ASSIGNMENT

ON

GUI Class hierarchy

COURSE NAME: **ADVANCE ENTERPRISE JAVA (THEORY)**

COURSE CODE: **SE 409**

SUBMITTED TO:

MD. SAFAET HOSSAIN

ASSOCIATE PROFESSOR AND HEAD

DEPARTMENT OF CSE

SUBMITTED BY:

ZULKAR NINE

163432601

43RD BATCH

DEPARTMENT OF CSE

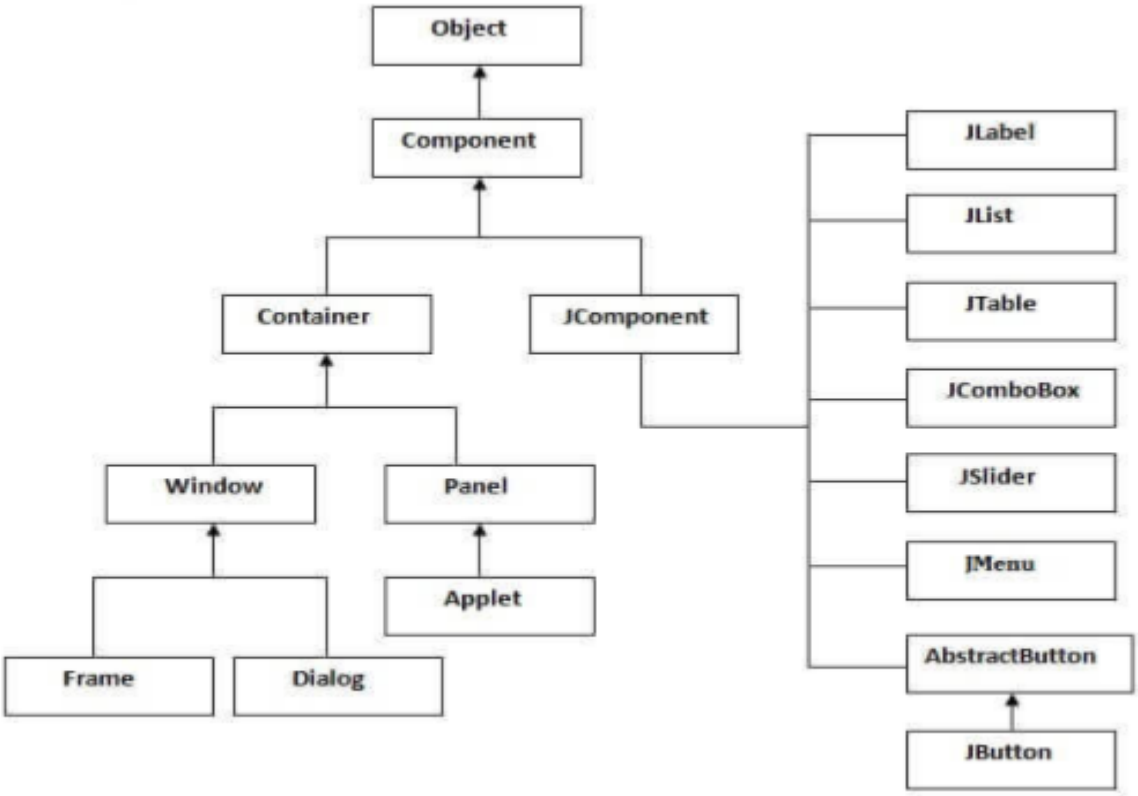
SUBMITTED DATE: 30-10-2020

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.

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

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.



Java JButton

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

```
1.  import javax.swing.*;
2.  public class ButtonExample {
3.  public static void main(String[] args) {
4.      JFrame f=new JFrame("Button Example");
5.      JButton b=new JButton("Click Here");
6.      b.setBounds(50,100,95,30);
7.      f.add(b);
8.      f.setSize(400,400);
```

```
9.      f.setLayout(null);
10.     f.setVisible(true);
11. }
12. }
```

Java JLabel

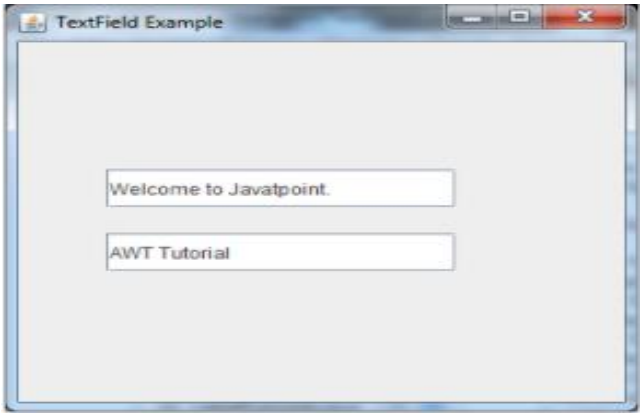
The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

```
1.  import javax.swing.*;
2.  class LabelExample
3.  {
4.  public static void main(String args[])
5.     {
6.     JFrame f= new JFrame("Label Example");
7.     JLabel l1,l2;
8.     l1=new JLabel("First Label.");
9.     l1.setBounds(50,50, 100,30);
10.    l2=new JLabel("Second Label.");
11.    l2.setBounds(50,100, 100,30);
12.    f.add(l1); f.add(l2);
13.    f.setSize(300,300);
14.    f.setLayout(null);
15.    f.setVisible(true);
16.    }
17.    }
```

Java JTextField

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

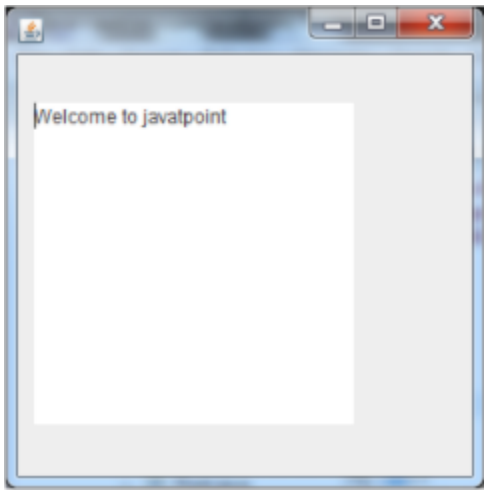
```
1.  import javax.swing.*;
2.  class TextFieldExample
3.  {
4.  public static void main(String args[])
5.      {
6.      JFrame f= new JFrame("TextField Example");
7.      JTextField t1,t2;
8.      t1=new JTextField("Welcome to Javatpoint.");
9.      t1.setBounds(50,100, 200,30);
10.     t2=new JTextField("AWT Tutorial");
11.     t2.setBounds(50,150, 200,30);
12.     f.add(t1); f.add(t2);
13.     f.setSize(400,400);
14.     f.setLayout(null);
15.     f.setVisible(true);
16.     }
17.     }
```



Java JTextArea

The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class

```
1.  import javax.swing.*;
2.  public class TextAreaExample
3.  {
4.      TextAreaExample(){
5.        JFrame f= new JFrame();
6.        JTextArea area=new JTextArea("Welcome to javatpoint");
7.        area.setBounds(10,30, 200,200);
8.        f.add(area);
9.        f.setSize(300,300);
10.       f.setLayout(null);
11.       f.setVisible(true);
12.    }
13. public static void main(String args[])
14.    {
15.   new TextAreaExample();
16.    }}
```



Java JPasswordField

The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

```
1.  import javax.swing.*;
2.  public class PasswordFieldExample {
3.     public static void main(String[] args) {
4.     JFrame f=new JFrame("Password Field Example");
5.      JPasswordField value = new JPasswordField();
```

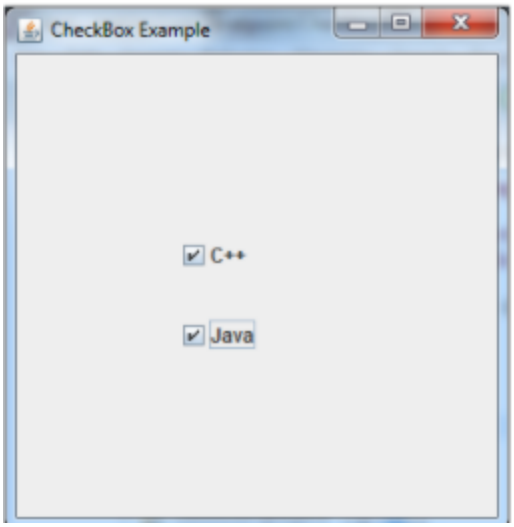
```
6.      JLabel l1=new JLabel("Password:");
7.        l1.setBounds(20,100, 80,30);
8.         value.setBounds(100,100,100,30);
9.           f.add(value);  f.add(l1);
10.          f.setSize(300,300);
11.          f.setLayout(null);
12.          f.setVisible(true);
13. }
14. }
```

Java 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.

```
1.  import javax.swing.*;
2.  public class CheckBoxExample
3.  {
4.      CheckBoxExample(){
5.        JFrame f= new JFrame("CheckBox Example");
6.        JCheckBox checkBox1 = new JCheckBox("C++");
7.        checkBox1.setBounds(100,100, 50,50);
8.        JCheckBox checkBox2 = new JCheckBox("Java", true);
9.        checkBox2.setBounds(100,150, 50,50);
10.       f.add(checkBox1);
11.       f.add(checkBox2);
12.       f.setSize(400,400);
13.       f.setLayout(null);
14.       f.setVisible(true);
15.     }
```

```
16. public static void main(String args[])
17.     {
18.     new CheckBoxExample();
19.     }}
```



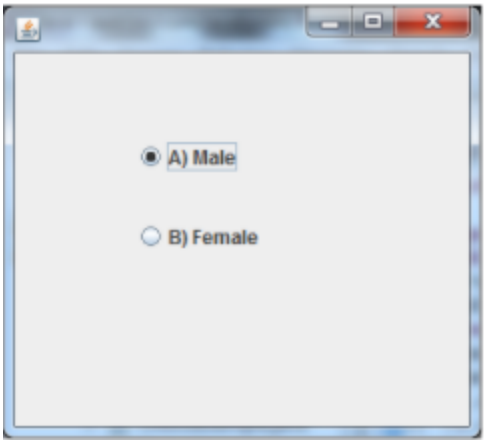
Java JRadioButton

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.

```
1.  import javax.swing.*;
2.  public class RadioButtonExample {
3.  JFrame f;
4.  RadioButtonExample(){
5.  f=new JFrame();
6.  JRadioButton r1=new JRadioButton("A) Male");
7.  JRadioButton r2=new JRadioButton("B) Female");
8.  r1.setBounds(75,50,100,30);
9.  r2.setBounds(75,100,100,30);
10. ButtonGroup bg=new ButtonGroup();
11. bg.add(r1);bg.add(r2);
12. f.add(r1);f.add(r2);
13. f.setSize(300,300);
14. f.setLayout(null);
15. f.setVisible(true);
16. }
17. public static void main(String[] args) {
```

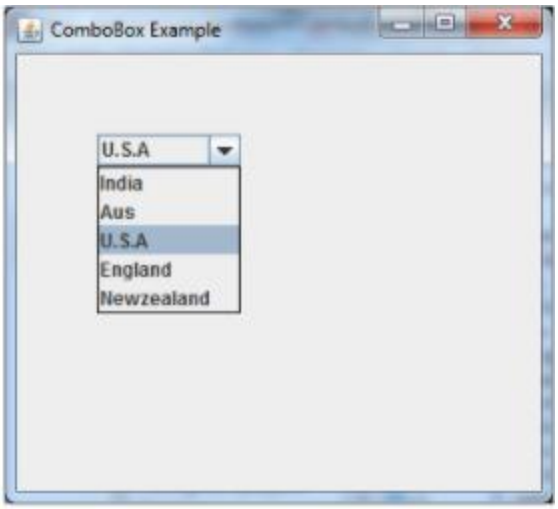
```
18.     new RadioButtonExample();
19. }
20. }
```



Java JComboBox

The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits JComponent class.

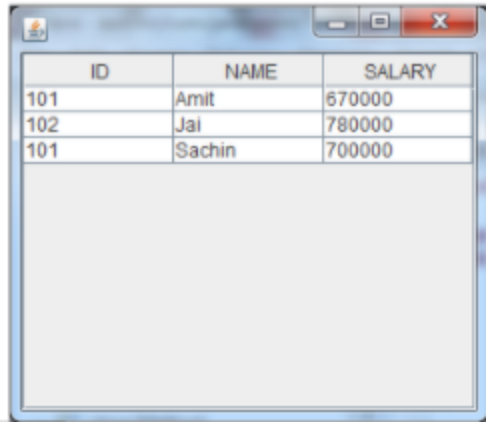
```
1.  import javax.swing.*;
2.  public class ComboBoxExample {
3.  JFrame f;
4.  ComboBoxExample(){
5.      f=new JFrame("ComboBox Example");
6.      String country[]={"India","Aus","U.S.A","England","Newzealand"};
7.      JComboBox cb=new JComboBox(country);
8.      cb.setBounds(50, 50,90,20);
9.      f.add(cb);
10.     f.setLayout(null);
11.     f.setSize(400,500);
12.     f.setVisible(true);
13. }
14. public static void main(String[] args) {
15.     new ComboBoxExample();
16. }
17. }
```



Java JTable

The JTable class is used to display data in tabular form. It is composed of rows and columns.

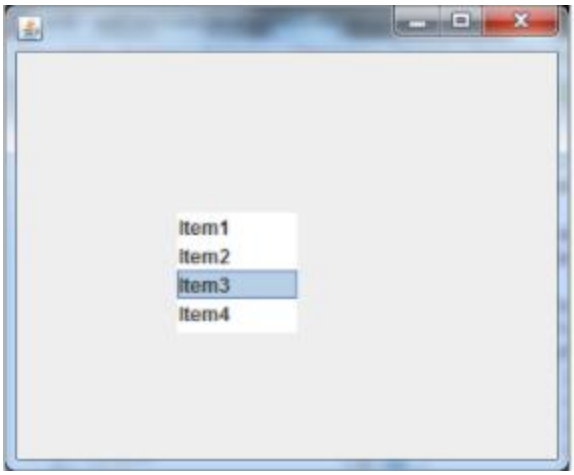
```
1.  import javax.swing.*;
2.  public class TableExample {
3.      JFrame f;
4.      TableExample(){
5.      f=new JFrame();
6.      String data[][]={ {"101","Amit","670000"},
7.                        {"102","Jai","780000"},
8.                        {"101","Sachin","700000"}};
9.      String column[]={"ID","NAME","SALARY"};
10.     JTable jt=new JTable(data,column);
11.     jt.setBounds(30,40,200,300);
12.     JScrollPane sp=new JScrollPane(jt);
13.     f.add(sp);
14.     f.setSize(300,400);
15.     f.setVisible(true);
16. }
17. public static void main(String[] args) {
18.     new TableExample();
19. }
20. }
```



Java JList

The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.

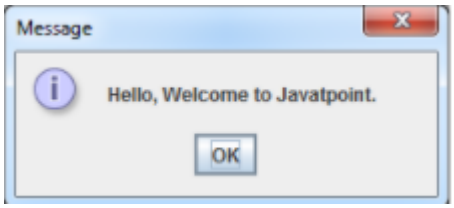
```
1.  import javax.swing.*;
2.  public class ListExample
3.  {
4.      ListExample(){
5.        JFrame f= new JFrame();
6.        DefaultListModel<String> l1 = new DefaultListModel<>();
7.          l1.addElement("Item1");
8.          l1.addElement("Item2");
9.          l1.addElement("Item3");
10.         l1.addElement("Item4");
11.         JList<String> list = new JList<>(l1);
12.         list.setBounds(100,100, 75,75);
13.         f.add(list);
14.         f.setSize(400,400);
15.         f.setLayout(null);
16.         f.setVisible(true);
17.    }
18. public static void main(String args[])
19.    {
20.   new ListExample();
21.    }}
```



Java JOptionPane

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

```
1.  import javax.swing.*;
2.  public class OptionPaneExample {
3.  JFrame f;
4.  OptionPaneExample(){
5.      f=new JFrame();
6.      JOptionPane.showMessageDialog(f,"Hello, Welcome to Javatpoint.");
7.  }
8.  public static void main(String[] args) {
9.      new OptionPaneExample();
10. }
11. }
```

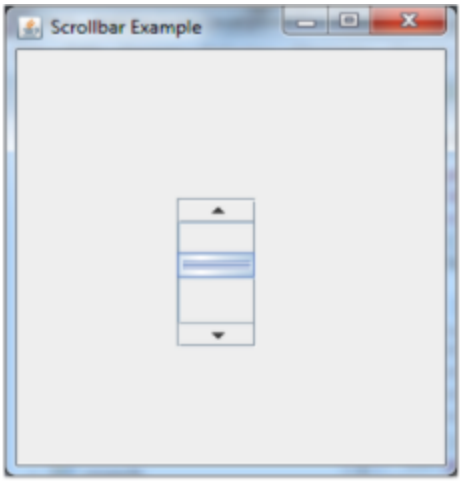


Java JScrollBar

The object of JScrollbar class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits JComponent class.

```
1.  import javax.swing.*;
2.  class ScrollBarExample
3.  {
4.  ScrollBarExample(){
```

```
5.      JFrame f= new JFrame("Scrollbar Example");
6.    JScrollBar s=new JScrollBar();
7.   s.setBounds(100,100, 50,100);
8.   f.add(s);
9.   f.setSize(400,400);
10. f.setLayout(null);
11. f.setVisible(true);
12. }
13. public static void main(String args[])
14. {
15. new ScrollBarExample();
16. }}
```



Java JMenuBar, JMenu and JMenuItem

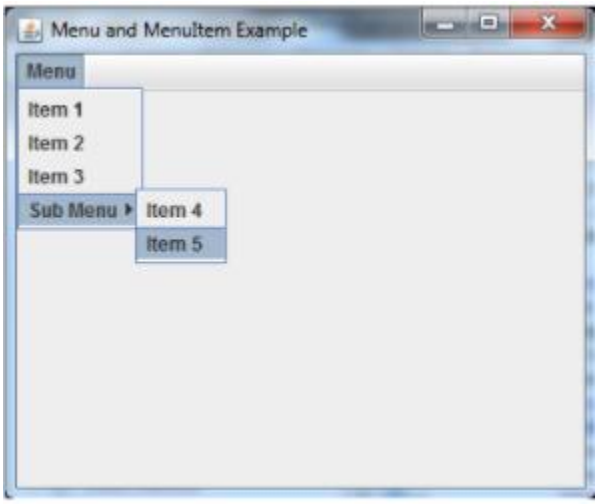
The JMenuBar class is used to display menubar on the window or frame. It may have several menus.

The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.

The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

```
1.  import javax.swing.*;
2.  class MenuExample
3.  {
4.          JMenu menu, submenu;
5.          JMenuItem i1, i2, i3, i4, i5;
6.          MenuExample(){
7.          JFrame f= new JFrame("Menu and MenuItem Example");
8.          JMenuBar mb=new JMenuBar();
```

```
9.          menu=new JMenu("Menu");
10.         submenu=new JMenu("Sub Menu");
11.         i1=new JMenuItem("Item 1");
12.         i2=new JMenuItem("Item 2");
13.         i3=new JMenuItem("Item 3");
14.         i4=new JMenuItem("Item 4");
15.         i5=new JMenuItem("Item 5");
16.         menu.add(i1); menu.add(i2); menu.add(i3);
17.         submenu.add(i4); submenu.add(i5);
18.         menu.add(submenu);
19.         mb.add(menu);
20.         f.setJMenuBar(mb);
21.         f.setSize(400,400);
22.         f.setLayout(null);
23.         f.setVisible(true);
24. }
25. public static void main(String args[])
26. {
27. new MenuExample();
28. }}
```



Java JSlider

The Java JSlider class is used to create the slider. By using JSlider, a user can select a value from a specific range.

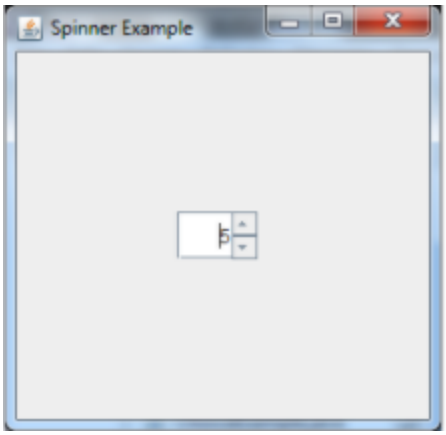
```
1. import javax.swing.*;
2. public class SliderExample1 extends JFrame{
3. public SliderExample1() {
```

```
4.  JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 50, 25);
5.  JPanel panel=new JPanel();
6.  panel.add(slider);
7.  add(panel);
8.  }
9.
10. public static void main(String s[]) {
11. SliderExample1 frame=new SliderExample1();
12. frame.pack();
13. frame.setVisible(true);
14. }
15. }
```

Java JSpinner

The object of JSpinner class is a single line input field that allows the user to select a number or an object value from an ordered sequence.

```
1.  import javax.swing.*;
2.  public class SpinnerExample {
3.     public static void main(String[] args) {
4.     JFrame f=new JFrame("Spinner Example");
5.     SpinnerModel value =
6.             new SpinnerNumberModel(5, //initial value
7.               0, //minimum value
8.               10, //maximum value
9.               1); //step
10.    JSpinner spinner = new JSpinner(value);
11.          spinner.setBounds(100,100,50,30);
12.          f.add(spinner);
13.          f.setSize(300,300);
14.          f.setLayout(null);
15.          f.setVisible(true);
16. }
17. }
```



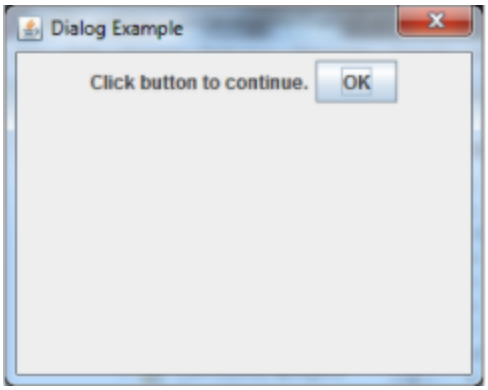
Java JDialog

The JDialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Dialog class.

Unlike JFrame, it doesn't have maximize and minimize buttons.

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.  public class DialogExample {
5.      private static JDialog d;
6.      DialogExample() {
7.          JFrame f= new JFrame();
8.          d = new JDialog(f , "Dialog Example", true);
9.          d.setLayout( new FlowLayout() );
10.         JButton b = new JButton ("OK");
11.         b.addActionListener ( new ActionListener()
12.         {
13.             public void actionPerformed( ActionEvent e )
14.             {
15.                 DialogExample.d.setVisible(false);
16.             }
17.         });
18.         d.add( new JLabel ("Click button to continue."));
19.         d.add(b);
20.         d.setSize(300,300);
21.         d.setVisible(true);
22.     }
23.     public static void main(String args[])
```

```
24.     {
25.         new DialogExample();
26.     }
27. }
```



Java JPanel

The JPanel is a simplest container class. It provides space in which an application can attach any other component. It inherits the JComponents class.

It doesn't have title bar.

```
1.  import java.awt.*;
2.  import javax.swing.*;
3.  public class PanelExample {
4.      PanelExample()
5.        {
6.        JFrame f= new JFrame("Panel Example");
7.        JPanel panel=new JPanel();
8.        panel.setBounds(40,80,200,200);
9.        panel.setBackground(Color.gray);
10.       JButton b1=new JButton("Button 1");
11.       b1.setBounds(50,100,80,30);
12.       b1.setBackground(Color.yellow);
13.       JButton b2=new JButton("Button 2");
14.       b2.setBounds(100,100,80,30);
15.       b2.setBackground(Color.green);
16.       panel.add(b1); panel.add(b2);
17.       f.add(panel);
18.             f.setSize(400,400);
19.             f.setLayout(null);
```

```
20.             f.setVisible(true);
21.         }
22.         public static void main(String args[])
23.         {
24.         new PanelExample();
25.         }
26.     }
```

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.

```
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. }
```