••••••••

Confirm Password : ••••••••

Address : limbdi

) Second.jsp


Declaration | Expression |
localhost:8084/login/seco
localhost:8084/login/second.jsp

First name is : Harsh
Last name is : Gohil
User name is : Harsh123
Email is : harsh171095@gmail.com
Password is : 123
Confirm Password is : 123
Address is : Ahmedabad

APPLICATION

1.Used to create registration form to get information

VIVA-VOCE

1. How is JSP better than Servlet technology?
2. How many objects of a servlet is created?
3. What is the life-cycle of a servlet?
4. When servlet object is created?
5. What is difference between Get and Post method?

Program No. 9

AIM: Write a JSP program for user login form with static and dynamic database.

THEORY

StaticDatabase

*FileName.
-Index.html : For UserName & Password.
-Process.jsp: For Static Database.
-Welcome.jsp :Welcome File

* ProgramCode

Index.html

```
   <head>
      <title></title>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   </head>
   <body>
      <form action="Process.jsp" method="post">
      Username:<input type="text" name="uname">
      password:<input type="password" name="pass">
      <input type="submit">
      </form>

   </body>
</html>
```

Process.jsp

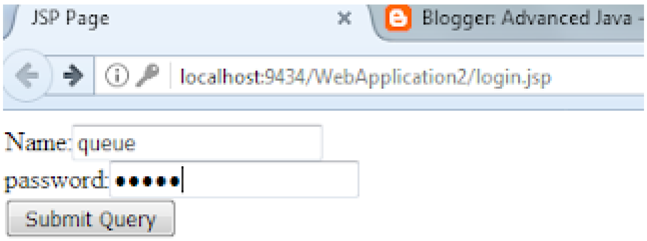
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<%

    String a=request.getParameter("uname");

    String b=request.getParameter("pass");
```

```
 if(a.equals("queue") && b.equals("queue"))

       {
             response.sendRedirect("Welcome.jsp");

       }

 %>
```

Welcome.jsp

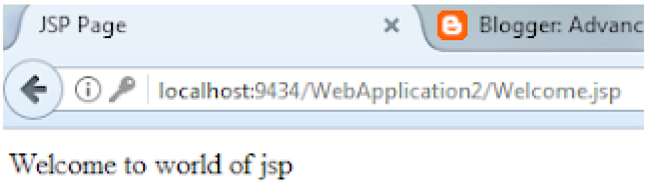
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%
    out.println("Welcome to world of jsp");
    %>
```

OUTPUT



localhost:9434/WebApplication2/login.jsp

Name: queue
password: •••••



localhost:9434/WebApplication2/Welcome.jsp

Welcome to world of jsp

***Dynamic Database**

*FileName

-index.jsp :For Register Form.

-servlet.jsp: Insert data in data base & redirect page login.jsp

-login.jsp : Make sure Login Data.

-wel.jsp:Welcome File.

*PROGRAM CODE

*index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <form action="servlet.jsp" method="post">
         Enter your UserName:<input type="text" name="name"><br>
         Enter your Password:<input type="password" name="pas"><br>
         Enter your email:<input type="email" name="email"><br>

         <input type="submit">
      </form>

   </body>
</html>
```

*servlet.jsp

```
<%@page import="java.sql.*"%>
<%@page import="java.io.*"%>
<%@page import="java.util.*"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <%
          String name=request.getParameter("name");
          String pass=request.getParameter("pas");
          String email=request.getParameter("email");
```

```
            Class.forName("com.mysql.jdbc.Driver");
            Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/phgraph","root","ujash7878
");
            PreparedStatement ps=con.prepareStatement("insert into reg values(?,?,?)");

ps.setString(1,name);
ps.setString(2, pass);
ps.setString(3, email);

 int i=ps.executeUpdate();

if(i>0)
{
   response.sendRedirect("login.jsp");
}
else
{
   out.println("Try Again");
}



      %>
   </head>
   <body>
</body>
</html>
```

*login.jsp

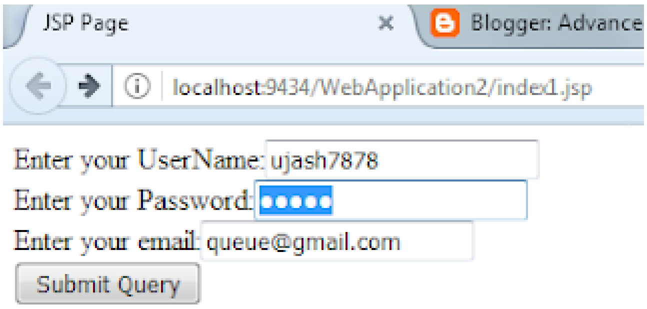
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <form action="wel.jsp" method="post">
         Name:<input type="text" name="name"><br>
```

```
            password:<input type="password" name="pas"><br>
            <input type="submit">
        </form>


    </body>
</html>
```

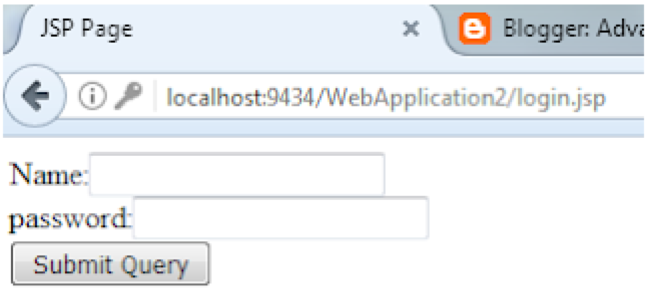
*Wel.jsp

```
<%@page import="java.io.*"%>
<%@page import="java.util.*" %>
<%@page import="java.sql.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            String name=request.getParameter("name");
              String pass=request.getParameter("pas");

 Class.forName("com.mysql.jdbc.Driver");
              Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/phgraph","root","ujash7878
");
              PreparedStatement ps=con.prepareStatement("select * from reg where name=? and
pas=?");
              ps.setString(1, name);
              ps.setString(2, pass);

   ResultSet rs=ps.executeQuery();

       if(rs.next())
       {
           out.println("<h1>Welcome :"+name+"</h1>");
       }
       else
       {
           out.println("<h1>Not Done</h1>");
       }
       %>
    </body>
</html>
```

OUTPUT



Enter your UserName: ujash7878
Enter your Password: •••••
Enter your email: queue@gmail.com



Name:
password:

VIVAVOCE

1. What are the life-cycle methods for a JSP?
2. Give the syntax for JSP comments
3. What are the JSP implicit objects?
4. How can we handle the exceptions in JSP?
5. How can we forward the request from JSP page to the servlet?

Program No. 10

AIM: Write a JSP program to display the grade of a student by accepting the marks of five subjects

PROGRAM CODE

File Name

- Marks.html :Insert five subject marks.
- marks.jsp: obtaining grade

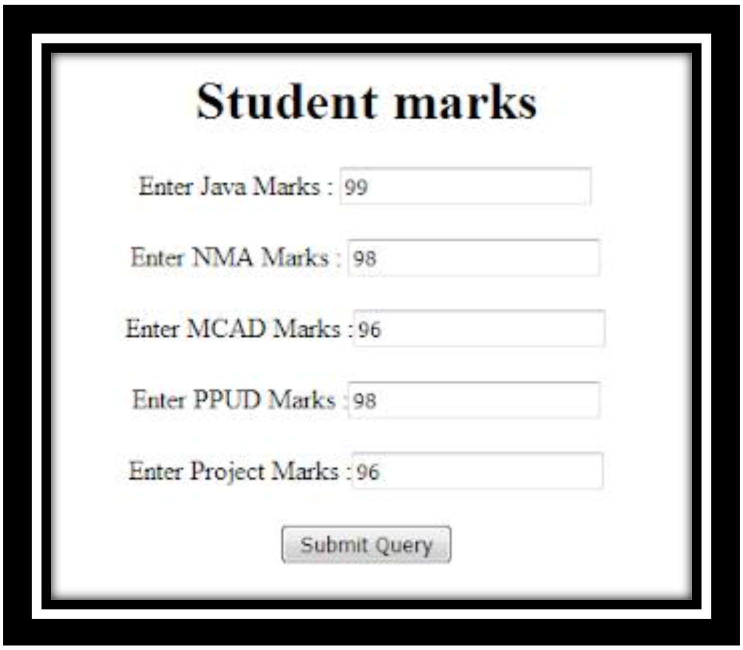
-marks.html

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>TODO supply a title</title>

    </head>
    <body>
    <center>
        <h1>Student marks</h1>
        <form action="marks.jsp" method="get">
           Enter Makrs in Java : <input type="text" name="java"><br><br>
            Enter NMA Marks : <input type="text"  name="NMA"><br><br>
            Enter MCAD Marks :<input type="text"  name="MCAD"><br><br>
            Enter PPUD Marks :<input type="text"  name="PPUD"><br><br>
            Enter Project Marks :<input type="text"  name="pro"><br><br>
            <input type="submit">
           </form>
        </center>
    </body>
</html>
```

Marks.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <%
          int java=Integer.parseInt(request.getParameter("java"));
          int NMA=Integer.parseInt(request.getParameter("NMA"));
          int MCAD=Integer.parseInt(request.getParameter("MCAD"));
          int PPUD=Integer.parseInt(request.getParameter("PPUD"));
          int Project=Integer.parseInt(request.getParameter("pro"));

         int c=java+NMA+MCAD+PPUD+Project;
         double avg=c/5;


       if(avg > 90 )
       {
          out.println(" your grade is A");
       }else if (avg >= 80) {
         out.println("your grade is b");
      } else if (avg >= 70) {
         out.println("your grade is c");
      } else if (avg >= 60) {
          out.println("your grade is d");
      } else {
          out.println("your grade is e");
     }
     %>
   </body>
</html>
```

OUTPUT



Student marks

Enter Java Marks : 99

Enter NMA Marks : 98

Enter MCAD Marks : 96

Enter PPUD Marks : 98

Enter Project Marks : 96

APPLICATION

1. Used to create web application
2. Used to create dynamic web content

VIVA-VOCE

1. Give the use of session object. and exception object
2. What is the difference between ServletContext and PageContext?-
3. What is JSTL?
4. How to disable session in JSP?
5. List the various action tags used in JSP.
6. What are the three tags used in JSP bean development?

Program No. 11

AIM: Case study on Java bean.

Theory

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.
Why use JavaBean?
According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.
Simple example of JavaBean class
//Employee.java

```
package mypack;
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;}
}
```

How to access the JavaBean class?
To access the JavaBean class, we should use getter and setter methods.

```
package mypack;
public class Test{
public static void main(String args[]){
Employee e=new Employee();//object is created
e.setName("Arjun");//setting value to the object
System.out.println(e.getName());
}}
```

Note: There are two ways to provide values to the object. One way is by constructor and second is by setter method.

JavaBean Properties
A JavaBean property is a named feature that can be accessed by the user of the object. The feature can be of any Java data type, containing the classes that you define.
Features of Java - Javatpoint

A JavaBean property may be read, write, read-only, or write-only. JavaBean features are accessed through two methods in the JavaBean's implementation class:

1. getPropertyName ()

For example, if the property name is firstName, the method name would be getFirstName() to read that property. This method is called the accessor.

2. setPropertyName ()

For example, if the property name is firstName, the method name would be setFirstName() to write that property. This method is called the mutator.

Advantages of JavaBean

The following are the advantages of JavaBean:/p>

The JavaBean properties and methods can be exposed to another application.

It provides an easiness to reuse the software components.

Disadvantages of JavaBean

The following are the disadvantages of JavaBean:

JavaBeans are mutable. So, it can't take advantages of immutable objects.

Creating the setter and getter method for each property separately may lead to the boilerplate code.

APPLICATION

1. Beans are capable of serilizable interface.
2. It is used to create Dynamic Web pages?

VIVA-VOCE

1. What are entity Beans?
2. Why do I get null pointer Exception when loading a jar file?
3. State the purpose of introspeaction?
4. Define bean persistent Property?
5. Why beans is not an applet?

Program No. 12

AIM: Case study on Enterprise Java Beans.

THEORY:

What is EJB

EJB is an acronym for *enterprise java bean*. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.To get information about distributed applications.

To run EJB application, you need an *application server* (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:life cycle management,security,transaction management, and object pooling.

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.

When use Enterprise Java Bean?

Application needs Remote Access. In other words, it is distributed.

Application needs to be scalable. EJB applications supports load balancing, clustering and fail-over.

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 Bean

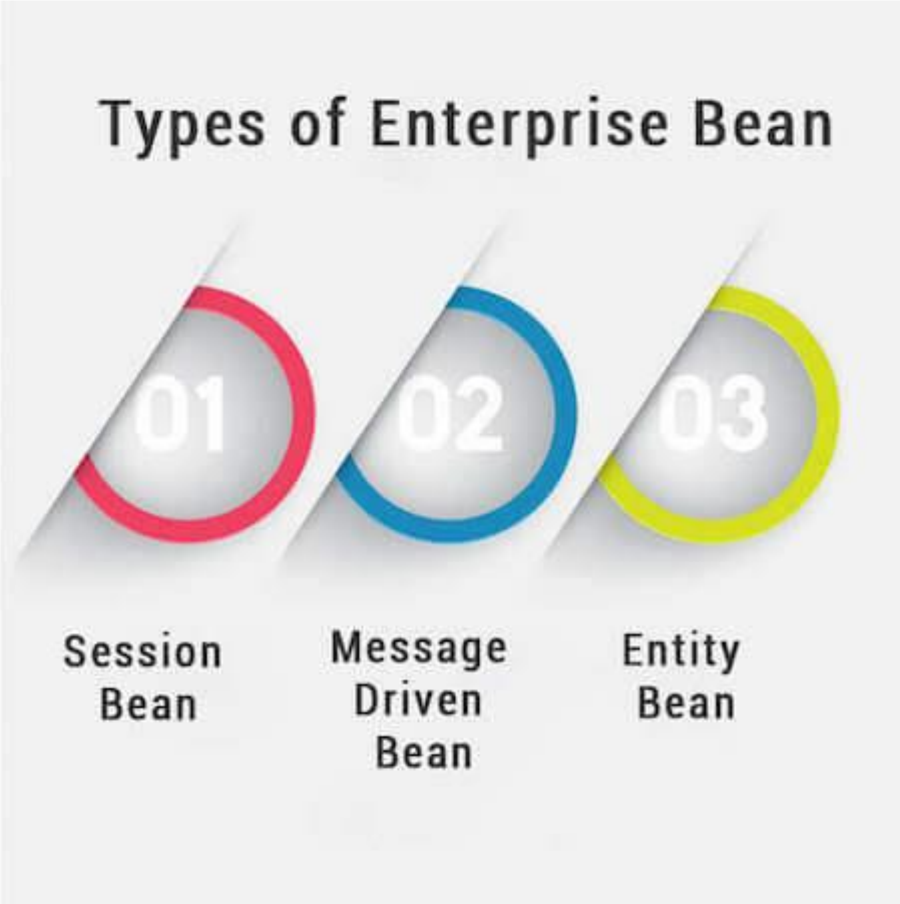
Session bean contains business logic that can be invoked by local, remote or webservice client.

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).



EJB and Webservice
In EJB, bean component and bean client both must be written in java language.
If bean client need to be written in other language such as .net, php etc, we need to go with webservices (SOAP or REST). So EJB with web service will be better option.

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.

APPLICATION

1. Simplified development of large scale enterprise level application.
2. It is used in transaction handling and persistent mechanism.

VIVA-VOCE

1. What are the different types of enterprise bean?

2. Define singleton session bean?
3. When should I use EJB?
4. Difference Between EJB and Java Beans?
5. Can we run EJB in web server like Tomcat?