**Lab Manual**

**Java AWT**

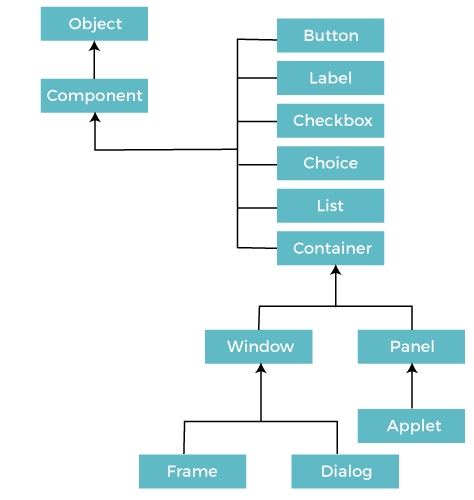
**Java AWT** (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt pakage provides classes for AWT API such as Textfield, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

**Java AWT Hierarchy**

The hierarchy of Java AWT classes are given below.



**Components**

All the elements like the button, text fields, scroll bars, etc. are called components. In Java AWT, there are classes for each component as shown in above diagram. In order to place every component in a particular position on a screen, we need to add them to a container.

**Container**

The Container is a component in AWT that can contain another components like [buttons](https://www.javatpoint.com/java-awt-button), textfields, labels etc. The classes that extends Container class are known as container such as **Frame, Dialog** and **Panel**.

It is basically a screen where the where the components are placed at their specific locations. Thus it contains and controls the layout of components.

**Types of containers:**

There are four types of containers in Java AWT:

Window

Panel

Frame

Dialog

**Window**

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window. We need to create an instance of Window class to create this container.

**Panel**

The Panel is the container that doesn't contain title bar, border or menu bar. It is generic container for holding the components. It can have other components like button, text field etc. An instance of Panel class creates a container, in which we can add components.

**Frame**

The Frame is the container that contain title bar and border and can have menu bars. It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.

**Useful Methods of Component Class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void add(Component c) | Inserts a component on this component. |
| public void setSize(int width,int height) | Sets the size (width and height) of the component. |
| public void setLayout(LayoutManager m) | Defines the layout manager for the component. |
| public void setVisible(boolean status) | Changes the visibility of the component, by default false. |

**Java AWT Example**

To create simple AWT example, you need a frame. There are two ways to create a GUI using Frame in AWT.

By extending Frame class (**inheritance**)

By creating the object of Frame class (**association**)

**AWT Example by Inheritance**

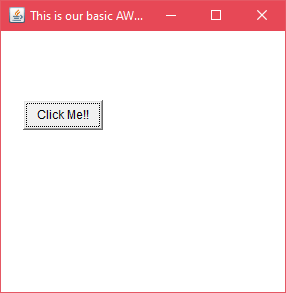
Let's see a simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

**AWTExample1.java**

1. // importing Java AWT class
2. **import** java.awt.\*;
4. // extending Frame class to our class AWTExample1
5. **public** **class** AWTExample1 **extends** Frame {
7. // initializing using constructor
8. AWTExample1() {
10. // creating a button
11. Button b = **new** Button("Click Me!!");
13. // setting button position on screen
14. b.setBounds(30,100,80,30);
16. // adding button into frame
17. add(b);
19. // frame size 300 width and 300 height
20. setSize(300,300);
22. // setting the title of Frame
23. setTitle("This is our basic AWT example");
25. // no layout manager
26. setLayout(**null**);
28. // now frame will be visible, by default it is not visible
29. setVisible(**true**);
30. }
32. // main method
33. **public** **static** **void** main(String args[]) {
35. // creating instance of Frame class
36. AWTExample1 f = **new** AWTExample1();
38. }
40. }

The setBounds(int x-axis, int y-axis, int width, int height) method is used in the above example that sets the position of the awt button.

**Output:**



**AWT Example by Association**

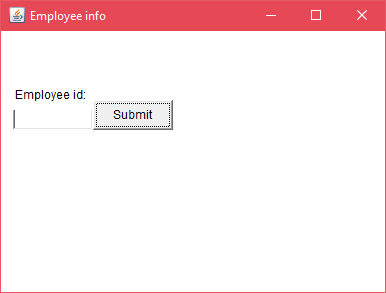
Let's see a simple example of AWT where we are creating instance of Frame class. Here, we are creating a TextField, Label and Button component on the Frame.

AD

**AWTExample2.java**

1. // importing Java AWT class
2. **import** java.awt.\*;
4. // class AWTExample2 directly creates instance of Frame class
5. **class** AWTExample2 {
7. // initializing using constructor
8. AWTExample2() {
10. // creating a Frame
11. Frame f = **new** Frame();
13. // creating a Label
14. Label l = **new** Label("Employee id:");
16. // creating a Button
17. Button b = **new** Button("Submit");
19. // creating a TextField
20. TextField t = **new** TextField();
22. // setting position of above components in the frame
23. l.setBounds(20, 80, 80, 30);
24. t.setBounds(20, 100, 80, 30);
25. b.setBounds(100, 100, 80, 30);
27. // adding components into frame
28. f.add(b);
29. f.add(l);
30. f.add(t);
32. // frame size 300 width and 300 height
33. f.setSize(400,300);
35. // setting the title of frame
36. f.setTitle("Employee info");
38. // no layout
39. f.setLayout(**null**);
41. // setting visibility of frame
42. f.setVisible(**true**);
43. }
45. // main method
46. **public** **static** **void** main(String args[]) {
48. // creating instance of Frame class
49. AWTExample2 awt\_obj = **new** AWTExample2();
51. }
53. }

**Output:**



**Java Swing:**

**Java Swing tutorial** is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java 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 JFrame**

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

**JFrame Example**

1. **import** java.awt.FlowLayout;
2. **import** javax.swing.JButton;
3. **import** javax.swing.JFrame;
4. **import** javax.swing.JLabel;
5. **import** javax.swing.JPanel;
6. **public** **class** JFrameExample {
7. **public** **static** **void** main(String s[]) {
8. JFrame frame = **new** JFrame("JFrame Example");
9. JPanel panel = **new** JPanel();
10. panel.setLayout(**new** FlowLayout());
11. JLabel label = **new** JLabel("JFrame By Example");
12. JButton button = **new** JButton();
13. button.setText("Button");
14. panel.add(label);
15. panel.add(button);
16. frame.add(panel);
17. frame.setSize(200, 300);
18. frame.setLocationRelativeTo(**null**);
19. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
20. frame.setVisible(**true**);
21. }
22. }



**Difference between AWT and Swing**

There are many differences between java awt and swing that are given below.

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

**Hierarchy of Java Swing classes**



**Commonly used Methods of Component class**

The methods of Component class are widely used in java swing that are given below.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void add(Component c) | add a component on another component. |
| public void setSize(int width,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. |

**Java Swing Examples**

There are two ways to create a frame:

By creating the object of Frame class (association)

By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

**Java Code Example for Button Click Events**

The Java code example below displays the number of clicks a user has so far made when they click **Button1**:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class ClicksCount implements ActionListener{

int count = 0;// store number of clicks

ClicksCount(){

JFrame frame = new JFrame();

JButton button1 = new JButton("Button1");

JButton button2 = new JButton("Button2");

button1.addActionListener(this);

frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y\_AXIS));

frame.add(button1);

frame.add(button2);

frame.getRootPane().setDefaultButton(button1); // sets default button

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(450,450);

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

count++;

System.out.println("You have clicked the ACTIVE button " + count + " times");

}

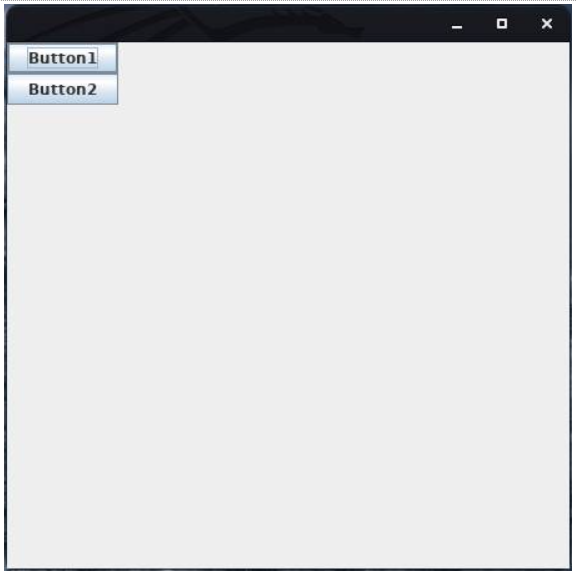
public static void main(String args[]){

ClicksCount Clicks = new ClicksCount();

}

}

When you compile and run the code above, you should see two buttons. If you take good notice, you will observe that **Button1** has been highlighted. This is because it has been set as the *default* button:



The *default* button is the button that initially appears to have the focus when the program is first run. When you press **Enter** on your keyboard, the program clicks this button since it was already selected by default. Pressing **Tab** will shift focus to the other button.

You can only have, at most, one default button, and you set it by calling the **setDefaultButton()** method on the root pane of a top-level container.

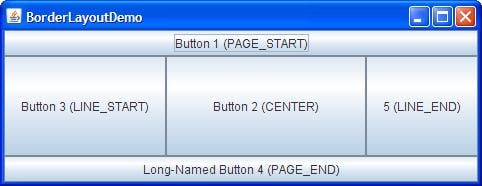
If you click **Button2** in this example, you will notice that there is not a message displayed. This is because no event handler has been registered to listen for events on this button. In other words, you would have to use the **addActionListener()** method with **Button2** to ensure that **actionPerformed(ActionEvent e)** is called when it is clicked.

**Final Thoughts on Buttons and Events in Java**

Since you are dealing with Swing components when using buttons and **JButtons**, remember to import the **javax.swing** library into your Java code. Also, in order to use an event listener, you need to add the **java.awt** library, as shown in the last code example. If you do not include these libraries, you will get a compilation error.

**Java BorderLayout**

A BorderLayout places components in up to five areas: top, bottom, left, right, and center. It is the default layout manager for every java JFrame



Java BorderLayout

**Java FlowLayout**

FlowLayout is the default layout manager for every jPanel. It simply lays out components in a single row one after the other.



**Java FlowLayout**

**Java GridBagLayout**

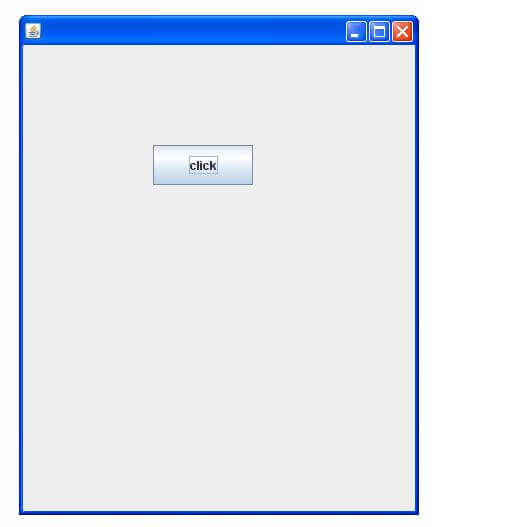
It is the more sophisticated of all layouts. It aligns components by placing them within a grid of cells, allowing components to span more than one cell.



**Java GridBagLayout**

*File: FirstSwingExample.java*

1. **import** javax.swing.\*;
2. **public** **class** FirstSwingExample {
3. **public** **static** **void** main(String[] args) {
4. JFrame f=**new** JFrame();//creating instance of JFrame
6. JButton b=**new** JButton("click");//creating instance of JButton
7. b.setBounds(130,100,100, 40);//x axis, y axis, width, height
9. f.add(b);//adding button in JFrame
11. f.setSize(400,500);//400 width and 500 height
12. f.setLayout(**null**);//using no layout managers
13. f.setVisible(**true**);//making the frame visible
14. }
15. }



**Example of Swing by Association inside constructor**

We can also write all the codes of creating JFrame, JButton and method call inside the java constructor.

*File: Simple.java*

1. **import** javax.swing.\*;
2. **public** **class** Simple {
3. JFrame f;
4. Simple(){
5. f=**new** JFrame();//creating instance of JFrame
7. JButton b=**new** JButton("click");//creating instance of JButton
8. b.setBounds(130,100,100, 40);
10. f.add(b);//adding button in JFrame
12. f.setSize(400,500);//400 width and 500 height
13. f.setLayout(**null**);//using no layout managers
14. f.setVisible(**true**);//making the frame visible
15. }
17. **public** **static** **void** main(String[] args) {
18. **new** Simple();
19. }
20. }

The setBounds(int xaxis, int yaxis, int width, int height)is used in the above example that sets the position of the button.

**Example of Swing by inheritance**

We can also inherit the JFrame class, so there is no need to create the instance of JFrame class explicitly.

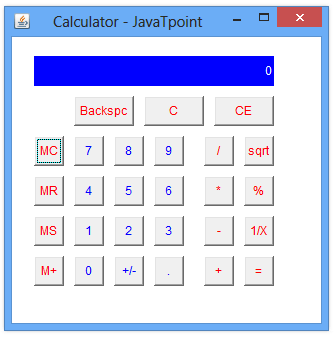
*File: Simple2.java*

1. **import** javax.swing.\*;
2. **public** **class** Simple2 **extends** JFrame{//inheriting JFrame
3. JFrame f;
4. Simple2(){
5. JButton b=**new** JButton("click");//create button
6. b.setBounds(130,100,100, 40);
8. add(b);//adding button on frame
9. setSize(400,500);
10. setLayout(**null**);
11. setVisible(**true**);
12. }
13. **public** **static** **void** main(String[] args) {
14. **new** Simple2();
15. }}

**Calculator in Java with Source Code**

**Calculator in Java with Source Code:** We can develop calculator in java with the help of AWT/Swing with event handling. Let's see the code of creating calculator in java.

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. Save this file as MyCalculator.java
3. to compile it use
4. javac MyCalculator.java
5. to use the calcuator do this
6. java MyCalculator
8. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
9. **import** java.awt.\*;
10. **import** java.awt.event.\*;
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. **public** **class** MyCalculator **extends** Frame
14. {
16. **public** **boolean** setClear=**true**;
17. **double** number, memValue;
18. **char** op;
20. String digitButtonText[] = {"7", "8", "9", "4", "5", "6", "1", "2", "3", "0", "+/-", "." };
21. String operatorButtonText[] = {"/", "sqrt", "\*", "%", "-", "1/X", "+", "=" };
22. String memoryButtonText[] = {"MC", "MR", "MS", "M+" };
23. String specialButtonText[] = {"Backspc", "C", "CE" };
25. MyDigitButton digitButton[]=**new** MyDigitButton[digitButtonText.length];
26. MyOperatorButton operatorButton[]=**new** MyOperatorButton[operatorButtonText.length];
27. MyMemoryButton memoryButton[]=**new** MyMemoryButton[memoryButtonText.length];
28. MySpecialButton specialButton[]=**new** MySpecialButton[specialButtonText.length];
30. Label displayLabel=**new** Label("0",Label.RIGHT);
31. Label memLabel=**new** Label(" ",Label.RIGHT);
33. **final** **int** FRAME\_WIDTH=325,FRAME\_HEIGHT=325;
34. **final** **int** HEIGHT=30, WIDTH=30, H\_SPACE=10,V\_SPACE=10;
35. **final** **int** TOPX=30, TOPY=50;
36. ///////////////////////////
37. MyCalculator(String frameText)//constructor
38. {
39. **super**(frameText);
41. **int** tempX=TOPX, y=TOPY;
42. displayLabel.setBounds(tempX,y,240,HEIGHT);
43. displayLabel.setBackground(Color.BLUE);
44. displayLabel.setForeground(Color.WHITE);
45. add(displayLabel);
47. memLabel.setBounds(TOPX,  TOPY+HEIGHT+ V\_SPACE,WIDTH, HEIGHT);
48. add(memLabel);
50. // set Co-ordinates for Memory Buttons
51. tempX=TOPX;
52. y=TOPY+2\*(HEIGHT+V\_SPACE);
53. **for**(**int** i=0; i<memoryButton.length; i++)
54. {
55. memoryButton[i]=**new** MyMemoryButton(tempX,y,WIDTH,HEIGHT,memoryButtonText[i], **this**);
56. memoryButton[i].setForeground(Color.RED);
57. y+=HEIGHT+V\_SPACE;
58. }
60. //set Co-ordinates for Special Buttons
61. tempX=TOPX+1\*(WIDTH+H\_SPACE); y=TOPY+1\*(HEIGHT+V\_SPACE);
62. **for**(**int** i=0;i<specialButton.length;i++)
63. {
64. specialButton[i]=**new** MySpecialButton(tempX,y,WIDTH\*2,HEIGHT,specialButtonText[i], **this**);
65. specialButton[i].setForeground(Color.RED);
66. tempX=tempX+2\*WIDTH+H\_SPACE;
67. }
69. //set Co-ordinates for Digit Buttons
70. **int** digitX=TOPX+WIDTH+H\_SPACE;
71. **int** digitY=TOPY+2\*(HEIGHT+V\_SPACE);
72. tempX=digitX;  y=digitY;
73. **for**(**int** i=0;i<digitButton.length;i++)
74. {
75. digitButton[i]=**new** MyDigitButton(tempX,y,WIDTH,HEIGHT,digitButtonText[i], **this**);
76. digitButton[i].setForeground(Color.BLUE);
77. tempX+=WIDTH+H\_SPACE;
78. **if**((i+1)%3==0){tempX=digitX; y+=HEIGHT+V\_SPACE;}
79. }
81. //set Co-ordinates for Operator Buttons
82. **int** opsX=digitX+2\*(WIDTH+H\_SPACE)+H\_SPACE;
83. **int** opsY=digitY;
84. tempX=opsX;  y=opsY;
85. **for**(**int** i=0;i<operatorButton.length;i++)
86. {
87. tempX+=WIDTH+H\_SPACE;
88. operatorButton[i]=**new** MyOperatorButton(tempX,y,WIDTH,HEIGHT,operatorButtonText[i], **this**);
89. operatorButton[i].setForeground(Color.RED);
90. **if**((i+1)%2==0){tempX=opsX; y+=HEIGHT+V\_SPACE;}
91. }
93. addWindowListener(**new** WindowAdapter()
94. {
95. **public** **void** windowClosing(WindowEvent ev)
96. {System.exit(0);}
97. });
99. setLayout(**null**);
100. setSize(FRAME\_WIDTH,FRAME\_HEIGHT);
101. setVisible(**true**);
102. }
103. //////////////////////////////////
104. **static** String getFormattedText(**double** temp)
105. {
106. String resText=""+temp;
107. **if**(resText.lastIndexOf(".0")>0)
108. resText=resText.substring(0,resText.length()-2);
109. **return** resText;
110. }
111. ////////////////////////////////////////
112. **public** **static** **void** main(String []args)
113. {
114. **new** MyCalculator("Calculator - JavaTpoint");
115. }
116. }
118. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
120. **class** MyDigitButton **extends** Button **implements** ActionListener
121. {
122. MyCalculator cl;
124. //////////////////////////////////////////
125. MyDigitButton(**int** x,**int** y, **int** width,**int** height,String cap, MyCalculator clc)
126. {
127. **super**(cap);
128. setBounds(x,y,width,height);
129. **this**.cl=clc;
130. **this**.cl.add(**this**);
131. addActionListener(**this**);
132. }
133. ////////////////////////////////////////////////
134. **static** **boolean** isInString(String s, **char** ch)
135. {
136. **for**(**int** i=0; i<s.length();i++) **if**(s.charAt(i)==ch) **return** **true**;
137. **return** **false**;
138. }
139. /////////////////////////////////////////////////
140. **public** **void** actionPerformed(ActionEvent ev)
141. {
142. String tempText=((MyDigitButton)ev.getSource()).getLabel();
144. **if**(tempText.equals("."))
145. {
146. **if**(cl.setClear)
147. {cl.displayLabel.setText("0.");cl.setClear=**false**;}
148. **else** **if**(!isInString(cl.displayLabel.getText(),'.'))
149. cl.displayLabel.setText(cl.displayLabel.getText()+".");
150. **return**;
151. }
153. **int** index=0;
154. **try**{
155. index=Integer.parseInt(tempText);
156. }**catch**(NumberFormatException e){**return**;}
158. **if** (index==0 && cl.displayLabel.getText().equals("0")) **return**;
160. **if**(cl.setClear)
161. {cl.displayLabel.setText(""+index);cl.setClear=**false**;}
162. **else**
163. cl.displayLabel.setText(cl.displayLabel.getText()+index);
164. }//actionPerformed
165. }//class defination
167. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
169. **class** MyOperatorButton **extends** Button **implements** ActionListener
170. {
171. MyCalculator cl;
173. MyOperatorButton(**int** x,**int** y, **int** width,**int** height,String cap, MyCalculator clc)
174. {
175. **super**(cap);
176. setBounds(x,y,width,height);
177. **this**.cl=clc;
178. **this**.cl.add(**this**);
179. addActionListener(**this**);
180. }
181. ///////////////////////
182. **public** **void** actionPerformed(ActionEvent ev)
183. {
184. String opText=((MyOperatorButton)ev.getSource()).getLabel();
186. cl.setClear=**true**;
187. **double** temp=Double.parseDouble(cl.displayLabel.getText());
189. **if**(opText.equals("1/x"))
190. {
191. **try**
192. {**double** tempd=1/(**double**)temp;
193. cl.displayLabel.setText(MyCalculator.getFormattedText(tempd));}
194. **catch**(ArithmeticException excp)
195. {cl.displayLabel.setText("Divide by 0.");}
196. **return**;
197. }
198. **if**(opText.equals("sqrt"))
199. {
200. **try**
201. {**double** tempd=Math.sqrt(temp);
202. cl.displayLabel.setText(MyCalculator.getFormattedText(tempd));}
203. **catch**(ArithmeticException excp)
204. {cl.displayLabel.setText("Divide by 0.");}
205. **return**;
206. }
207. **if**(!opText.equals("="))
208. {
209. cl.number=temp;
210. cl.op=opText.charAt(0);
211. **return**;
212. }
213. // process = button pressed
214. **switch**(cl.op)
215. {
216. **case** '+':
217. temp+=cl.number;**break**;
218. **case** '-':
219. temp=cl.number-temp;**break**;
220. **case** '\*':
221. temp\*=cl.number;**break**;
222. **case** '%':
223. **try**{temp=cl.number%temp;}
224. **catch**(ArithmeticException excp)
225. {cl.displayLabel.setText("Divide by 0."); **return**;}
226. **break**;
227. **case** '/':
228. **try**{temp=cl.number/temp;}
229. **catch**(ArithmeticException excp)
230. {cl.displayLabel.setText("Divide by 0."); **return**;}
231. **break**;
232. }//switch
234. cl.displayLabel.setText(MyCalculator.getFormattedText(temp));
235. //cl.number=temp;
236. }//actionPerformed
237. }//class
239. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
241. **class** MyMemoryButton **extends** Button **implements** ActionListener
242. {
243. MyCalculator cl;
245. /////////////////////////////////
246. MyMemoryButton(**int** x,**int** y, **int** width,**int** height,String cap, MyCalculator clc)
247. {
248. **super**(cap);
249. setBounds(x,y,width,height);
250. **this**.cl=clc;
251. **this**.cl.add(**this**);
252. addActionListener(**this**);
253. }
254. ////////////////////////////////////////////////
255. **public** **void** actionPerformed(ActionEvent ev)
256. {
257. **char** memop=((MyMemoryButton)ev.getSource()).getLabel().charAt(1);
259. cl.setClear=**true**;
260. **double** temp=Double.parseDouble(cl.displayLabel.getText());
262. **switch**(memop)
263. {
264. **case** 'C':
265. cl.memLabel.setText(" ");cl.memValue=0.0;**break**;
266. **case** 'R':
267. cl.displayLabel.setText(MyCalculator.getFormattedText(cl.memValue));**break**;
268. **case** 'S':
269. cl.memValue=0.0;
270. **case** '+':
271. cl.memValue+=Double.parseDouble(cl.displayLabel.getText());
272. **if**(cl.displayLabel.getText().equals("0") || cl.displayLabel.getText().equals("0.0")  )
273. cl.memLabel.setText(" ");
274. **else**
275. cl.memLabel.setText("M");
276. **break**;
277. }//switch
278. }//actionPerformed
279. }//class
281. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
283. **class** MySpecialButton **extends** Button **implements** ActionListener
284. {
285. MyCalculator cl;
287. MySpecialButton(**int** x,**int** y, **int** width,**int** height,String cap, MyCalculator clc)
288. {
289. **super**(cap);
290. setBounds(x,y,width,height);
291. **this**.cl=clc;
292. **this**.cl.add(**this**);
293. addActionListener(**this**);
294. }
295. //////////////////////
296. **static** String backSpace(String s)
297. {
298. String Res="";
299. **for**(**int** i=0; i<s.length()-1; i++) Res+=s.charAt(i);
300. **return** Res;
301. }
303. //////////////////////////////////////////////////////////
304. **public** **void** actionPerformed(ActionEvent ev)
305. {
306. String opText=((MySpecialButton)ev.getSource()).getLabel();
307. //check for backspace button
308. **if**(opText.equals("Backspc"))
309. {
310. String tempText=backSpace(cl.displayLabel.getText());
311. **if**(tempText.equals(""))
312. cl.displayLabel.setText("0");
313. **else**
314. cl.displayLabel.setText(tempText);
315. **return**;
316. }
317. //check for "C" button i.e. Reset
318. **if**(opText.equals("C"))
319. {
320. cl.number=0.0; cl.op=' '; cl.memValue=0.0;
321. cl.memLabel.setText(" ");
322. }
324. //it must be CE button pressed
325. cl.displayLabel.setText("0");cl.setClear=**true**;
326. }//actionPerformed
327. }//class
329. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
330. Features not implemented and few bugs
332. i)  No coding done for "+/-" button.
333. ii) Menubar is not included.
334. iii)Not for Scientific calculation
335. iv)Some of the computation may lead to unexpected result
336. due to the representation of Floating point numbers in computer
337. is an approximation to the given value that can be stored
338. physically in memory.
339. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



**Lab Tasks**

1. Create a registration page for your semester enrollment. Add all the necessary controls to complete the form.
2. Write a calculator program using java swing.

**Step 1:** Start the program

**Step 2:** Using the swing components design the buttons of the calculator.

**Step 3:** Use key events and key listener to listen the events of the calculator

**Step 4:** Do the necessary manipulations.

**Step 5:** Stop the program.