# Computer Science Workshop-2 (CSE3141) (Media: Graphics, Audio, Video)

by,
Smita Mohanty
Assistant Professor
Department of Computer Science & Engineering
ITER, SOA Deemed To Be University
Email Id: smitamohanty@soa.ac.in

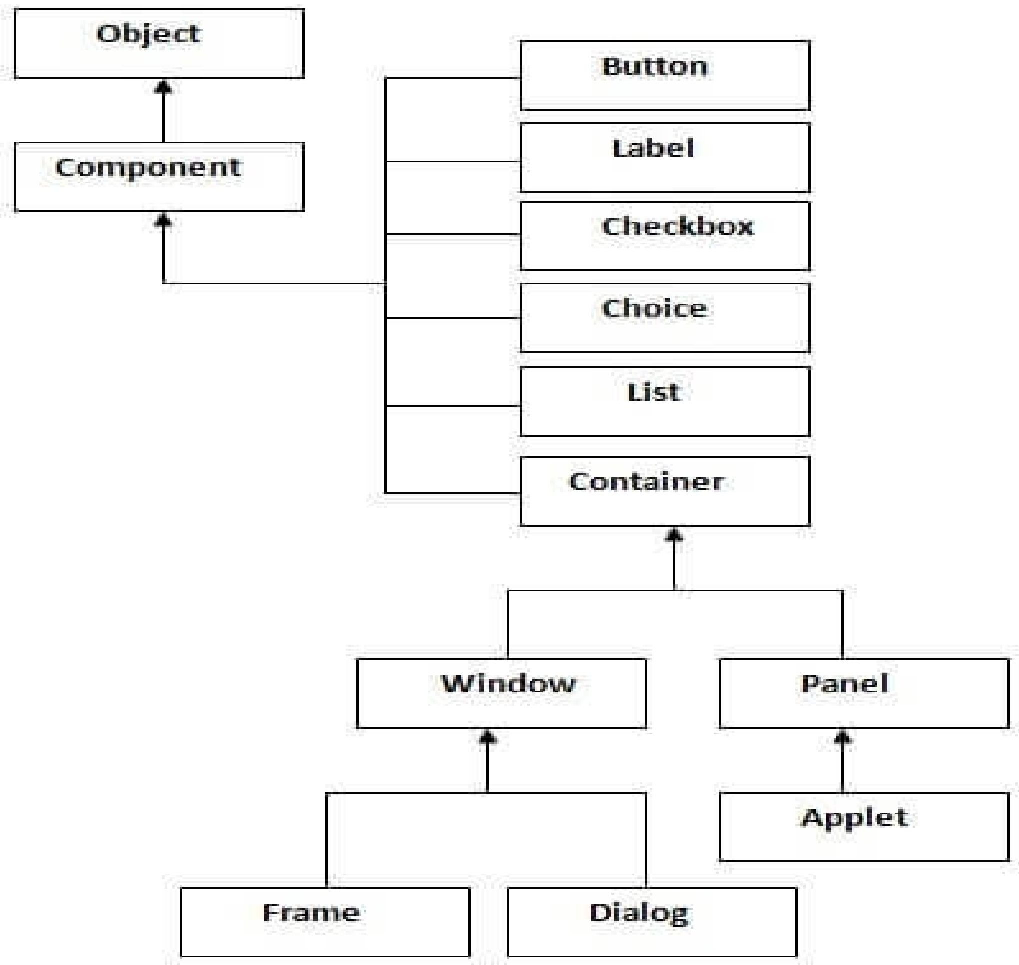
## Introduction

- Java has had windowing capabilities since its earliest days. The first version made public was the Abstract Windowing Toolkit, or AWT. Because it used the native toolkit components, AWT was relatively small and simple.
- The second major implementation was the Swing classes, released in 1998 as part of the Java Foundation Classes. Swing is a full-function, professional-quality GUI toolkit designed to enable almost any kind of client-side GUI-based interaction.
- AWT lives inside, or rather underneath, Swing, and, for this reason, many programs begin by importing both java.awt and javax.swing.

## Java AWT

- AWT - Abstract Window Toolkit
- API to develop GUI or window based application.
- AWT components are platform dependent. i.e., look and feel depends on OS
- Java.awt package is used.

# AWT Hierarchy



## AWT Hierarchy

- Container is a component in awt which can contain other components like button, textfield etc.
- Frame, Dialog, Applet extends Container.
- So through all these classes we can create Container and add components to it.
- Window is also a container, but it has no border and menu bar.
- Panel is a container which does not have title bar, menu bar.
- Frame contains title bar, menu bar and other components like button, text field etc.

## Containers and Components

- Component: Components are elementary GUI entities, such as Button, Label, and TextField.
- Container: Containers, such as Frame and Panel, are used to hold components in a specific layout (such as FlowLayout or GridLayout). A container can also hold sub-containers.

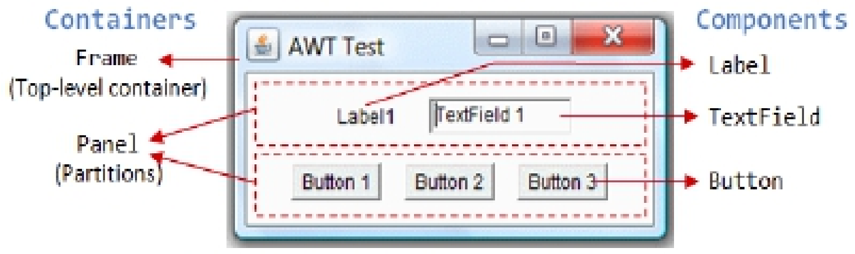


Figure 14.1: Containers and Components

## Containers and Components

- **AWT Container Classes**
- Each GUI program has a top-level container. The commonly-used top-level containers in AWT are Frame, Dialog and Applet:
- A Frame provides the "main window" for your GUI application. It has a title bar (containing an icon, a title, the minimize, maximize/restore-down and close buttons), an optional menu bar, and the content display area.
- An AWT Dialog is a "pop-up window" used for interacting with the users. A Dialog has a title-bar (containing an icon, a title and a close button) and a content display area, as illustrated.

## Containers and Components

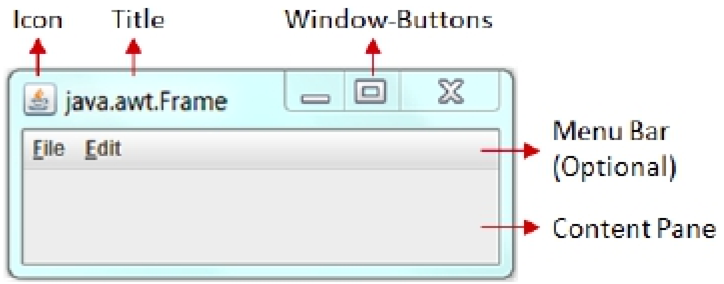


Figure 14.2: AWT Container Classes

## Important methods of Component class

- public void add(Component c) - adds a component to container
- public void setSize(int width, int height) - sets the size of window
- public void setLayout(LayoutManager m) - defines different layout like flow, grid layout etc.
- public void setVisible(boolean status) - changes the visibility of component as per requirement

# AWT UI Elements

1) Label: Does not create any event.

- User not allowed to change the text
- Label l = new Label("CSE");
- l.setText("CSE classes");
- l.getText();                    // returns the string value

2) TextField: Allows user to edit single line of text.

- Event is generated by user typing in text field.
- textField tf = new TextField(10);
- tf.setText();

3) Button: Generates an event when pressed/released.

- Button b = new Button("Click Here");

# AWT UI Elements

4) CheckBox: used to turn an option true or false

- No radio class is there in AWT.
- CheckBox c1 = new CheckBox("CSE", null, true);
- CheckBox c2 = new CheckBox("CSIT", null, false);

- CheckBoxGroup cbg = new CheckBoxGroup(); // we can create radio

5) List: represents a list of text items.

- User can choose any of them

6) Choice: Used to show popup menu of choices

- Selected choice is shown on the top of menu.

# AWT UI Elements

7) textArea: provides multiline editor area.

8) Menu:

- MenuBar and MenuItem come under this

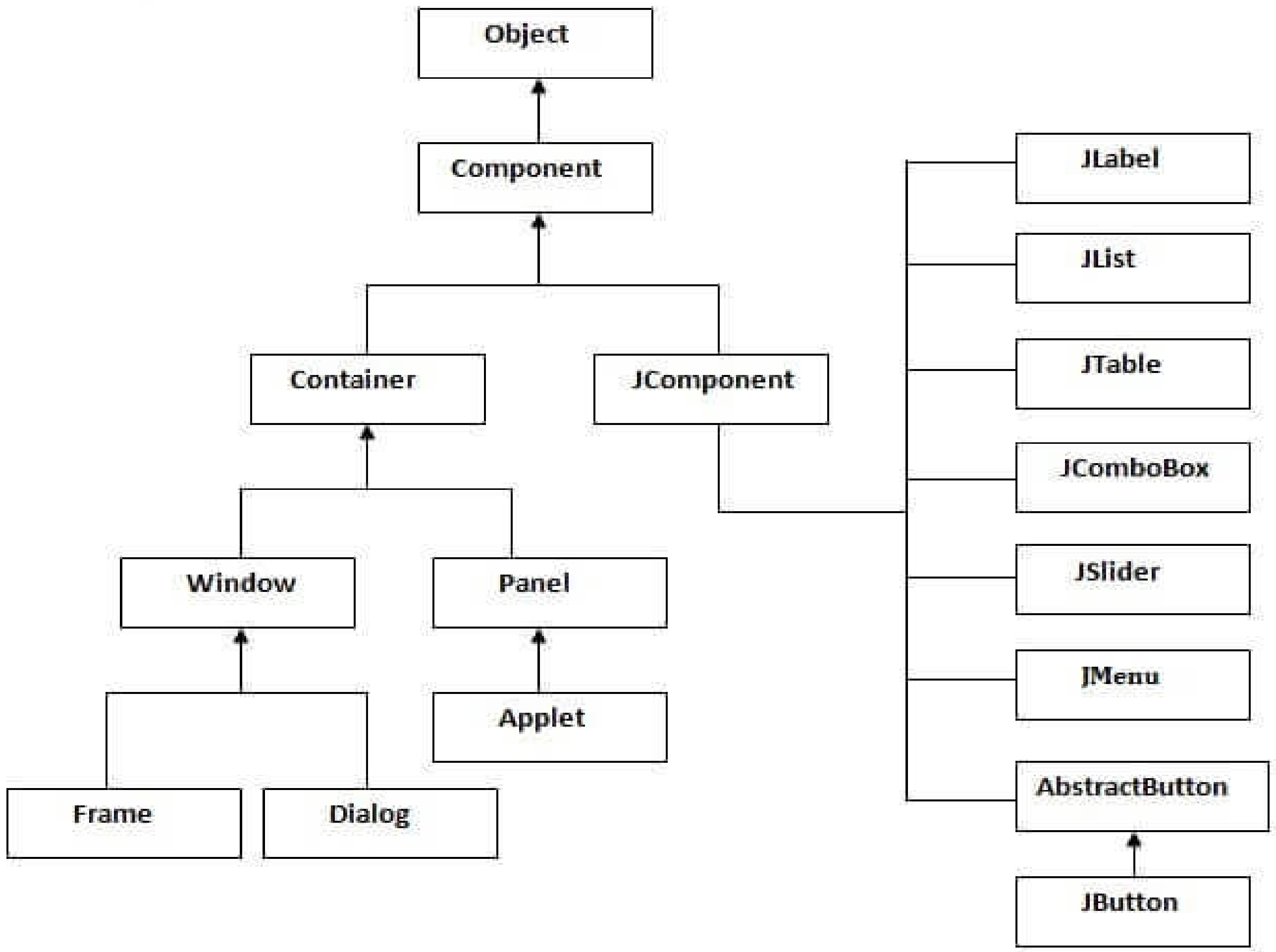
9) ScrollBar:

- can be horizontal or vertical

## JAVA Swing

- Swing is part of Java Foundation classes(JFC) that is used to create window-based applications.
- Built on top of AWT API and entirely written in java.
- Used for developing windows based application

## Swing Hierarchy



## AWT vs. Swing

- AWT (components) is platform dependent, the look and feel depends on the os.
- Swing (components) is platform independent.

- AWT components are heavyweight,
- Swing components are lightweight, need less no. of resource.

## AWT vs. Swing

- AWT does not support pluggable look and feel.
- Swing supports pluggable look and feel.

- AWT supports less no. of components
- Swing supports more no. of components

- AWT does not follow MVC architecture
- Swing follows MVC architecture.

## Important java Swing classes

1) JButton class - used to create a button

| | |
|---|---|
| JButton() | - No text/icon |
| JButton(String s) | - text |
| JButton(Icon i) | - icon |
| setText(String s) | |
| getText() | |

2) JRadioButton class - used to create radio button

- choose one/single from multiple options

JRadioButton() - no text

JRadioButton(String s) - text

JRadioButton(String s, boolean select) - text with selected state

## Java Swing Examples

- There are two ways to create a frame:

  By creating the object of Frame class (association)

  By extending Frame class (inheritance)

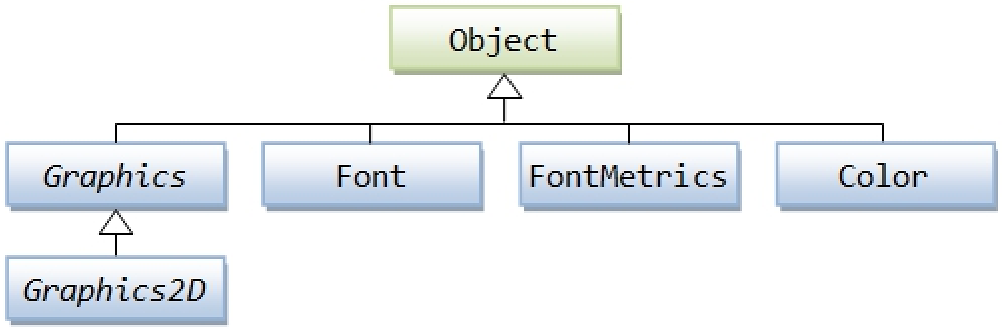
- We can write the code of swing inside the main(), constructor or any other method.

## Displaying graphics in swing: (Custom Graphics)

- A graphics context provides the capabilities of drawing on the screen.
- The graphics context maintains states such as the color and font used in drawing, as well as interacting with the underlying operating system to perform the drawing.
- In java, custom painting is done via the java.awt.Graphics class, which manages a graphics context, and provides a set of device-independent methods for drawing texts, figures and images on the screen on different platforms.

# Displaying graphics: (Custom Graphics)

- The java.awt.Graphics is an abstract class, as the actual act of drawing is system-dependent and device-dependent. Each operating platform will provide a subclass of Graphics to perform the actual drawing under the platform, but conform to the specification defined in Graphics.



## Graphics class Drawing methods

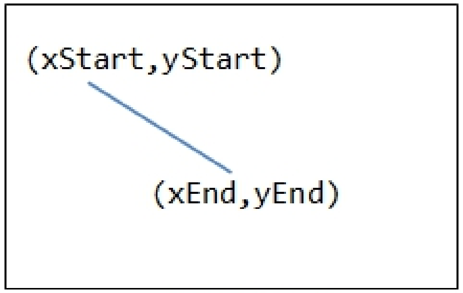
- The Graphics class provides methods for drawing three types of graphical objects:

1. Text strings: via the drawString() method. Take note that System.out.println() prints to the system console, not to the graphics screen.
2. Vector-graphic primitives and shapes: via methods drawXxx() and fillXxx(), where Xxx could be Line, Rect, Oval, Arc, PolyLine, RoundRect, or 3DRect.
3. Bitmap images: via the drawImage() method.

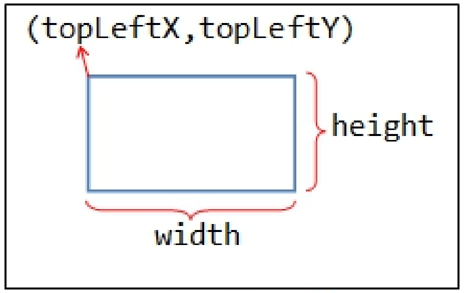
# Graphics class Drawing methods

```
// Drawing (or printing) texts on the graphics screen:
drawString(String str, int xBaselineLeft, int yBaselineLeft);

// Drawing lines:
drawLine(int x1, int y1, int x2, int y2);
drawPolyline(int[] xPoints, int[] yPoints, int numPoint);

// Drawing primitive shapes:
drawRect(int xTopLeft, int yTopLeft, int width, int height);
drawOval(int xTopLeft, int yTopLeft, int width, int height);
drawArc(int xTopLeft, int yTopLeft, int width, int height, int startAngle, int arcAngle);
draw3DRect(int xTopLeft, int, yTopLeft, int width, int height, boolean raised);
drawRoundRect(int xTopLeft, int yTopLeft, int width, int height, int arcWidth, int arcHeight)
drawPolygon(int[] xPoints, int[] yPoints, int numPoint);

// Filling primitive shapes:
fillRect(int xTopLeft, int yTopLeft, int width, int height);
fillOval(int xTopLeft, int yTopLeft, int width, int height);
fillArc(int xTopLeft, int yTopLeft, int width, int height, int startAngle, int arcAngle);
fill3DRect(int xTopLeft, int, yTopLeft, int width, int height, boolean raised);
fillRoundRect(int xTopLeft, int yTopLeft, int width, int height, int arcWidth, int arcHeight)
fillPolygon(int[] xPoints, int[] yPoints, int numPoint);

// Drawing (or Displaying) images:
drawImage(Image img, int xTopLeft, int yTopLeft, ImageObserver obs);  // draw image with its size
```

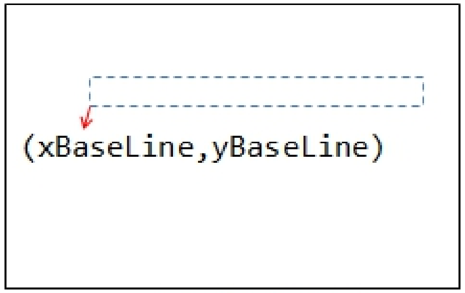
# Graphics class Drawing methods



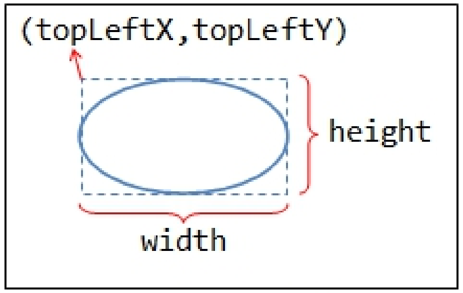
**drawLine()**



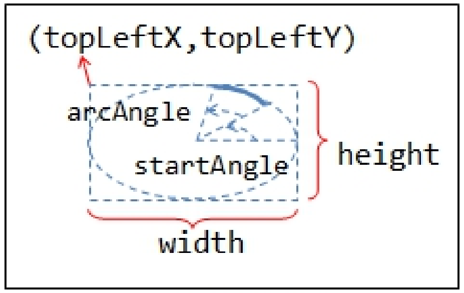
**drawRect()**



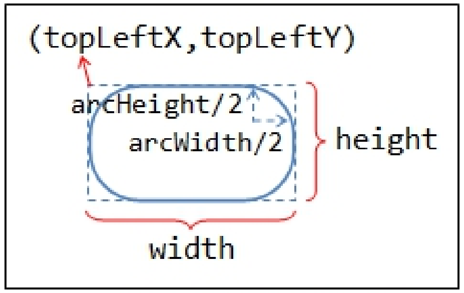
**drawString()**



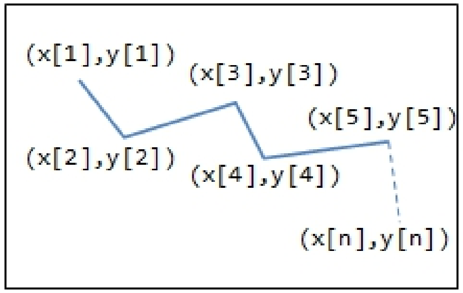
**drawOval()**

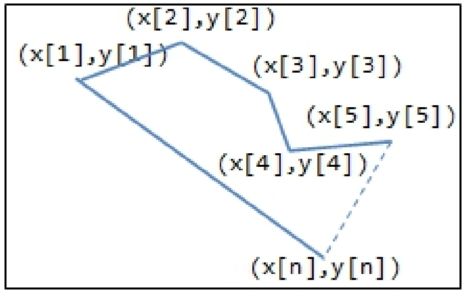


**drawArc()**



**drawRoundRect()**





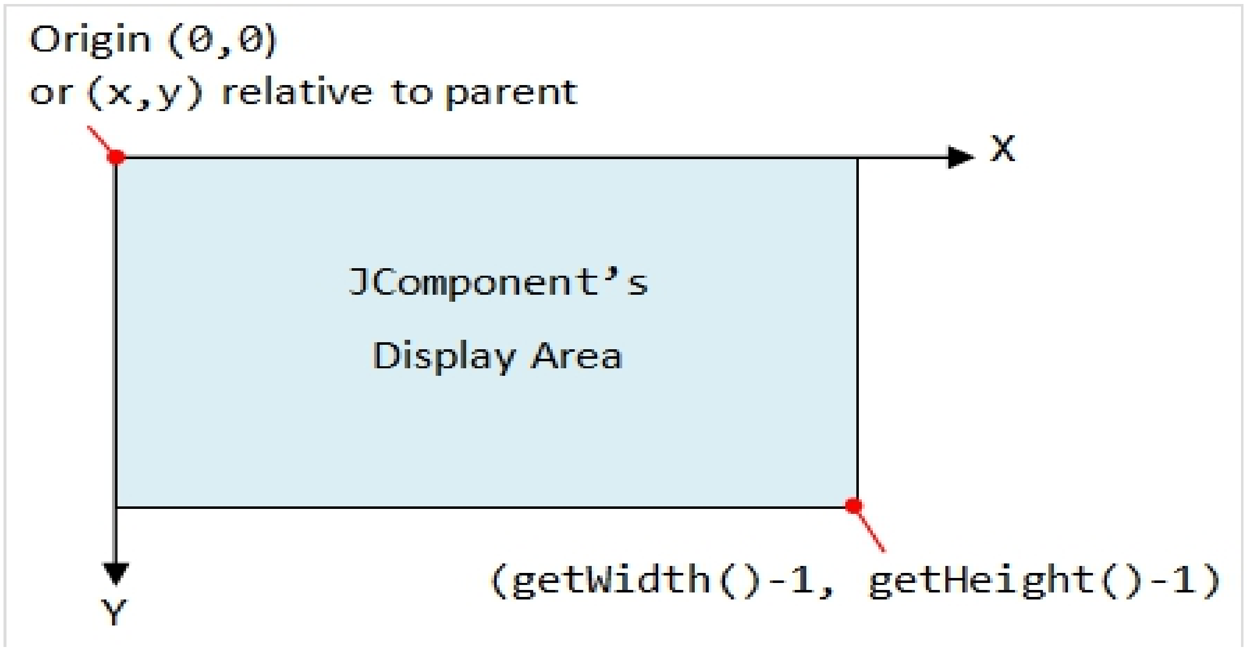
## Graphics class Drawing methods

- The drawXxx() methods draw the outlines; while fillXxx() methods fill the internal. Shapes with negative width and height will not be painted.

## Graphics Coordinate System

- In Java Windowing Subsystem (like most of the 2D Graphics systems), the origin (0,0) is located at the top-left corner.
- EACH component/container has its own coordinate system, ranging for (0,0) to (width-1, height-1) as illustrated.
- You can use method getWidth() and getHeight() to retrieve the width and height of a component/container. You can use getX() or getY() to get the top-left corner (x,y) of this component's origin relative to its parent.

## Graphics Coordinate System



THANK YOU