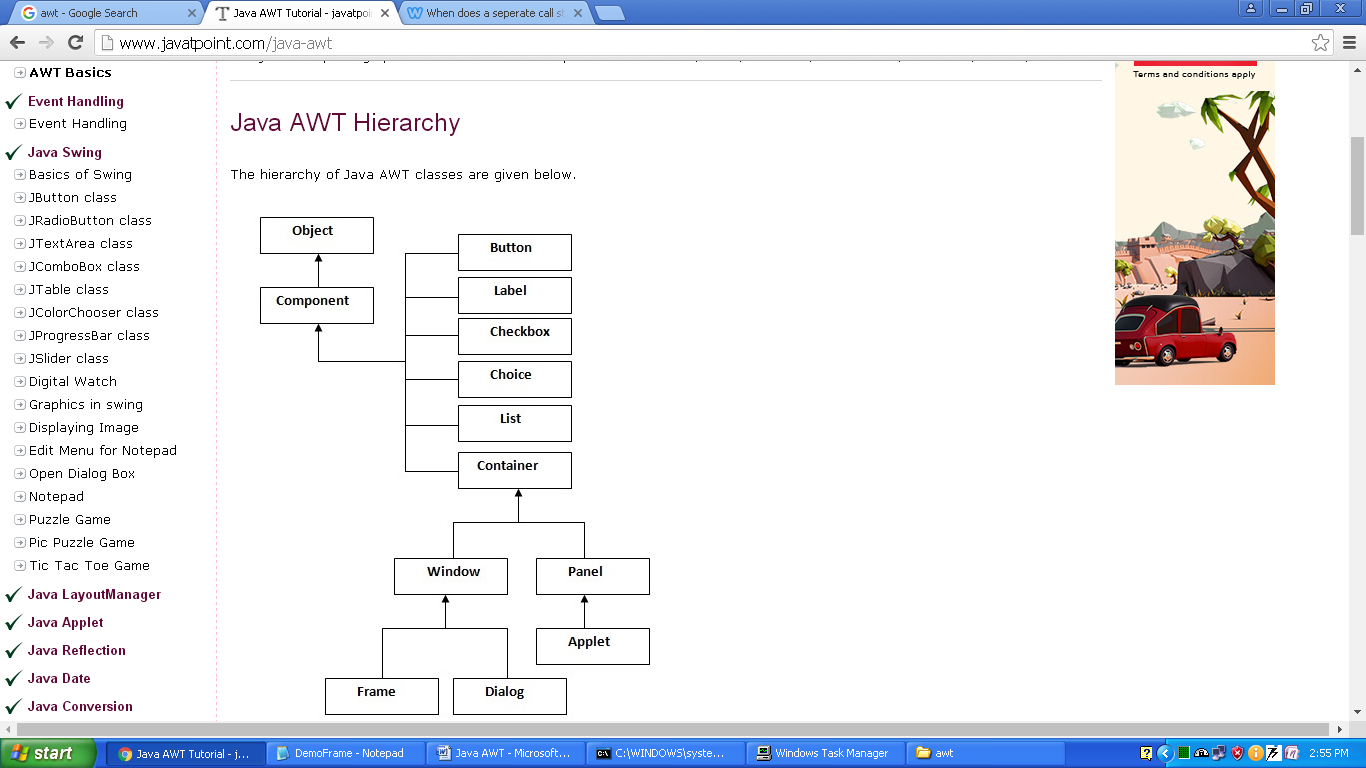
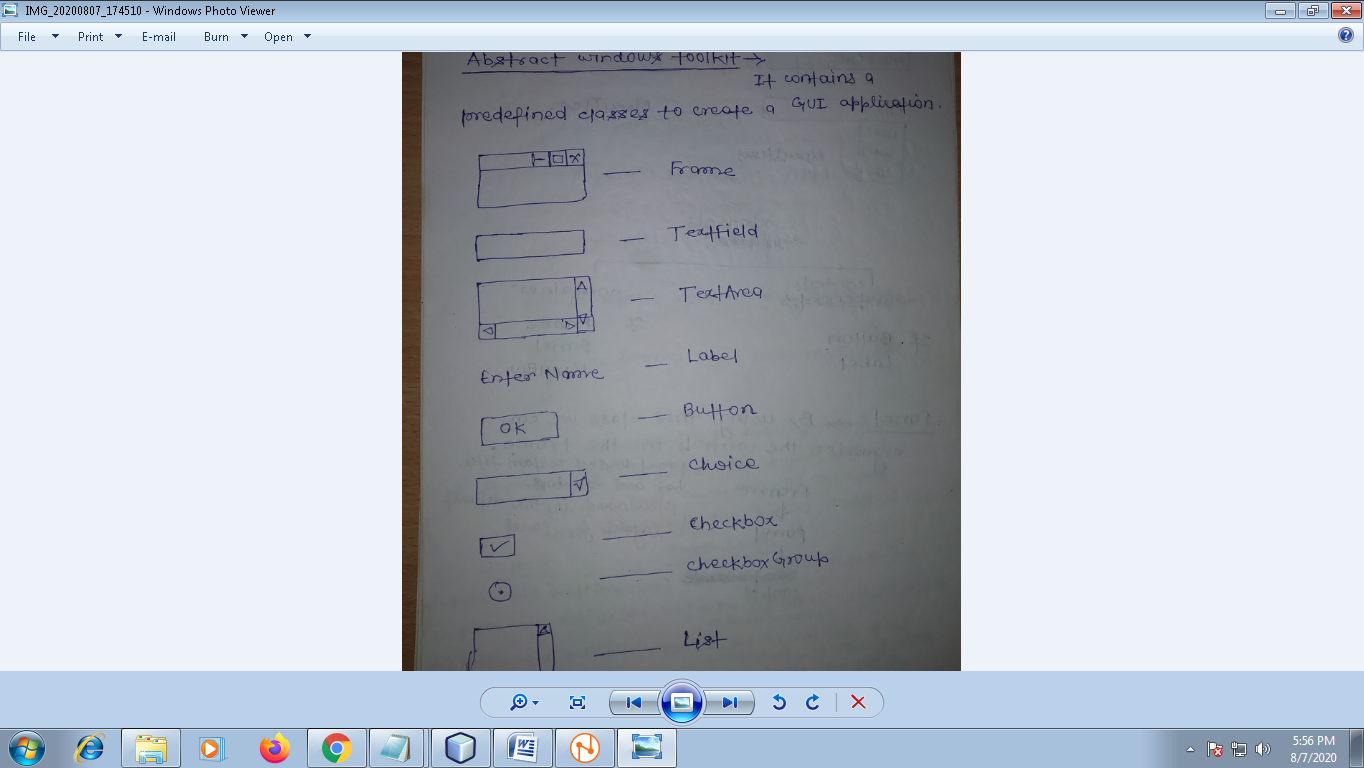
**Java AWT**

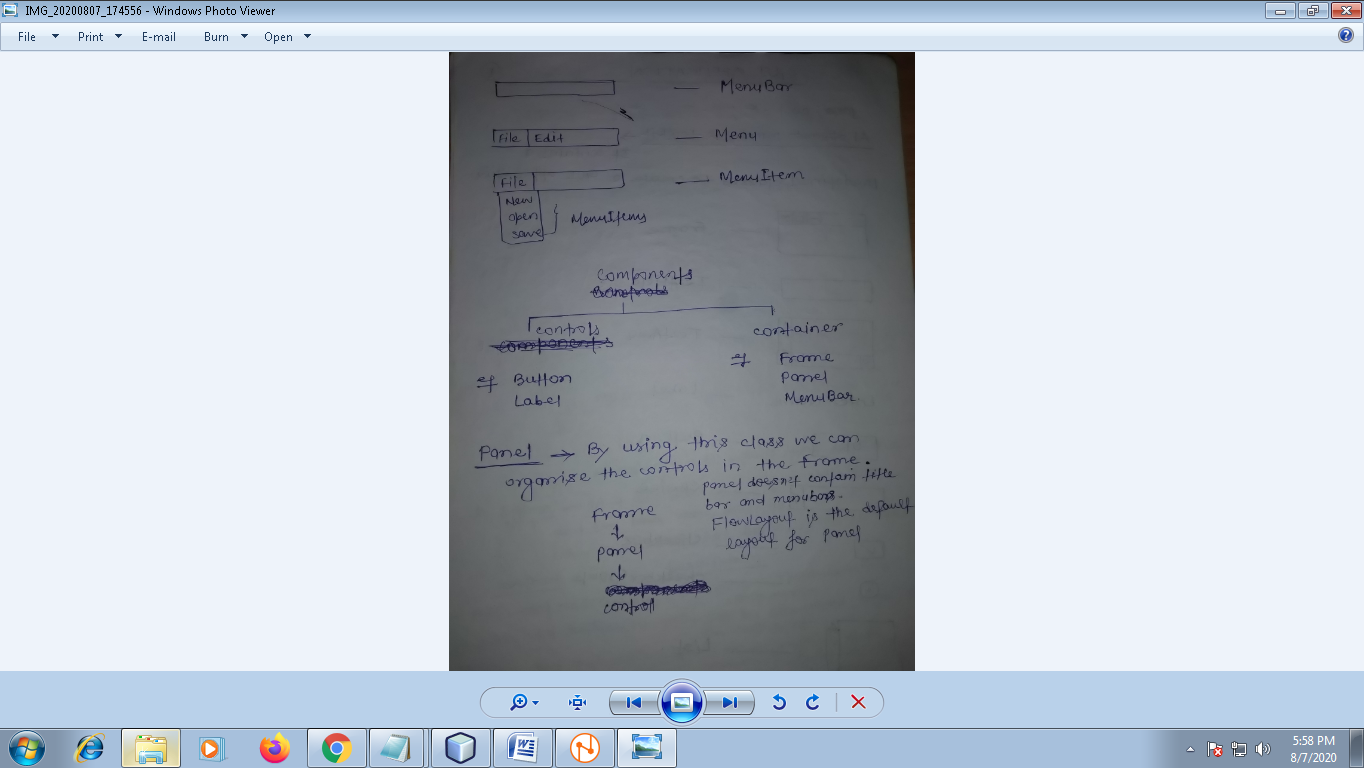
Java **AWT (Abstract Windowing Toolkit) is an API** to develop ***GUI or window-based application*** in java.

Java AWT components are ***platform-dependent*** i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components uses the resources of system.

The ***java.awt*** package provides classes for AWT API such as **TextField, Label, TextArea, RadioButton, CheckBox, Choice, List** etc.



****

****

**Container**

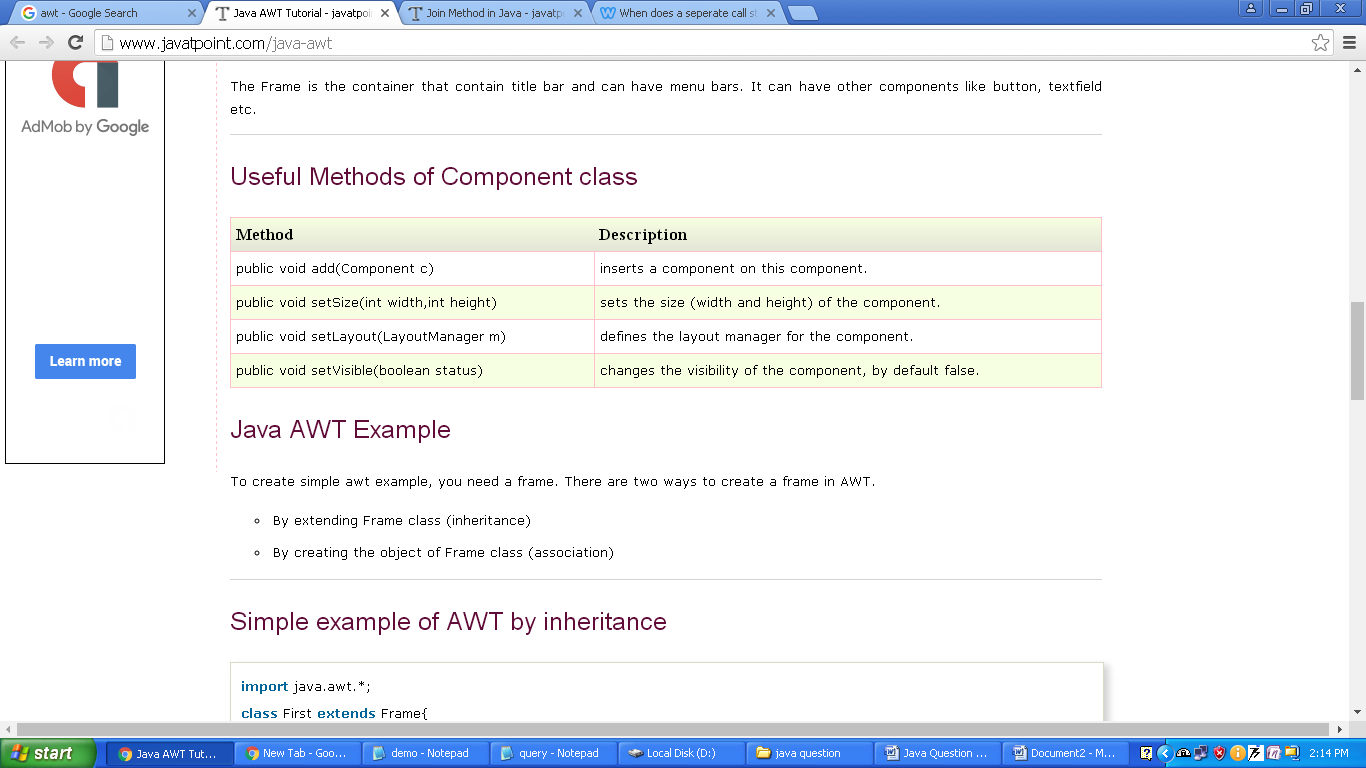
The Container is a component in AWT that can contain another component like ***Buttons, TextField, Label* etc.**

**Panel**

* By using ***“Panel” class*** we can organize the ***component like Button, TextField*** etc in the container ***i.e Frame or Applet***.
* The **Panel** is the container that doesn't contain ***title bar*** and ***menu bars.*** “**FlowLayout”** is the ***default layout for Panel.***

**Frame**

The Frame class is the container that contain title bar and menu bars. It can have other ***components like Button, TextField etc. “BorderLayout”*** is the ***default layout for Frame.***

****

**Create AWT application**

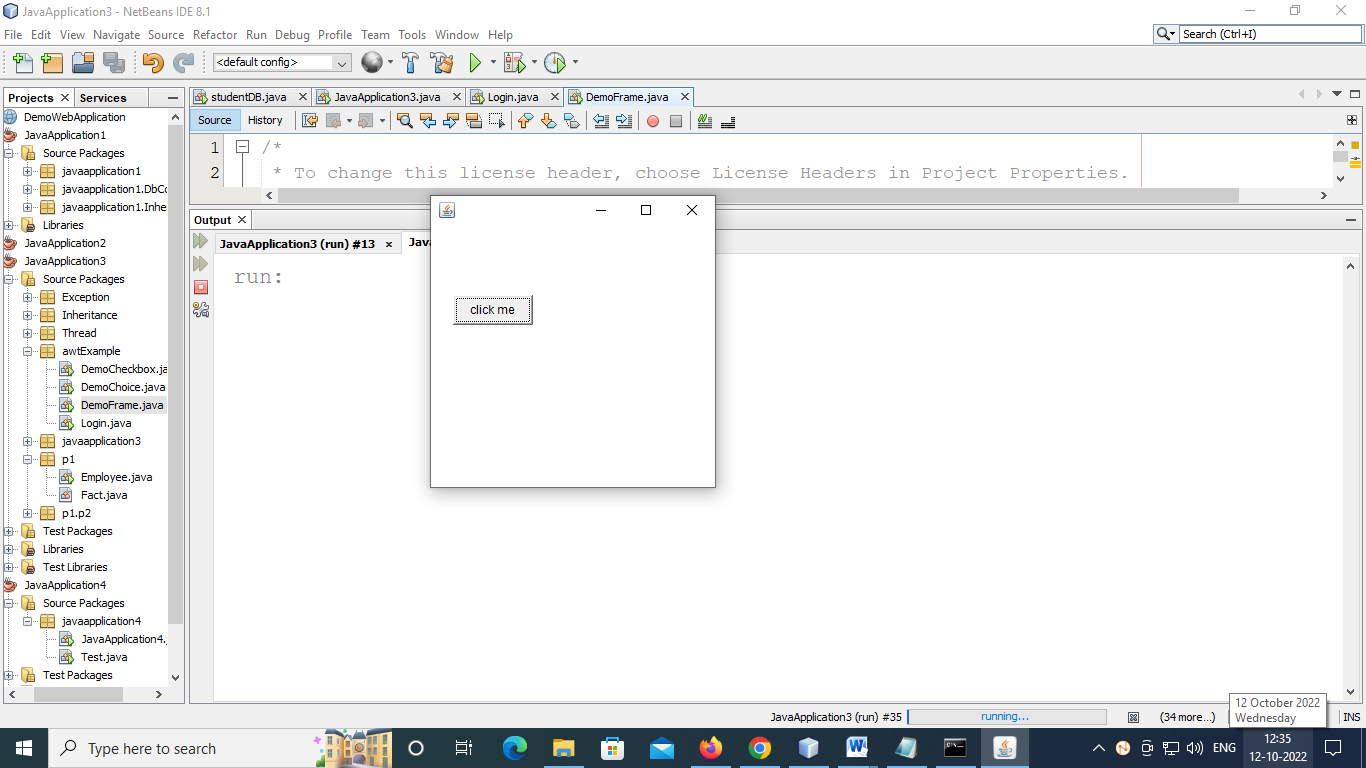
* To create AWT application, you need a ***Frame.***
* There are two ways to create a Frame in AWT.

1. By extending Frame class (inheritance)
2. By creating the object of Frame class

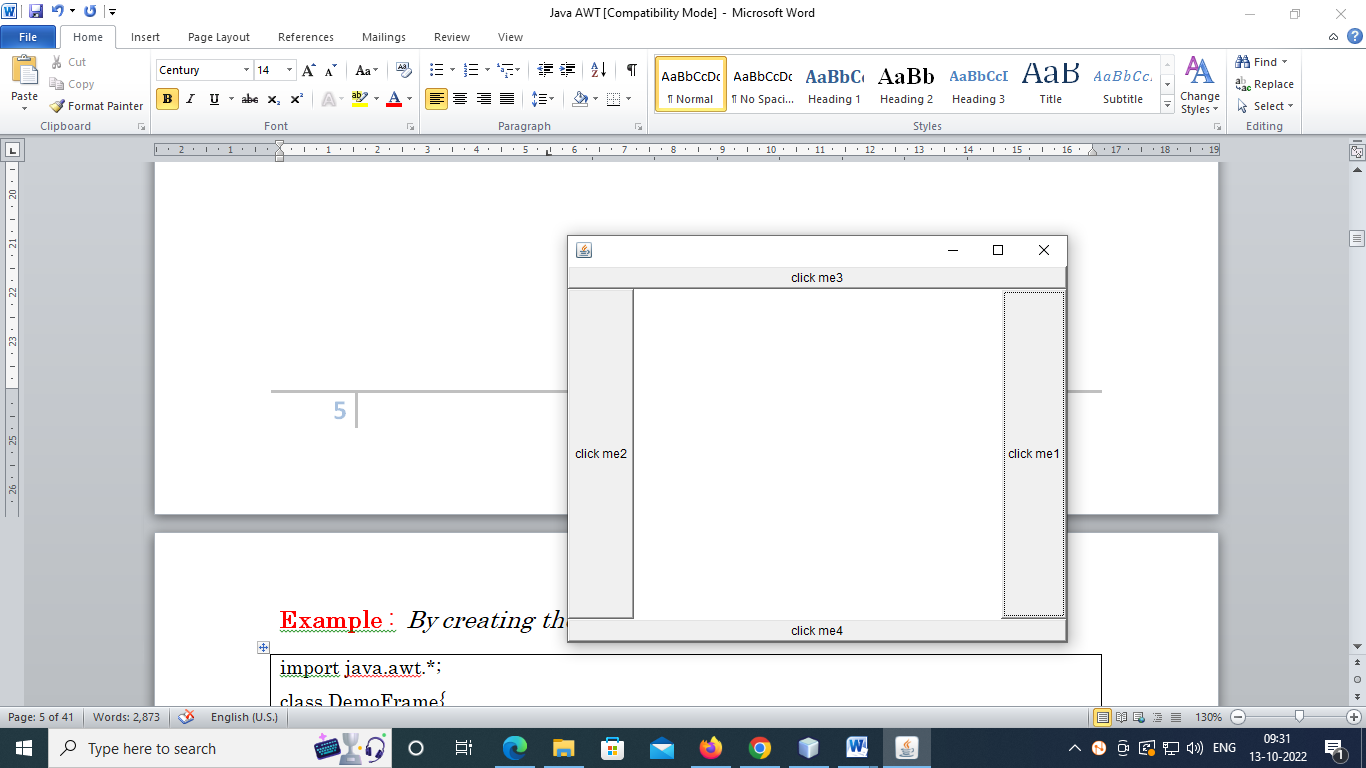
**Example:** *By extending Frame class*

|  |
| --- |
| import java.awt.\*;  class ***DemoFrame*** extends ***Frame***{  DemoFrame() //Constructor  {  Button b=new Button("click me");  b.setBounds(30,100,80,30); // setting button position  add(b); //adding button into frame  setSize(300,300); //frame size 300 width and 300 height  setLayout(null); //no layout manager  setVisible(true); //now frame will be visible, by default not visible  }  public static void main(String args[]){  new DemoFrame();  }  } |

***Note:*** The **setBounds(int xaxis, int yaxis, int width, int height)** method is used to sets the position of the AWT button.



|  |
| --- |
| **import java.awt.\*;**  **class DemoFrame extends Frame{**  **DemoFrame() //Constructor**  **{**  **Button b1=new Button("click me1");**  **Button b2=new Button("click me2");**  **Button b3=new Button("click me3");**  **Button b4=new Button("click me4");**  **add("East",b1); //adding button into frame**  **add("West",b2);**  **add("North",b3);**  **add("South",b4);**    **setSize(300,300); //frame size 300 width and 300 height**    **setVisible(true); //now frame will be visible, by default not visible**  **}**  **public static void main(String args[]){**  **new DemoFrame();**  **}**  **}** |



**Example :** *By creating the object of Frame class*

|  |
| --- |
| import java.awt.\*;  class DemoFrame{  DemoFrame(){ //constructor  Frame f=new Frame();  Button b=new Button("click me");  b.setBounds(30,50,80,30);  f.add(b);  f.setSize(300,300);  f.setLayout(null);  f.setVisible(true);  }  public static void main(String args[]){  new DemoFrame();  }  } |

**TextField**

TextField class is a text component that allows the editing of a single line text.

Eg:

|  |
| --- |
| **Constructor**  **TextField(int columns)**  Constructs a new empty text field with the specified number of columns.  **TextField** t1;  t1=new **TextField(20);**  p1.add(t1); //add TextField to the panel |

|  |
| --- |
| \*To accept password character in TextField  t1.setEchoChar(‘\*’); |
| \*To retrieve the data from the TextField    t1.getText(); |
| \*To set the data in the TextField  t1.setText(); |

**Panel**

The Panel class is a simplest container class. It provides space in which an application can attach any other component. **Or** you can say that the Panel class is used to organize the controls in the Frame.

|  |
| --- |
| **Constructor**  Panel()  Creates a new panel using the default layout manager.  Panel p1;  p1=new Panel(); or p1=new Panel(null); |

|  |
| --- |
| **\*To add panel to the Frame**  add(p1); |
| **\*To add other component in the Panel**  p1.add(t1); |

**Label**

Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

|  |
| --- |
| **Constructor**  **Label(String text)**  Constructs a new label with the specified string of text, left justified.  Eg:  Label lblrno;  lblrno=new Label(“Enter the Roll Number:”);  p1.add(lblrno); |

|  |
| --- |
| **Label()**  Constructs an empty label.  Eg:  Label lblrno=new Label();  **\*To set the caption to Label**  lblrno.setText(“Enter the Roll Number”);  eg:  Label lblrno=new Label();  lblrno.setText(“Enter the Roll Number”); |

**Button**

The button class is used to create a labeled button. The application result in some action when the button is pushed.

|  |
| --- |
| **Button(String text)**  Constructs a new button with specified label.  Eg:  **Button** btn;  btn=new Button(“OK”);  p1.add(btn); |

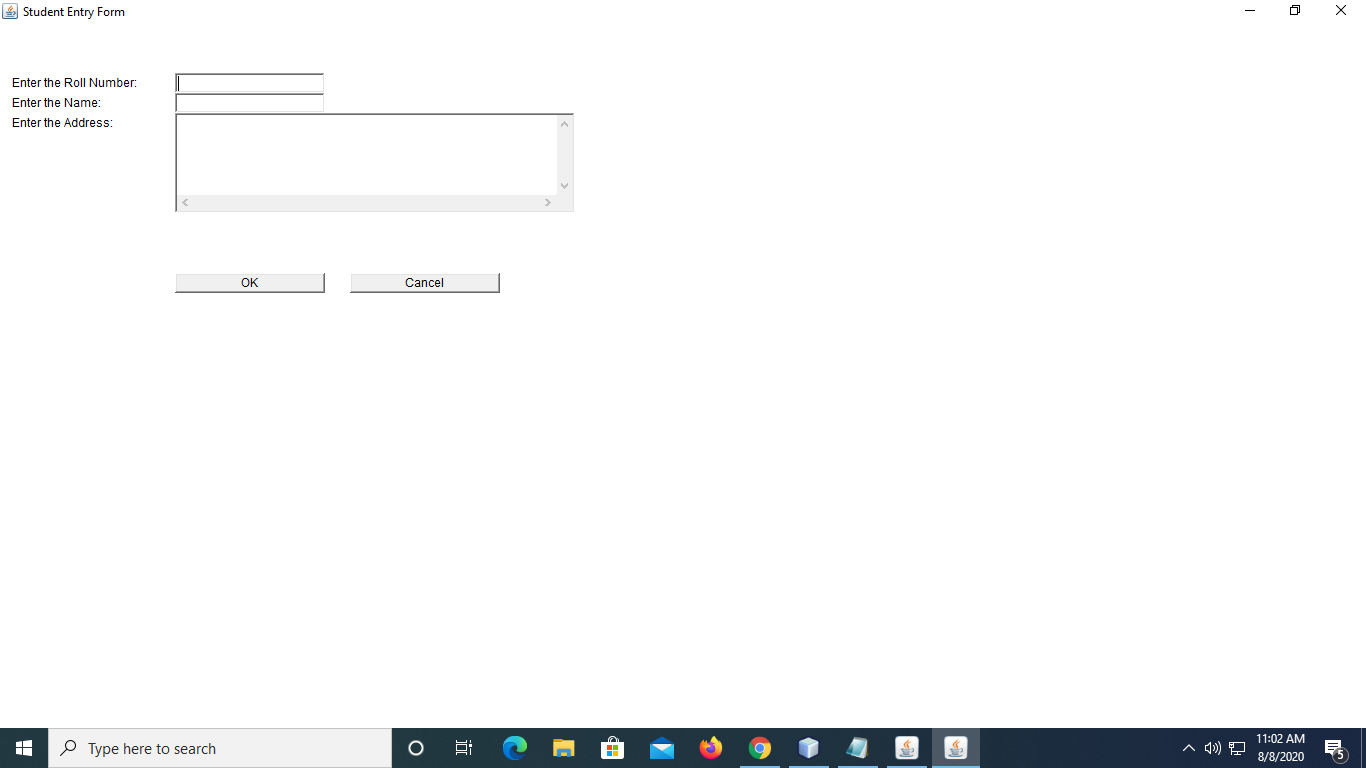
|  |
| --- |
| **Button()**  Constructs a button with an empty string for its label.  Eg:  **Button** btn=new **Button**();  **\*To set the caption to Button**  btn.setLabel(“OK”);  eg:  Button btn=new Button();  btn.setLabel(“OK”); |

TextArea

TextArea class is a multi line region that displays text.

|  |
| --- |
| **TextArea(int rows, int columns)**  Constructs a new text area with the specified number of rows and columns and the empty string as text.  Eg:  TextArea ta;  ta=new TextArea(4,4); |

Q: Demonstrate the AWT program for the following Student Entry Form using Frame



import java.awt.\*;

public class StudentForm extends Frame {

Label lblRno,lblName,lblAddr;

TextField txtRno,txtName;

TextArea txtAddr;

Button btnOk,btnCancel;

Panel p1;

public StudentForm()

{

super("Student Entry Form");

lblRno=new Label("Enter the Roll Number:");

lblRno.setBounds(10,50,150,20);

lblName=new Label("Enter the Name:");

lblName.setBounds(10,70,150,20);

lblAddr=new Label("Enter the Address:");

lblAddr.setBounds(10,90,150,20);

txtRno=new TextField(20);

txtRno.setBounds(175,50,150,20);

txtName=new TextField(20);

txtName.setBounds(175,70,150,20);

txtAddr=new TextArea(5,4);

txtAddr.setBounds(175,90,400,100);

btnOk=new Button("OK");

btnOk.setBounds(175,250,150,20);

btnCancel=new Button("Cancel");

btnCancel.setBounds(350,250,150,20);

p1=new Panel(null);

add(p1);

p1.add(lblRno); p1.add(txtRno);

p1.add(lblName); p1.add(txtName);

p1.add(lblAddr); p1.add(txtAddr);

p1.add(btnOk);p1.add(btnCancel);

setSize(400, 400);

setVisible(true);

}

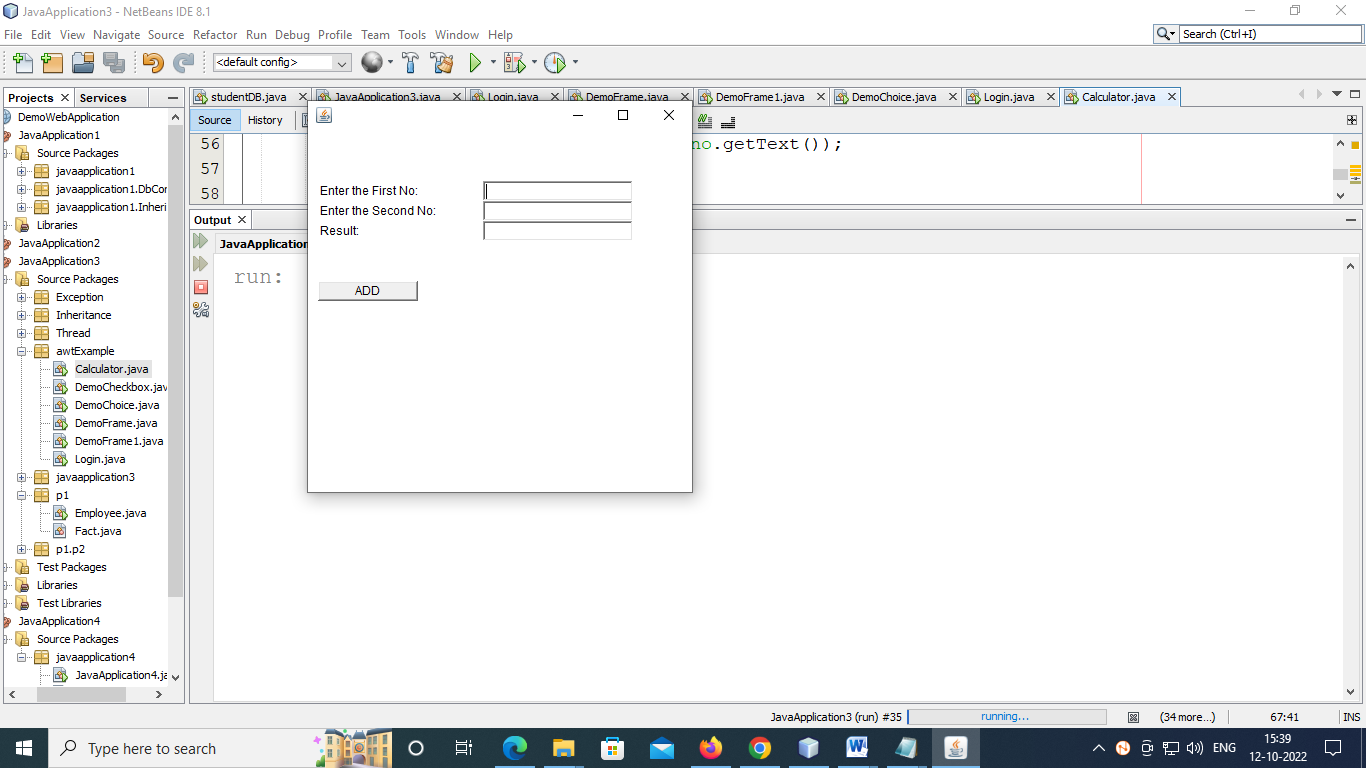
public static void main(String[] args) {

StudentForm sf=new StudentForm();

}

}

Q. Q: Demonstrate the AWT program for the following Entry Form using Frame



|  |
| --- |
| **import java.awt.\*;**  **class Calculator extends Frame**  **{**  **Label lblfno,lblsno,lblrs;**  **TextField txtfno,txtsno,txtrs;**  **Button btnAdd;**  **Panel p1;**  **public Calculator()**  **{**  **lblfno=new Label("Enter the First No:");**  **lblfno.setBounds( 10, 50, 150, 20 );**  **lblsno=new Label("Enter the Second No:");**  **lblsno.setBounds( 10, 70, 150, 20 );**  **lblrs=new Label("Result:");**  **lblrs.setBounds( 10, 90, 150, 20 );**  **txtfno=new TextField(20);**  **txtfno.setBounds(175, 50, 150, 20);**  **txtsno=new TextField(20);**  **txtsno.setBounds(175, 70, 150, 20);**  **txtrs=new TextField(20);**  **txtrs.setBounds(175, 90, 150, 20);**  **btnAdd=new Button("ADD");**  **btnAdd.setBounds(10,150,100,20);**  **p1=new Panel(null);**  **add(p1);**  **p1.add(lblfno); p1.add(txtfno);**  **p1.add(lblsno); p1.add(txtsno);**  **p1.add(lblrs); p1.add(txtrs);**  **p1.add(btnAdd);**  **}**    **public static void main(String args[])**  **{**  **Calculator c=new Calculator();**    **c.setSize(400, 400);**  **c.setLocation(100,200);**  **c.setVisible(true);**  **c.setLocation(300,100);**  **}**  **}** |

**Choice**

Choice control is used to show pop up menu of choices.

|  |
| --- |
| **Constructor**  **Choice()**  Creates a new choice menu.  Eg:  Choice c1;  c1=new Choice();  **void add(String item)**  Adds an item to this Choice menu.  c1.add(“MCA”);  c1.add(“MBA”);  c1.add(“ME”);  c1.add(“MTech”);  p1.add(c1); |

|  |
| --- |
| **void insert(String item, int index)**  Inserts the item into this choice at the specified position.  **int getSelectedIndex()**  Returns the index of the currently selected item.  **String getSelectedItem()**  Retrieve the selected Item.  **void remove(int position)**  Removes an item from the choice menu at the specified position.  **void remove(String item)**  Removes the first occurrence of item from the Choice menu. |

**List**

The List represents a list of text items. The list can be configured that user can choose either one item or multiple items.

|  |
| --- |
| **Constructor**  **List(int rows, boolean multipleMode)**  Creates a new scrolling list initialized to display the specified number of rows.  Eg:  List l1;  l1=new List(5,true); // 5 is number of visible items, and true for multi-selection and & false for single.  **void add(String item)**  Adds the specified item to the end of scrolling list.  l1.add(“MCA”);  l1.add(“M.Tech”);  l1.add(“MBA”);  l1.add(“MCS”);  p1.add(l1); |

|  |
| --- |
| **void remove(int position)**  Removes the item at the specified position from this scrolling list.  **void remove(String item)**  Removes the first occurrence of an item from the list.  **String getSelectedItem()**  Gets the selected item on this scrolling list.  **String[] getSelectedItems()**  Gets the selected items on this scrolling list. |

**Checkbox**

Checkbox control is used to turn an option on(true) or off(false). There is label for each checkbox representing what the checkbox does. The state of a checkbox can be changed by clicking on it.

|  |
| --- |
| **Constructor**  **Checkbox(String label)**  Creates a check box with the specified label.  Eg:  Checkbox c1,c2;  c1=new Checkbox(“JAVA”);  c2=new Checkbox(“C++”);  p1.add(c1);  p1.add(c2); |

|  |
| --- |
| **String getLabel()**  Gets the label of this check box.  **boolean getState()**  Determines whether this check box is in the **on** or **off** state.  **void setLabel(String label)**  Sets this check box's label to be the string argument.  **void setState(boolean state)**  Sets the state of this check box to the specified state. |

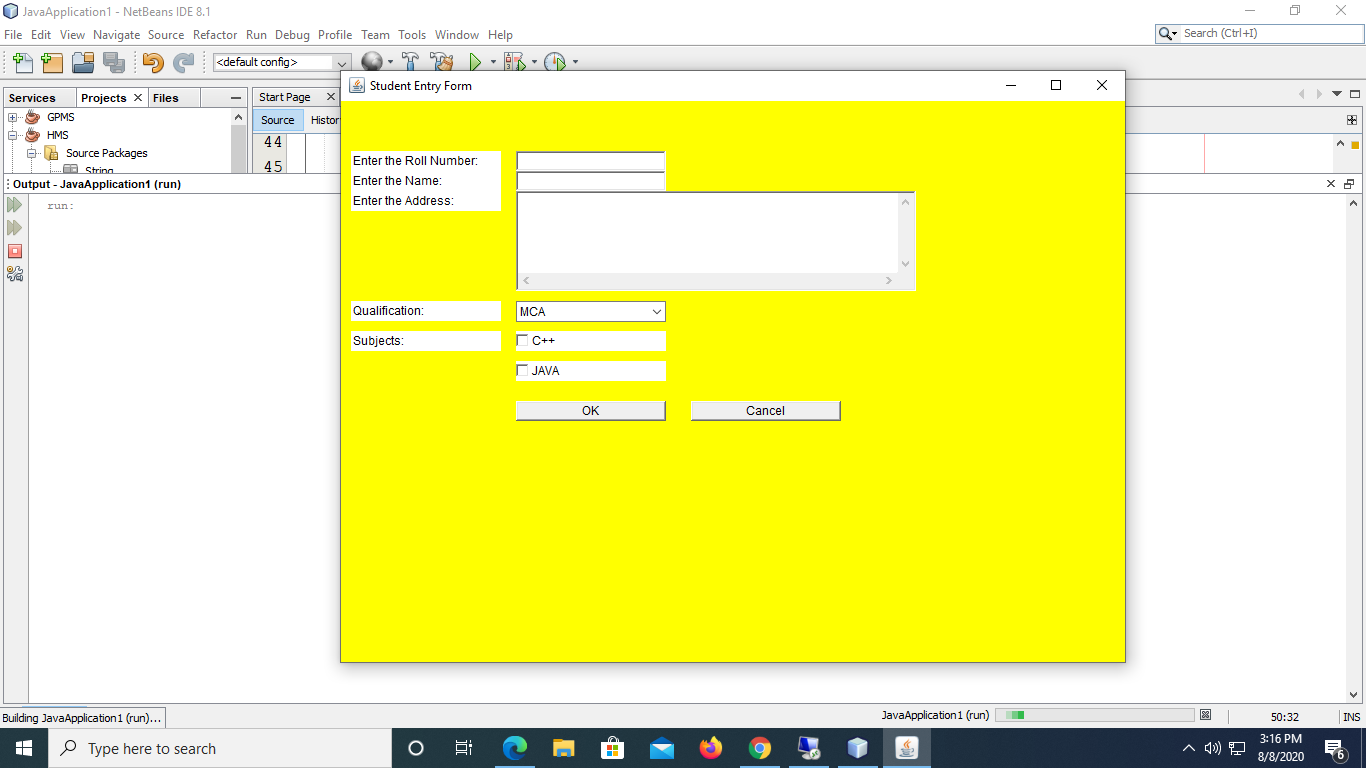
**RadioButton**

The CheckboxGroup class is used to group the set of checkbox.

|  |
| --- |
| CheckboxGroup cg;  Checkbox c1,c2;  cg=new CheckboxGroup();  c1=new Checkbox(“Male”,cg,true);  c2=new Checkbox(“Female”,cg,false);  p1.add(c1);  p1.add(c2); |

|  |
| --- |
| **Checkbox getSelectedCheckbox()**  To get the value of selected radio button we use getSelectedCheckbox() method of CheckboxGroup class.  Eg:  Checkbox chk\_value=cg.getSelectedCheckbox();  String str=chk\_value.getLabel(); |

Example:



import java.awt.\*;

import java.awt.event.\*;

public class StudentForm1 extends Frame{

Label lblRno,lblName,lblAddr,lblQ,lblSubject;

TextField txtRno,txtName;

TextArea txtAddr;

Button btnOk,btnCancel;

Choice c1;

Checkbox chk1,chk2;

Panel p1;

public StudentForm1()

{

super("Student Entry Form");

lblRno=new Label("Enter the Roll Number:");

lblRno.setBounds(10,50,150,20);

lblName=new Label("Enter the Name:");

lblName.setBounds(10,70,150,20);

lblAddr=new Label("Enter the Address:");

lblAddr.setBounds(10,90,150,20);

lblQ=new Label("Qualification:");

lblQ.setBounds(10,200,150,20);

lblSubject=new Label("Subjects:");

lblSubject.setBounds(10,230,150,20);

txtRno=new TextField(20);

txtRno.setBounds(175,50,150,20);

txtName=new TextField(20);

txtName.setBounds(175,70,150,20);

txtAddr=new TextArea(5,4);

txtAddr.setBounds(175,90,400,100);

c1=new Choice();

c1.add("MCA");

c1.add("MCS");

c1.add("MTech");

c1.add("MBA");

c1.add("BBA");

c1.add("B.Tech");

c1.setBounds(175,200,150,20);

chk1=new Checkbox("C++");

chk1.setBounds(175,230,150,20);

chk2=new Checkbox("JAVA");

chk2.setBounds(175,260,150,20);

btnOk=new Button("OK");

btnOk.setBounds(175,300,150,20);

btnCancel=new Button("Cancel");

btnCancel.setBounds(350,300,150,20);

p1=new Panel(null);

add(p1);

p1.add(lblRno); p1.add(txtRno);

p1.add(lblName); p1.add(txtName);

p1.add(lblAddr); p1.add(txtAddr);

p1.add(lblSubject); p1.add(lblQ); p1.add(c1);

p1.add(chk1); p1.add(chk2);

p1.add(btnOk);p1.add(btnCancel);

setSize(800, 600);

setVisible(true);

p1.setBackground(Color.yellow);

}

public static void main(String[] args) {

StudentForm1 st=new StudentForm1();

}

}

**What is an Event?**

Changing the state of an object is known as event.

Eg: click on button, dragging mouse, entering a character through keyboard, selecting an item from list etc.

**Types of Event**

The events can be broadly classified into two categories:

* **Foreground Events -** Those events which require the direct interaction of user i.e user interacting with the Graphical User Interface (GUI). For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.
* **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

## What is Event Handling?

* Event Handling is the mechanism that controls the event and decides what should happen if an event occurs.
* These mechanisms have the code which is known as event handler that is executed when an event occurs.
* Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.

The Delegation Event Model has the following key participants namely:

* **Source -** The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler i.e type of event, source of event etc.
* **Listener -** It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received, the listener processes the event.

Types of Event Classes

|  |  |
| --- | --- |
| **Event Classes** | **Description** |
| ActionEvent | This event gets generated when you activate any control i.e click on Button, MenuItem etc |
| ItemEvent | This event gets generated when you change the state of Checkbox, Choice, List etc. |
| KeyEvent | This event gets generated when you press any key on keyboard. |
| MoueseEvent | This event gets generated when you press the mouse, release the mouse, drag the mouse, move the mouse etc. |
| WindowEvent | This event gets generated when you click on close, maximize or minimize button on Frame. |
| AdjustmentEvent | This event gets generated when you change the value of scrollbar. |

To handle these events, predefined methods are available in particular interfaces. For each event class one particular interface is there which contains method to handle the event.

**Event Listener Interfaces**

* Class hierarchy and methods
  + java.util.**EventListener**
    - java.awt.event.**ActionListener**
      * actionPerformed
    - java.awt.event.**AdjustmentListener**
      * adjustmentValueChanged
    - java.awt.event.**ComponentListener**
      * componentHidden, componentMoved, componentResized, componentShown
    - java.awt.event.**FocusListener**
      * focusGained, focusLost
    - java.awt.event.**ItemListener**
      * itemStateChanged
    - java.awt.event.**KeyListener**
      * keyPressed, keyReleased, keyTyped
    - java.awt.event.**MouseListener**
      * mouseEntered, mouseExited,

mousePressed, mouseReleased, mouseClicked

* + - java.awt.event.**MouseMotionListener**
      * mouseDragged, mouseMoved
    - java.awt.event.**TextListener**
      * textValueChanged
    - java.awt.event.**WindowListener**

windowOpened, windowClosing, windowClosed, windowActivated, windowDeactivated, windowIconified, windowDeiconified

**Event Sources and Their Listeners Interface**

* Frame - WindowListener
* Button - ActionListener
* Choice - ItemListener
* Checkbox - ItemListener
* CheckboxMenuItem - ItemListener

### List - ItemListener, ActionListener

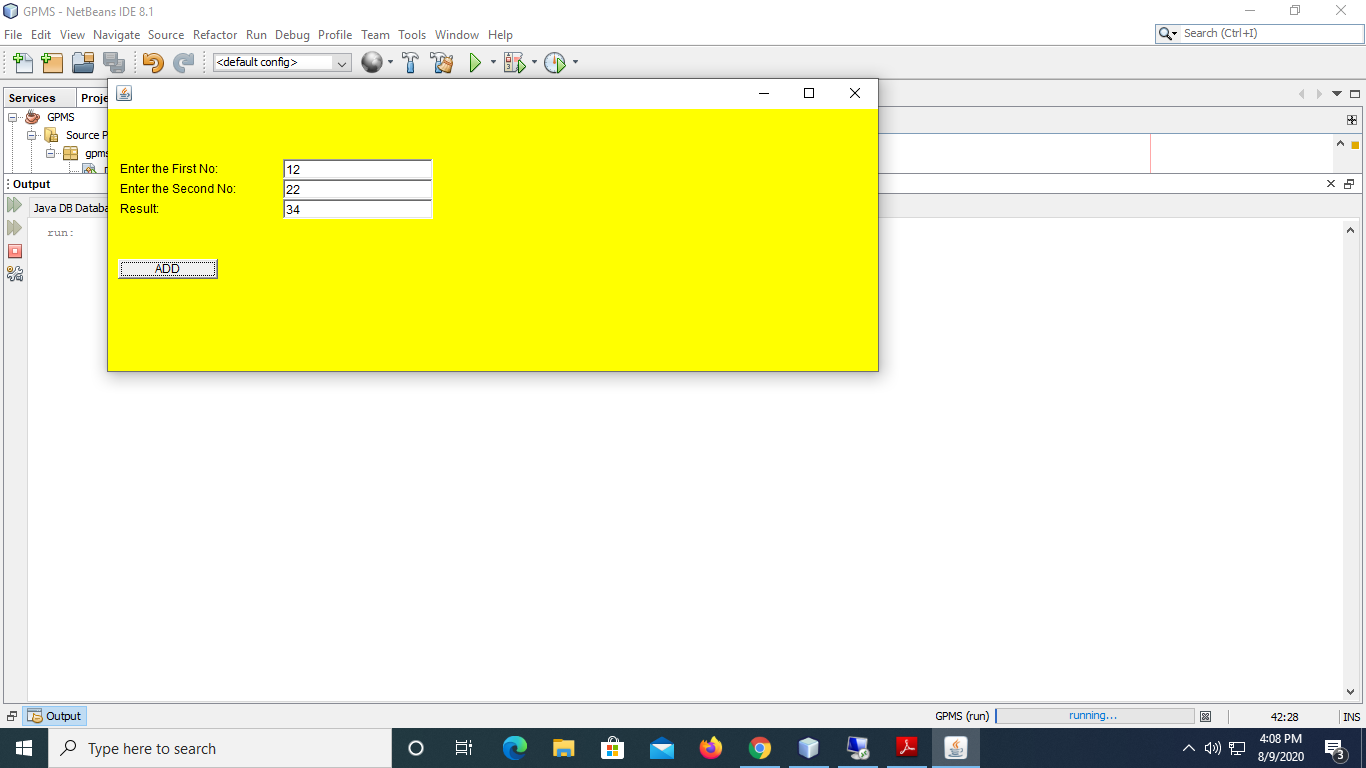
when an item is double-clicked

* MenuItem - ActionListener
* Scrollbar - AdjustmentListener
* TextField - ActionListener, TextListener
* TextArea - TextListener

**Steps to handle the Event**

* 1. Import the package i.e import java.awt.event.\*;
  2. Implement the particular interface in your class.
  3. Register the event on the component on which you want to generate the event.
  4. Add the Event handler method.

**Q: WAP to create two textfields and one button for sum of two numbers. If user clicks button, sum of two input values is displayed on the result textfield.**

****

**import java.awt.\*;**

**import java.awt.event.\*;**

**class Calculator extends Frame implements ActionListener**

**{**

**Label lblfno,lblsno,lblrs;**

**TextField txtfno,txtsno,txtrs;**

**Button btnAdd;**

**Panel p1;**

**public Calculator()**

**{**

**lblfno=new Label("Enter the First No:");**

**lblfno.setBounds( 10, 50, 150, 20 );**

**lblsno=new Label("Enter the Second No:");**

**lblsno.setBounds( 10, 70, 150, 20 );**

**lblrs=new Label("Result:");**

**lblrs.setBounds( 10, 90, 150, 20 );**

**txtfno=new TextField(20);**

**txtfno.setBounds(175, 50, 150, 20);**

**txtsno=new TextField(20);**

**txtsno.setBounds(175, 70, 150, 20);**

**txtrs=new TextField(20);**

**txtrs.setBounds(175, 90, 150, 20);**

**btnAdd=new Button("ADD");**

**btnAdd.setBounds(10,150,100,20);**

**p1=new Panel(null);**

**add(p1);**

**p1.add(lblfno); p1.add(txtfno);**

**p1.add(lblsno); p1.add(txtsno);**

**p1.add(lblrs); p1.add(txtrs);**

**p1.add(btnAdd);**

**//register listener**

**btnAdd.addActionListener(this); //passing current instance**

**p1.setBackground(Color.yellow);**

**}**

**public void actionPerformed(ActionEvent evt)**

**{**

**Object obj=evt.getSource();**

**int a,b,rs;**

**a=Integer.parseInt(txtfno.getText());**

**b=Integer.parseInt(txtsno.getText());**

**if(obj==btnAdd)**

**{**

**rs=a+b;**

**txtrs.setText(String.valueOf(rs));**

**}**

**}**

**}**

**public class Test {**

**public static void main(String args[])**

**{**

**Calculator c=new Calculator();**

**c.setSize(400, 400);**

**c.setLocation(100,200);**

**c.setVisible(true);**

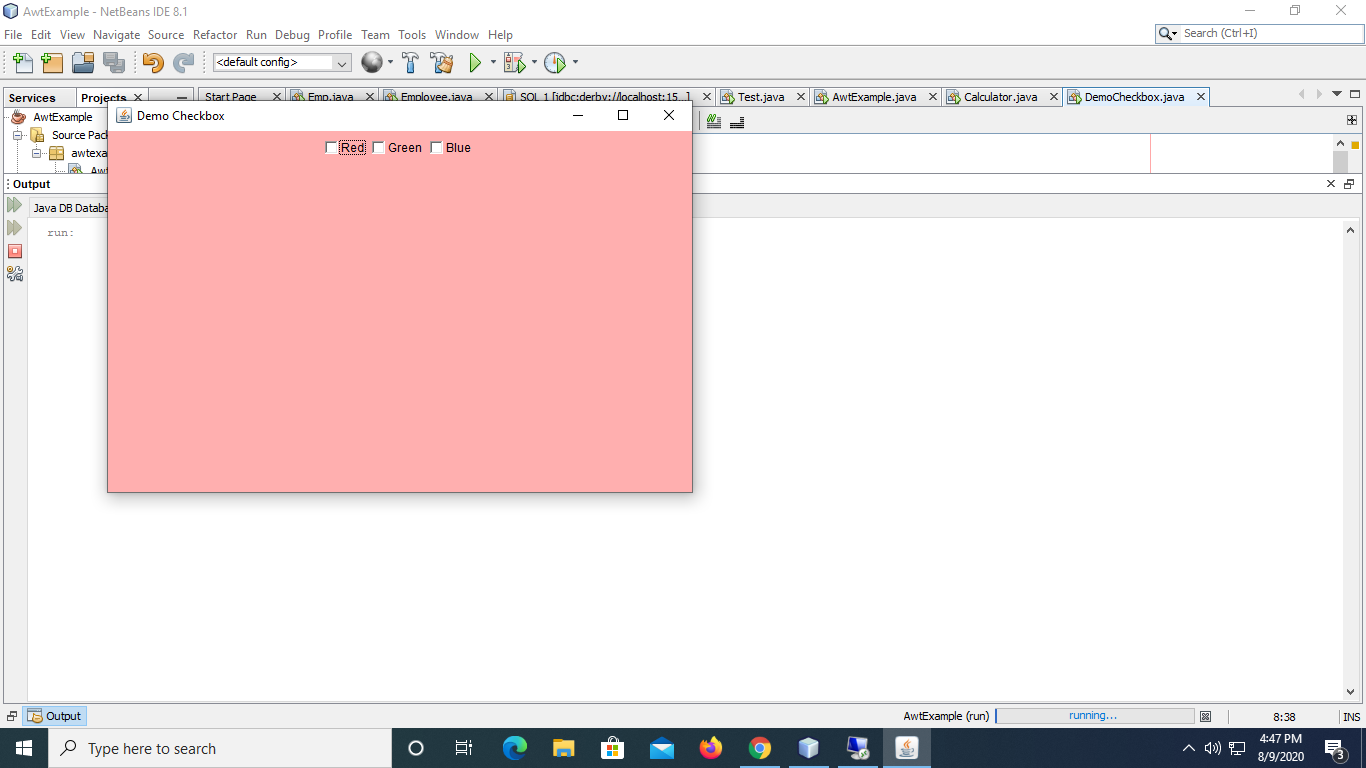
**c.setLayout(null);**

**c.setBackground(Color.yellow);**

**}**

**}**

**Q: Demonstrate the use of Checkbox Control in Java AWT.**

****

**import java.awt.\*;**

**import java.awt.event.\*;**

**public class DemoCheckbox extends Frame implements ItemListener{**

**Checkbox chkRed,chkGreen,chkBlue;**

**Panel p1;**

**public DemoCheckbox()**

**{**

**super("Demo Checkbox");**

**chkRed=new Checkbox("Red");**

**chkGreen=new Checkbox("Green");**

**chkBlue=new Checkbox("Blue");**

**p1=new Panel();**

**add(p1);**

**p1.add(chkRed);**

**p1.add(chkGreen);**

**p1.add(chkBlue);**

**chkRed.addItemListener(this);**

**chkGreen.addItemListener(this);**

**chkBlue.addItemListener(this);**

**p1.setBackground(Color.PINK);**

**}**

**public void itemStateChanged(ItemEvent evt)**

**{**

**Object obj=evt.getSource();**

**if(obj==chkRed)**

**{**

**p1.setBackground(Color.red);**

**chkGreen.setState(false);**

**chkBlue.setState(false);**

**}**

**else if(obj==chkGreen)**

**{**

**p1.setBackground(Color.green);**

**chkRed.setState(false);**

**chkBlue.setState(false);**

**}**

**else if(obj==chkBlue)**

**{**

**p1.setBackground(Color.blue);**

**chkGreen.setState(false);**

**chkRed.setState(false);**

**}**

**}**

**public static void main(String[] args) {**

**DemoCheckbox d=new DemoCheckbox();**

**d.setSize(600, 400);**

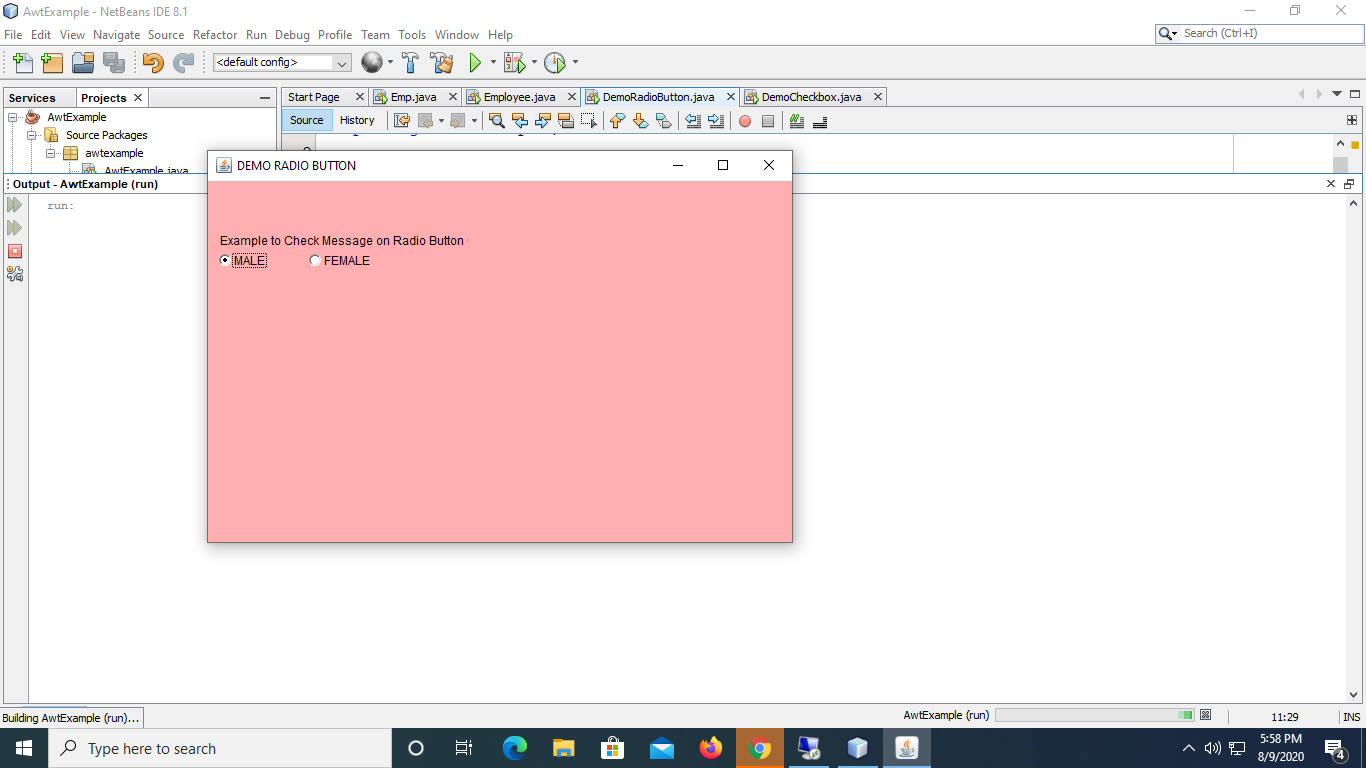
**d.setLocation(100, 100);**

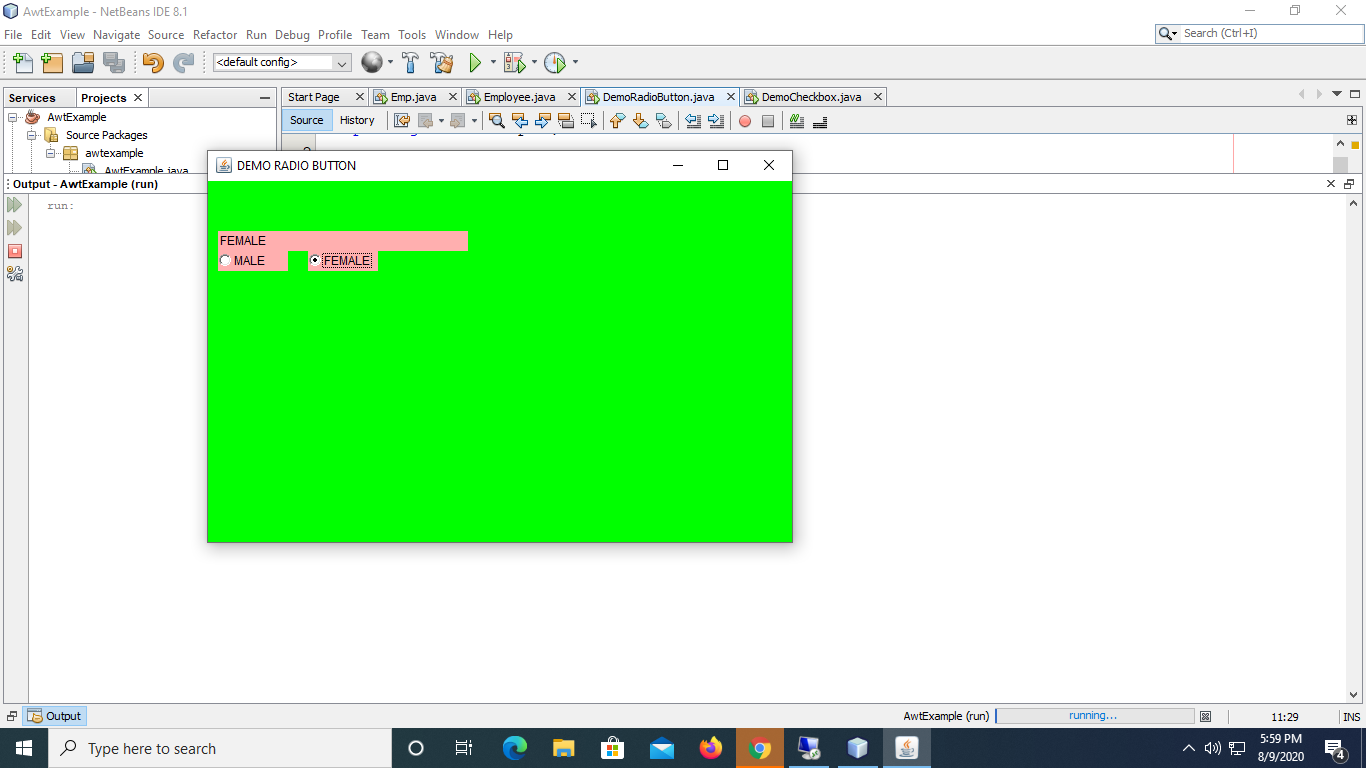
**d.setVisible(true);**

**}**

**}**

**Q: Demonstrate the use of RadioButton in Java AWT.**

****



import java.awt.\*;

import java.awt.event.\*;

public class DemoRadioButton extends Frame implements ItemListener{

CheckboxGroup cg;

Checkbox chkMale,chkFemale;

Label lblmsg;

Panel p1;

public DemoRadioButton()

{

super("DEMO RADIO BUTTON");

lblmsg=new Label("Example to Check Message on Radio Button Click");

lblmsg.setBounds(10,50,250,20);

cg=new CheckboxGroup();

chkMale=new Checkbox("MALE",cg,true);

chkMale.setBounds(10,70,70,20);

chkFemale=new Checkbox("FEMALE",cg,false);

chkFemale.setBounds(100,70,70,20);

p1=new Panel(null);

add(p1);

p1.add(lblmsg);

p1.add(chkMale);

p1.add(chkFemale);

chkMale.addItemListener(this);

chkFemale.addItemListener(this);

p1.setBackground(Color.PINK);

}

public void itemStateChanged(ItemEvent evt)

{

Object obj=evt.getSource();

Checkbox chk\_label;

if(obj==chkMale)

{

p1.setBackground(Color.red);

chk\_label=cg.getSelectedCheckbox();

lblmsg.setText(chk\_label.getLabel());

}

else if(obj==chkFemale)

{

p1.setBackground(Color.green);

chk\_label=cg.getSelectedCheckbox();

lblmsg.setText(chk\_label.getLabel());

}

}

public static void main(String[] args) {

DemoRadioButton d=new DemoRadioButton();

d.setSize(600, 400);

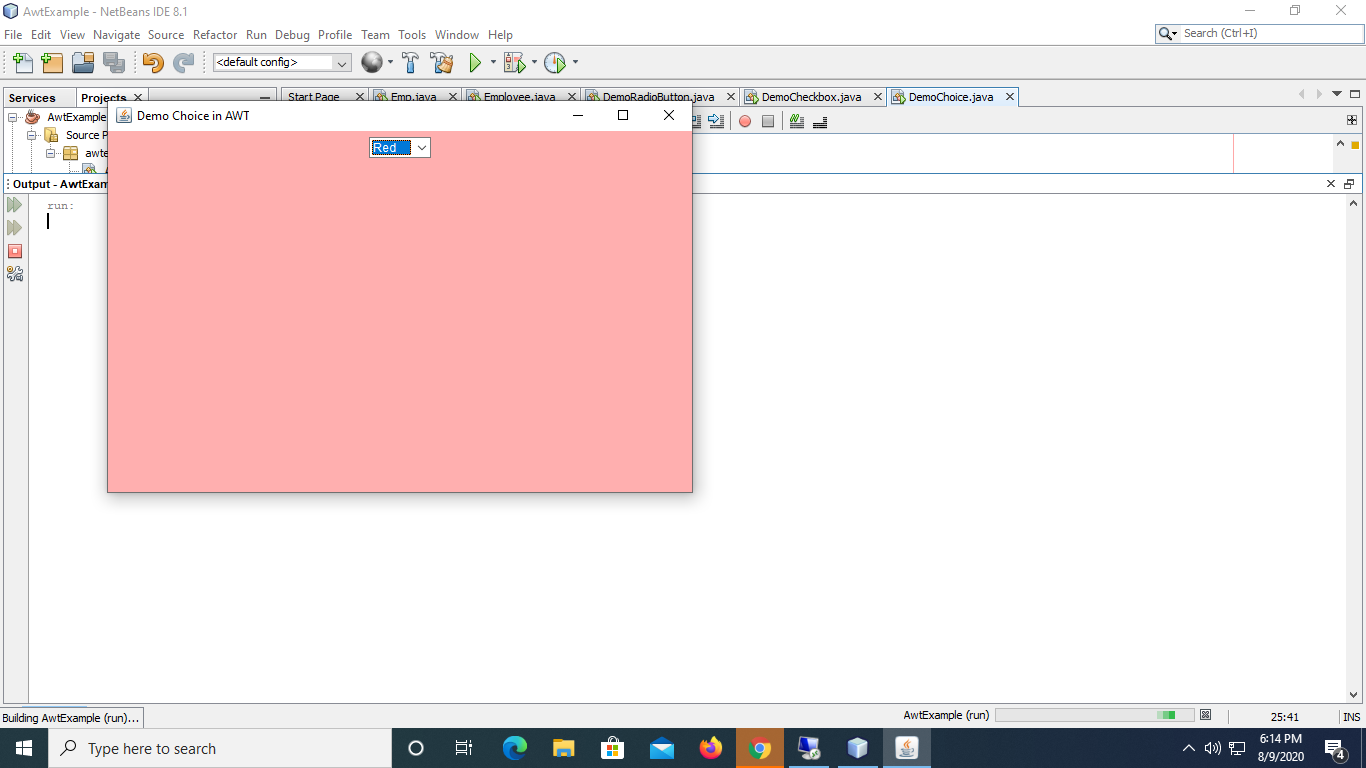
d.setVisible(true);

d.setLocation(200, 150);

}

}

**Q: Demonstrate the use of Checkbox Control in Java AWT.**



import java.awt.\*;

import java.awt.event.\*;

public class DemoChoice extends Frame implements ItemListener{

Choice chk\_Color;

Panel p1;

public DemoChoice()

{

super("Demo Choice in AWT");

chk\_Color=new Choice();

chk\_Color.add("Red");

chk\_Color.add("Green");

chk\_Color.add("Blue");

p1=new Panel();

add(p1);

p1.add(chk\_Color);

chk\_Color.addItemListener(this);

p1.setBackground(Color.pink);

}

public void itemStateChanged(ItemEvent evt)

{

Object obj=evt.getSource();

if(obj==chk\_Color)

{

String str=chk\_Color.getSelectedItem();

if(str.equals("Red"))

{

p1.setBackground(Color.red);

}

else if(str.equals("Green"))

{

p1.setBackground(Color.green);

}

else if(str.equals("Blue"))

{

p1.setBackground(Color.blue);

}

}

}

public static void main(String[] args) {

DemoChoice d=new DemoChoice();

d.setSize(600, 400);

d.setLocation(100, 100);

d.setVisible(true);

}

}

**Adapter class**

The Adapter class provides the default modification of all methods of an interface; we don't need to modify all the methods of the interface so we can say it reduces coding burden. Sometimes or often we need a few methods of an interface. For that the Adapter class is very helpful since it already modifies all the methods of an interface and by implementing the Adapter class, we only need to modify the required methods.

**The following examples contain the following Adapter classes:**

* ContainerAdapter class.
* KeyAdapter class.
* FocusAdapter class.
* WindowAdapter class.
* MouseAdapter class.
* ComponentAdapter class.
* MouseMotionAdapter class.

Examples

The frame created by the programmer and displayed on monitor comes with title bar and three icons(buttons) on the title bar

1. Minimize button
2. Maximize button
3. Close button

The first two button works implicitly and the close button does not work and requires extra code to close the Frame.

There are three different styles exist to close the Frame are:

1. WindowListener
2. WindowAdaptor
3. Anonymous inner class

**Example of WindowAdapter class**

import java.awt.\*;

import java.awt.event.\*;

class windowAdapter1 extends Frame

{

public windowAdapter1()

{

super("MyFrame");

//addWindowListener(new myClass()); or

myClass m=new myClass();

addWindowListener(m);

setSize(300,400);

setVisible(true);

}

public static void main(String args[])

{

new windowAdapter1();

}

}

class myClass extends WindowAdapter

{

public void windowClosing(WindowEvent evt)

{

System.exit(0);

}

}

**Anonymous class**

If you write a class without name then it’s known as Anonymous class.

import java.awt.\*;

import java.awt.event.\*;

class myFrame extends Frame

{

public myFrame()

{

super("My App");

**addWindowListener(new WindowAdapter()**

**{**

**public void windowClosing(WindowEvent evt)**

**{**

**System.exit(0);**

**}**

**});**

setSize(300,400);

setVisible(true);

}

public static void main(String args[])

{

new myFrame();

}

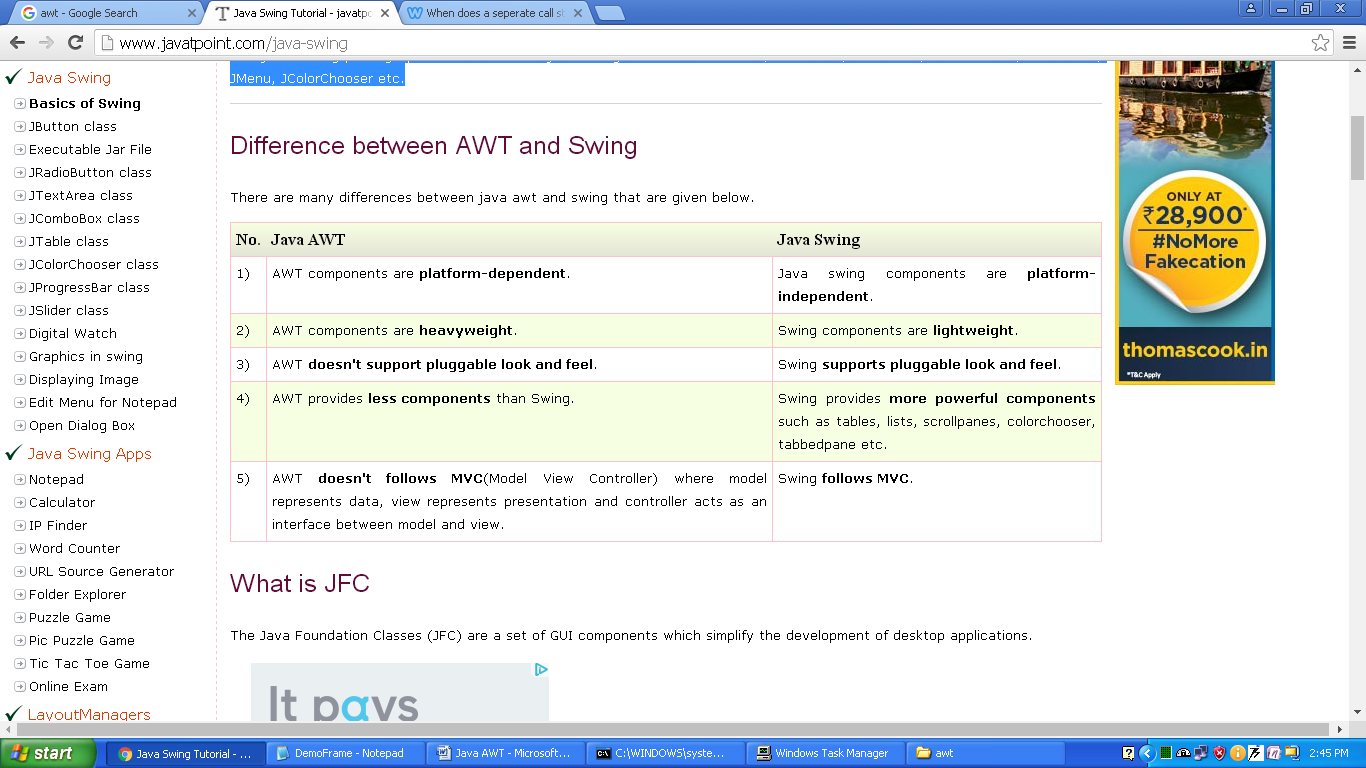
}

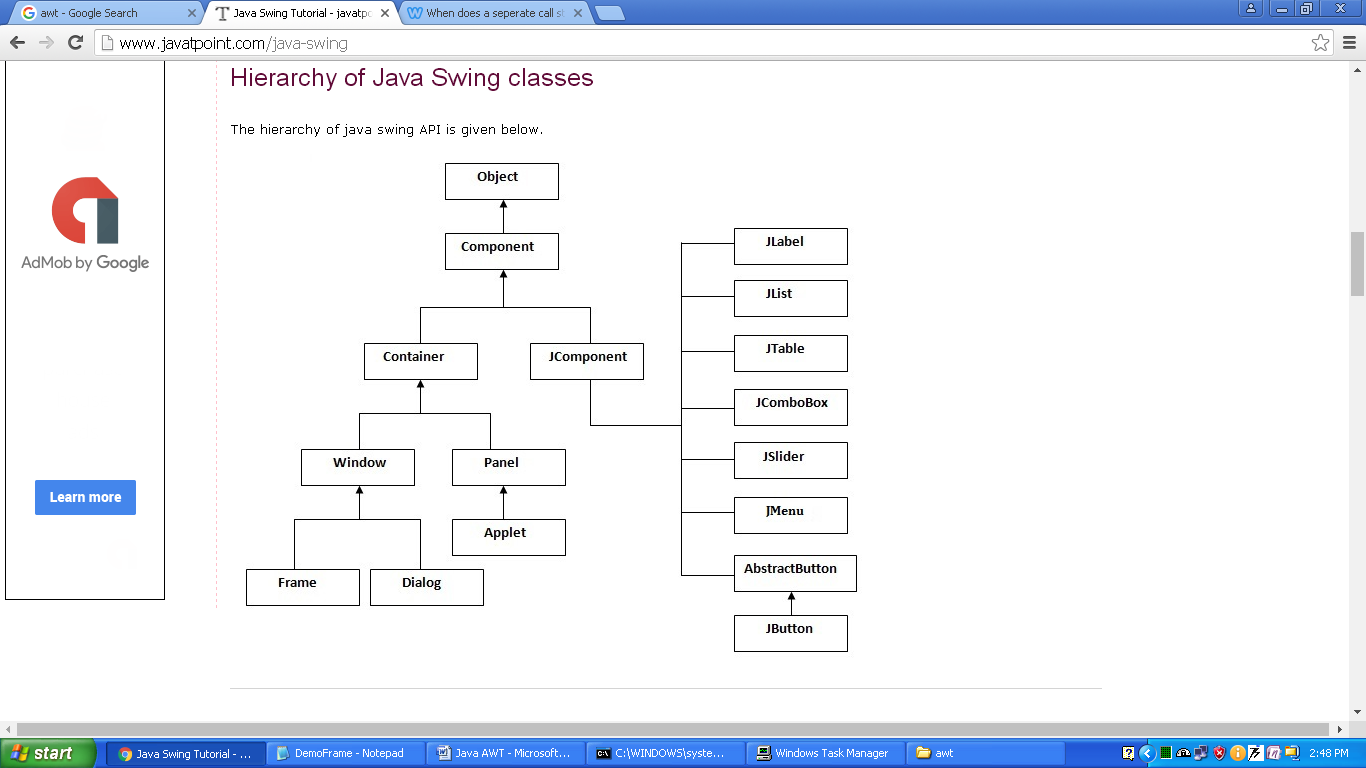
**Java Swing**

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT API.

Unlike AWT, Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.





Java Swing Examples

**There are two ways to create a frame:**

1. By extending Frame class (inheritance)
2. By creating the object of Frame class (association)

**Example: By extending Frame class (inheritance)**

import javax.swing.\*;

public class Simple2 extends JFrame{//inheriting JFrame

JFrame f;

Simple2(){

JButton b=new JButton("click");//create button

b.setBounds(130,100,100, 40);

add(b);//adding button on frame

setSize(400,500);

setLayout(null);

setVisible(true);

}

public static void main(String[] args) {

new Simple2();

}}

**Example: By creating the object of Frame class**

import javax.swing.\*;

public class FirstSwingExample {

public static void main(String[] args) {

JFrame f=new JFrame();//creating instance of JFrame

JButton b=new JButton("click");//creating instance of JButton

b.setBounds(130,100,100, 40);//x axis, y axis, width, height

f.add(b);//adding button in JFrame

f.setSize(400,500);//400 width and 500 height

f.setLayout(null);//using no layout managers

f.setVisible(true);//making the frame visible

}

}

**JCheckBox**

JCheckBox chkJava,chkC,chkDotNet;

chkJava=new JCheckBox(“Java”);

chkC=new JCheckBox(“C”);

chkDotNet=new JCheckBox(“DotNet”);

p1.add(chkJava);

p1.add(chkC);

p1.add(chkDotNet);

**JComboBox**

JComboBox drpLanguage;

drpLanguage=new JComboBox();

drpLanguage.add(“English”);  
drpLanguage.add(“French”);

drpLanguage.add(“Hindi”);

drpLanguage.add(“Marathi”);

p1.add(drpLanguage);

**JRadioButton**

JRadioButton rdoMale,rdoFemale;

rdoMale=new JRadioButton(“Male”,true);

rdoFemale=new JRadioButton(“Female”,false);

p1.add(rdoMale);

p1.add(rdoFemale);

**JPasswordField**

JPasswordField pwd;

pwd=new JPasswordField(20);

pwd.setEchoChar(‘\*’);

p1.add(pwd);

Q: Write a AWT program to calculates factorial values for an integer number,

while the input is taken from an Factorial form.

Enter the Number to find the fact:

Find Factorial

Result: