Java Swing | Abstract Window Toolkits

Part of Java Foundation Classes (JFC) that is used to create window-based applications. Swing is built on the top of AWT (Abstract Windowing Toolkit) API.

Last update – March, 2019

Table of Contents

SCOPE OF THIS DOCUMENT

In order to make it more fun to work on a Java project, let's use graphical user interface instead of text based only.

The following is meant to provide you rudimentary information to work with so that you can give your project a more modern appearance –

- window frame
- Text / Message panel
- Buttons
- Grids
- Event Handling – this is perhaps the most important feature

There is a wealth of information online. Here is a good one for reference:

- https://www.javatpoint.com/java-swing

IMPORTANT PACKAGES

Classes that you will use in this packet:

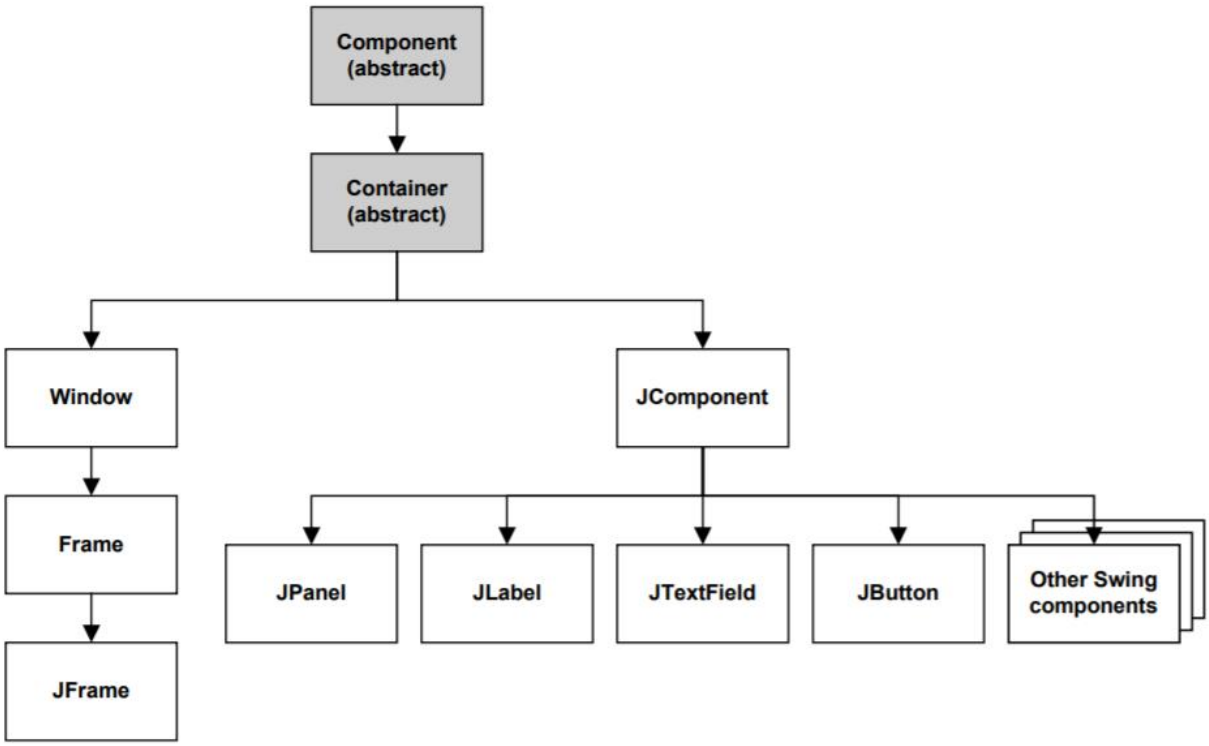
- JFrame
- JOptionPane
- JPanel
- GridLayout
- JButton
- JLabel
- Font
- Graphics
- Border
- ActionListener – Event Handler ***

```
import  javax.swing.*;
import  javax.swing.JFrame;
import  javax.swing.JPanel;
import  javax.swing.JButton;
import  javax.swing.JOptionPane;
import  javax.swing.JLabel;

import  java.awt.GridLayout;
import  java.awt.Point;
import  java.awt.Color;
import  java.awt.Font;
import  java.awt.BorderLayout;
import  javax.swing.BorderFactory;
import  javax.swing.border.Border;
import  javax.swing.border.BevelBorder;

import  java.awt.event.ActionEvent;
import  java.awt.event.ActionListener;
```

You will work with "container" classes.

- A container is a component class which can contain other components inside itself.
- It is also an instance of a subclass of java.awt.Container.
- java.awt.Container extends java.awt.Component so containers are themselves components.

Component class hierarchy:



Ref: https://web.csulb.edu/~pnguyen/cecs277/lecnotes/guipart1.pdf

Create the Frame

```
JFrame f = new JFrame("My Frame Title Name");

// NOTE: from now, all the samples below will use "f" as the main JFrame

// enable exit process by clicking the ☒
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

f.setSize(400,400);        // set width, height
f.setVisible(true);        // make it viewable
```

Change the Frame background Color

```
panel = new JPanel();

Color myColor = new Color (0x85, 0x15, 0x15);
f.setContentPane(panel);
f.getContentPane().setBackground(myColor);
```

Add Grids Layout

```
panel = new JPanel(new GridLayout(3,2,2,2)); //rows, cols, h-gap, V-gap
```

- instead of just panel = new JPanel();

Add Labels

```
labelL = new JLabel("L",JLabel.CENTER );        // set label field at 1st grid
labelR = new JLabel("R",JLabel.CENTER );        // set label field at 2nd grid

labelL.setFont(new Font("Arial", Font.PLAIN, 20));
labelR.setFont(new Font("Arial", Font.ITALIC, 20));

labelL.setForeground(Color.white);              // set font color
labelR.setForeground(Color.white);              // set font color

f.add(labelL);      // associate these components with the main Frame "f"
f.add(labelR);
```

Create Buttons

```
JButton theButton = new JButton("button 1");

theButton.setFont(new Font("Arial", Font.BOLD, 40));  // font family, bold, font size

f.add(theButton);
```

Creating several buttons with the grid # as button name.

```
for (int i=0; i<MaxGrids; i++) {

  // put grid # as string into inside the grids
  b[i] = new JButton(Integer.toString (i+1));
  b[i].setFont(new Font("Arial", Font.BOLD, 40));
  f.add(b[i]);

}
```

Polishing the border of the button

```
Border border = BorderFactory.createBevelBorder(
                    BevelBorder.RAISED,
                    new Color(0xff, 0xff, 0xff),
                    new Color(0xc0, 0xc0, 0xc0));

theButton.setBorder(border);
```

STOP HERE AND DO A COUPLE OF EXERCISES.

1- Write a Java program to create a frame with 9x9 grids. All grids should show the grid #.

```
Import ...

public class myGUI {

  public simpleGui( String  ...) {  {
        ...
  }

public static void main(String[] args) {
     simpleGui win = new simpleGui("My Grids");
  }
}
```

2- Do (1), except you will create your class inheriting JFame:

```
public class simpleGUI  extends JFrame {

  ...
}
```

Reminder

— Now you did not create a new JFrame object call "f". Thus, you can reference JFrame's methods directly , i.e. no "f."
— With inheritance, "this" means referencing yourself – the current object.

SET UP EVENT LISTENER RESPONDING TO CLICKS

Now, it will be good to trigger an action when you click on one of the buttons/grids. First, you should up an event listener…

```
static ActionListener evt;
      …

  evt = new ActionListener() {
     @Override
     public void actionPerformed(ActionEvent e)    {
     JButton b = (JButton) e.getSource();

     b.setText("Clicked");
  }
  };
```

ASSOCIATE THE EVENT LISTENER INTO EACH BUTTON

```
…

for (int i=0; i<MaxGrids; i++) {
   b[i] = new JButton( Integer.toString (i+1) );    //  put text  inside the grids
   b[i].setFont(new Font("Arial", Font.BOLD, 40));
   b[i].addActionListener(evt);
   f.add(b[i]);
}
```

HOW TO CREATE A MESSAGE WINDOW

```
JOptionPane p = new JOptionPane();

//the java layout manager will put pane in the center of the screen…

p.showMessageDialog((JFrame)null, "Let's Play a Game!");

// the java layout manager will put this pane on top of the current Frame.
// Remember they are in layers, so you can move the pane around as well.

p.showMessageDialog( ( theFrameObject, "Let's Play a Game!");
```

HOW DO YOU KNOW WHICH GRID IS CLICKED ON…

```
User the getSource( ) from event instance  from  actionPerformed(ActionEvent e)

static ActionListener evt;
…

  evt = new ActionListener() {
     @Override
     public void actionPerformed(ActionEvent e)    {

     if(e.getSource() instanceof JButton) {
        JButton b = (JButton) e.getSource();
        labelR.setText( "You clicked " +  b.getText()  );
        // or
        b.setText("Clicked!");
     }
     }
  };
```

EXERCISE

Write a program to do 2 player tic-tac-toe. Your code should use the following control structure :

- instanceof
- For each, e.g.

```
public static void main(String[] args) {
      int arr[] = { 1, 3, 10, 50 };
      int ct=0;
      for (int val : arr) {
          ct++;
          System.out.print( val + (arr.length==ct ? ".\n\n" : "," ));
      }

      ct=0;
      for(String arg: args) {
          ct++;
          System.out.print( arg + (args.length==ct ? ".\n\n" : "," ));
      }

}
```

Don't forget how to check the length

GET USER INPUT MESSAGE

CREATE AN INPUT DIALOG BOX AND READ IN USER INPUT.

```
...
    String s_num1 = JOptionPane.showInputDialog("Enter number 1: ");
    int i_num1 = Integer.parseInt(s_num1);

    String s_num2 = JOptionPane.showInputDialog("Enter number 2: ");
    int i_num2 = Integer.parseInt(s_num2);

    int sum = i_num1 + i_num2;

    JOptionPane.showMessageDialog(null, "the sum is : " + sum , "Results",
            JOptionPane.PLAIN_MESSAGE );
```

RESTRICT USER'S INPUT BY USING DROPDOWN...

```
...
    String selected = (String) JOptionPane.showInputDialog( null,
       "Select Meat...",                    // message to tell user what to do
       "Favorite Meat",                     // title of the message box
       JOptionPane.QUESTION_MESSAGE,
       null,
       food,
       food[0]);

    // selected will be null if the user clicks Cancel
    JOptionPane.showMessageDialog(null,     // use default frame
       selected ,                           // show what user has chosen
       "Good Choice!",                      // title of the message box
       JOptionPane.PLAIN_MESSAGE );
```

Draw a Shape

Inherits Jpanel

```
import java.awt.Graphics;

public class d3simple extends JPanel
{
  ...
     // implement your own paint method..
     public void paint(Graphics shape)
     {
        ...
     }
}
```

Draw a Rectangle

```
public void paint(Graphics g) {
     g.draw3DRect( int topX, int topY, int width, int height,   bool true|false);
                                                                  // raised | low
     g.fill3DRect( ... same parameters list   );
}
```

Draw an Oval

```
public void paint(Graphics g) {

  g.drawOval(25, 25, 120, 120);   topx1 = topx1+ w+gap;

  g.shape.fillOval( int topX, int topY, int width, int height);
  ...
}
```

Draw polygon

```
public void paint(Graphics g) {
     int xpoints[] = {25, 145, 25, 145, 25};
     int ypoints[] = {25, 25, 145, 145, 25};
     int npoints = 5;

     g.drawPolygon(xpoints, ypoints, npoints);
  }
```

MAKE A FANCY EMBOSS BORDER LOOK

```
public void paint(Graphics g) {
    int xpoints[] = {25, 145, 25, 145, 25};
    int ypoints[] = {25, 25, 145, 145, 25};
    int npoints = 5;

    g.drawPolygon(xpoints, ypoints, npoints);
}
```

EXERCISE

1) Write a program with methods to draw, a square, a rectangle, a circle, an ellipse, a right-angle triangle, an isosceles triangle, an scalene triangle.
2) Write a program which allows polymorphics

Ref: Graphics[api]