SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Coimbatore - 641035.

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++" Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

Department of Computer Applications

Course Code: 23CAT606

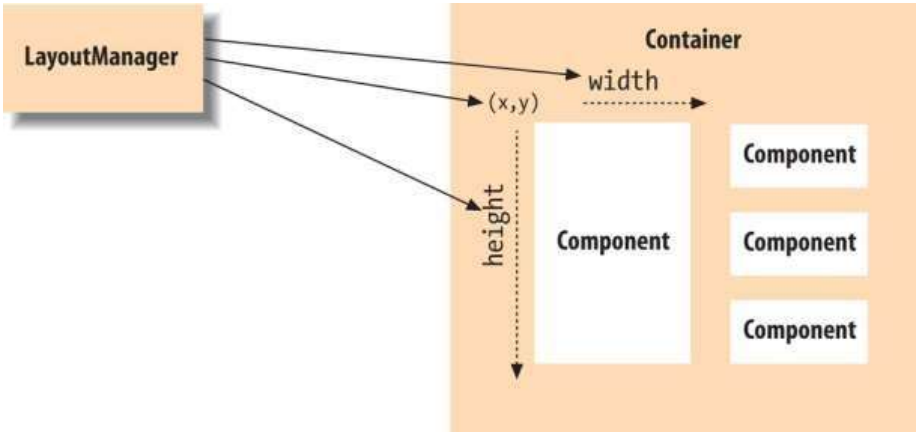
Course Name: Java Programming

Unit II: Package

Topic 9: Layout Container

INTRODUCTION: LAYOUT

1. Layout means the arrangement of components within the container. In other way we can say that placing the components at a particular position within the container.



LAYOUTS: TYPES

1. BorderLayout
2. BoxLayout
3. CardLayout
4. FlowLayout
5. GridBagLayout
6. GridLayout
7. GroupLayout
8. SpringLayout

BorderLayout

1. Every content pane is initialized to use a BorderLayout.
2. BorderLayout places components in up to **five areas**:
    1. Top
    2. Bottom
    3. Left
    4. Right
    5. Center.
3. All extra space is placed in the center area.
4. Tool bars that are created using JToolBar must be created within a BorderLayout container, if User want to be able to drag and drop the bars away from their starting positions.

public static final int NORTH

public static final int SOUTH

public static final int EAST

public static final int WEST

public static final int CENTER

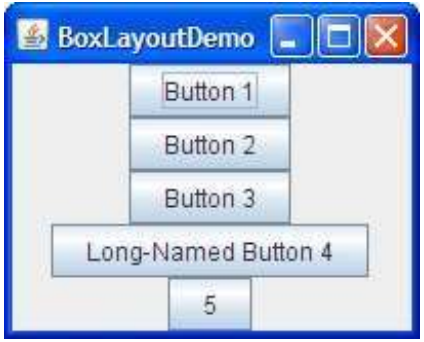
BorderLayout

```
import java.awt.*;
import javax.swing.*;
public class Border {
JFrame f;
Border(){
   f=new JFrame();
  JButton b1=new JButton("NORTH");;
  JButton b2=new JButton("SOUTH");;
  JButton b3=new JButton("EAST");;
  JButton b4=new JButton("WEST");;
  JButton b5=new JButton("CENTER");;
  f.add(b1,BorderLayout.NORTH);
  f.add(b2,BorderLayout.SOUTH);
  f.add(b3,BorderLayout.EAST);
  f.add(b4,BorderLayout.WEST);
  f.add(b5,BorderLayout.CENTER);
  f.setSize(300,300);
  f.setVisible(true);
}
```

```
public static void main(String[] args) {
   new Border();
}
}
```

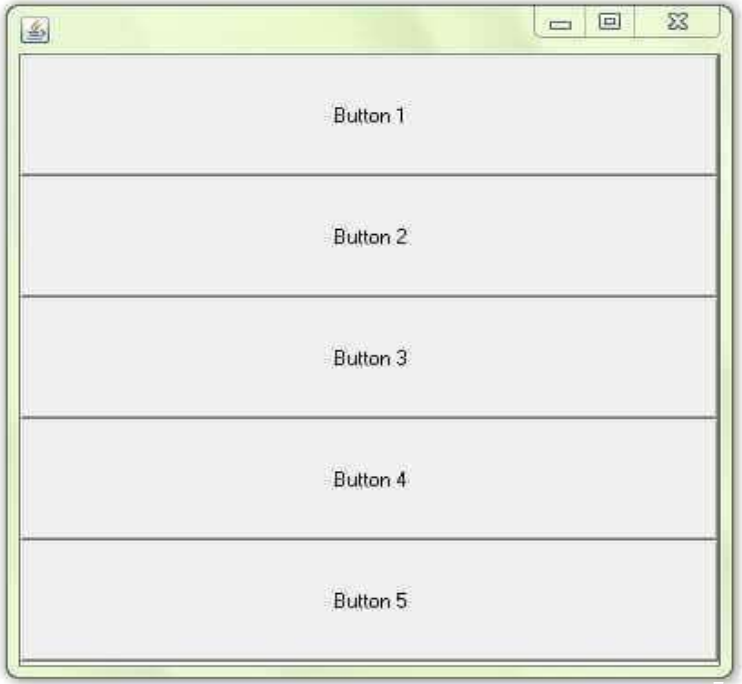
BoxLayout

1. The BoxLayout class puts components in a single row or column.
2. It respects the components' requested maximum sizes and also lets you align components.



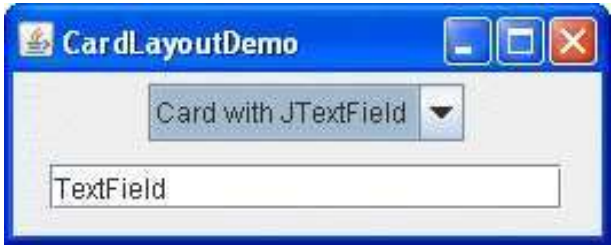
BoxLayout

```
import java.awt.*;
import javax.swing.*;
public class BoxLayoutExample1 extends Frame {
 Button buttons[];
 public BoxLayoutExample1 () {
   buttons = new Button [5];
   for (int i = 0;i<5;i++) {
     buttons[i] = new Button ("Button " + (i + 1));
     add (buttons[i]);
    }
setLayout (new BoxLayout (this, BoxLayout.Y_AXIS));
setSize(400,400);
setVisible(true);
}
public static void main(String args[]){
BoxLayoutExample1 b=new BoxLayoutExample1();
}
}
```



CardLayout

1. The CardLayout class lets to implement an area that contains different components at different times.
2. A CardLayout is often controlled by a combo box, with the state of the combo box determining which panel (group of components) the CardLayout displays.

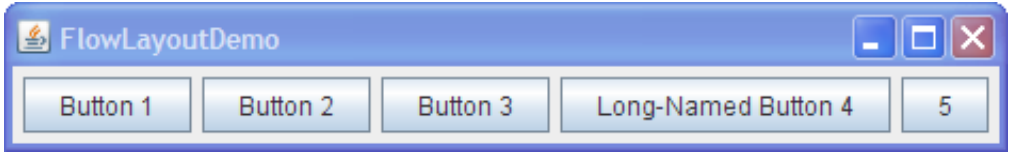


FlowLayout

1. FlowLayout is the default layout manager for every JPanel. It simply lays out components in a single row, starting a new row if its container is not sufficiently wide

Constructors of FlowLayout class

1. **FlowLayout():** creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
2. **FlowLayout(int align):** creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
3. **FlowLayout(int align, int hgap, int vgap):** creates a flow layout with the given alignment and the given horizontal and vertical gap.



FlowLayout

```
public class MyFlowLayout{
JFrame f;
MyFlowLayout(){
    f=new JFrame();
    JButton b1=new JButton("1");
    JButton b2=new JButton("2");
    JButton b3=new JButton("3");
    JButton b4=new JButton("4");
    JButton b5=new JButton("5");
    f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
    f.setLayout(new FlowLayout(FlowLayout.RIGHT));
    //setting flow layout of right alignment
    f.setSize(300,300);
    f.setVisible(true);
}
public static void main(String[] args) {
    new MyFlowLayout();
}
```

GridBag Layout

1. **GridBagLayout** is a sophisticated, flexible layout manager. It aligns components by placing them within a grid of cells, allowing components to span more than one cell. The rows in the grid can have different heights, and grid columns can have different widths.

GridLayout

1. GridLayout simply makes a bunch of components equal in size and displays them in the requested number of rows and columns.

.

Constructors of GridLayout class

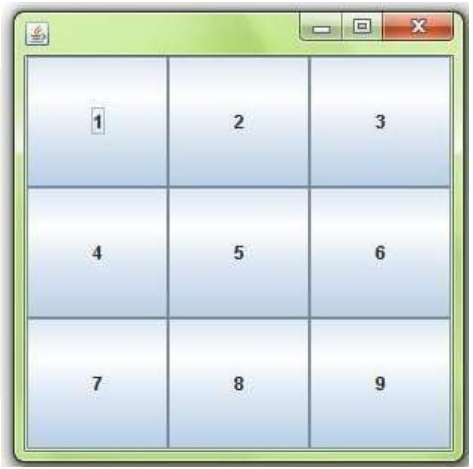
1.GridLayout(): creates a grid layout with one column per component in a row.

2.GridLayout(int rows, int columns): creates a grid layout with the given rows and columns but no gaps between the components.

3.GridLayout(int rows, int columns, int hgap, int vgap): creates a grid layout with the given rows and columns alongwith given horizontal and vertical gaps.

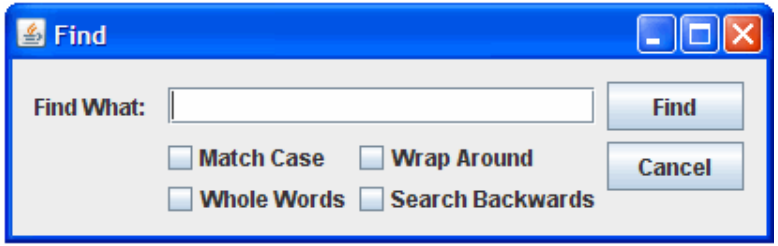
GridLayout

```
import java.awt.*;
import javax.swing.*;
public class MyGridLayout{
JFrame f;
MyGridLayout(){
   f=new JFrame();
   JButton b1=new JButton("1");
   JButton b2=new JButton("2");
   JButton b3=new JButton("3");
   JButton b4=new JButton("4");
   JButton b5=new JButton("5");
   JButton b6=new JButton("6");
   JButton b7=new JButton("7");
   JButton b8=new JButton("8");
   JButton b9=new JButton("9");
   f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
   f.add(b6);f.add(b7);f.add(b8);f.add(b9);
   f.setLayout(new GridLayout(3,3));
   f.setLayout(new GridLayout(3,3));
   f.setSize(300,300);
   f.setVisible(true);
}
public static void main(String[] args) {
   new MyGridLayout();
}
}
```

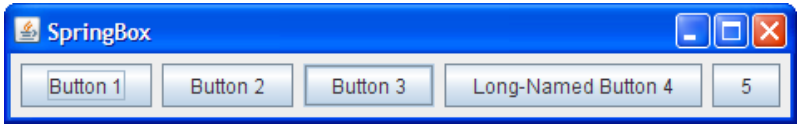


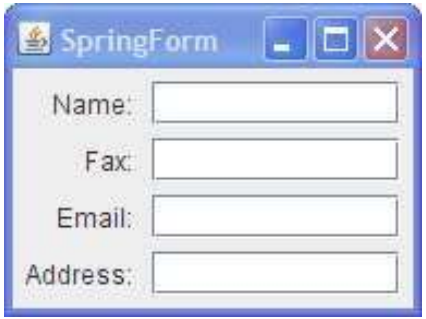
Group Layout

1. GroupLayout is a layout manager that was developed for use by GUI builder tools, but it can also be used manually. GroupLayout works with the horizontal and vertical layouts separately.



Spring Layout





1. SpringLayout is a flexible layout manager designed for use by GUI builders. It lets you specify precise relationships between the edges of components under its control.

Java LayoutManagers

1. java.awt.BorderLayout
2. java.awt.FlowLayout
3. java.awt.GridLayout
4. java.awt.CardLayout
5. java.awt.GridBagLayout
6. javax.swing.BoxLayout
7. javax.swing.GroupLayout
8. javax.swing.ScrollPaneLayout
9. javax.swing.SpringLayout

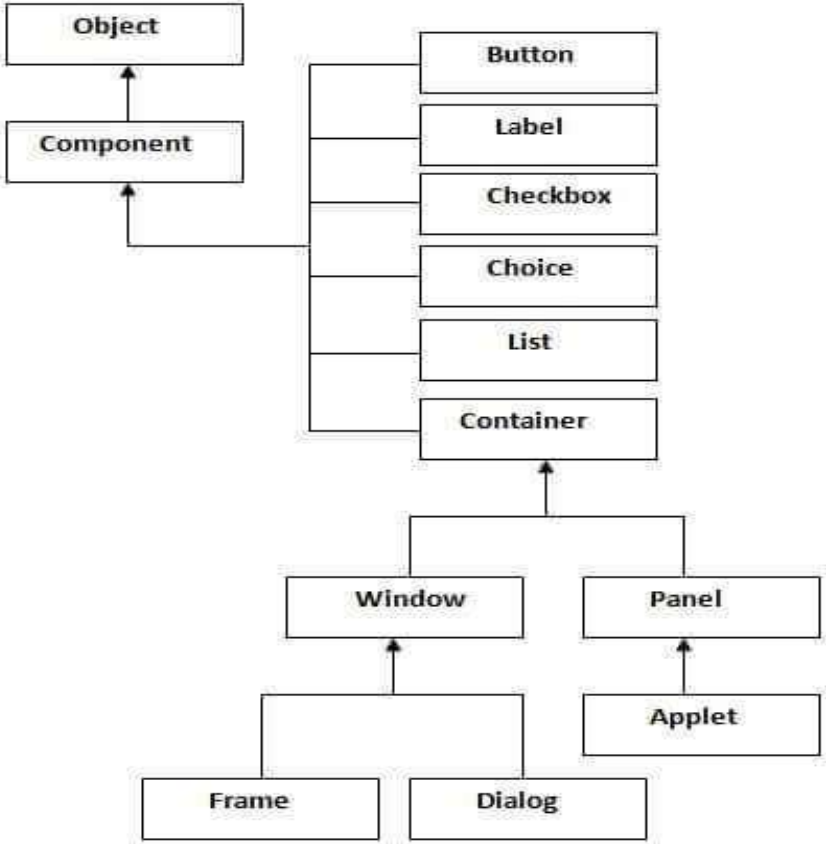
Reference

1. Herbert Schildt " The Complete Reference Java 2, 8th edition , Tata McGraw Hill, 2011
2. Ralph Bravaco, Shai Simonson, "Java Programming: From the Ground up Tata McGraw Hill, 2012
3. https://www.javatpoint.com

INTRODUCTION: AWT

1. **Java AWT** (Abstract Window Toolkit) is an API to **develop GUI or window-based** applications in java.
2. 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 are using the resources of OS.
3. The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

AWT Hierarchy



USING AWT COMPONENTS

- COMPONENT
  - CANVAS
  - SCROLLBAR
  - BUTTON
  - CHECKBOX
  - LABEL
  - LIST
  - CHOICE
  - TEXT
    - TEXTAREA
    - TEXTFIELD

- COMPONENT
  - CONTAINER
    - PANEL
    - WINDOW
      - DIALOG
      - FRAME

- MENU COMPONENT
  - MENU ITEM
    - MENU

AWT Hierarchy

Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such **as Frame, Dialog and Panel**.

Window

The window is the container that have no borders and menu bars. Use frame, dialog or another window for creating a window.

Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

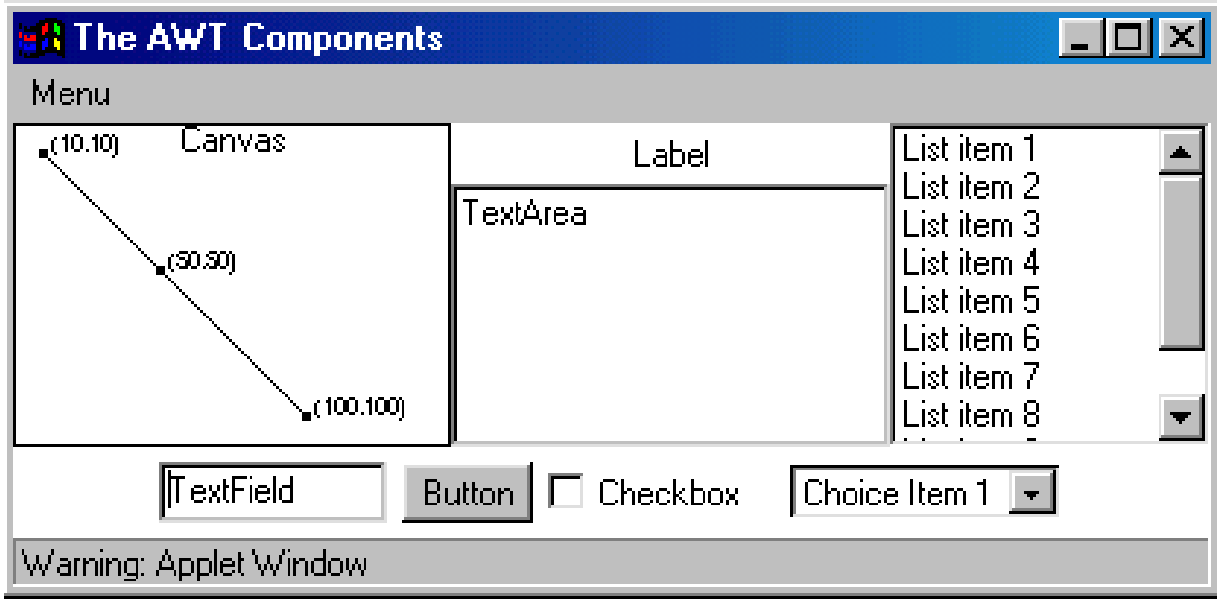
Frame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

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.

AWT COMPONENTS



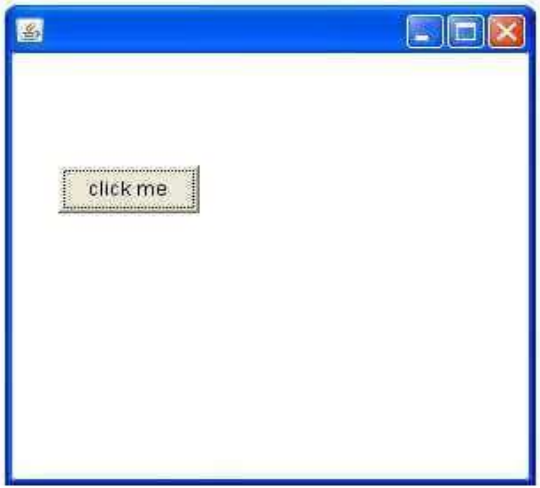
FRAME

```
import java.awt.*;

public class TestFrame extends Frame {
   public TestFrame(String title){
      super(title);
   }
   public static void main(String[] args){
      Frame f = new TestFrame(“The AWT Components");
      f.setSize(400,400);
      f.setLocation(100,100);
      f.show();
   }
}
```

HOW TO USE BUTTONS?

```
import java.awt.*;
class First2
{
First2()
{
        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[]){
        First2 f=new First2();
        }}
```



HOW TO USE LABELS?

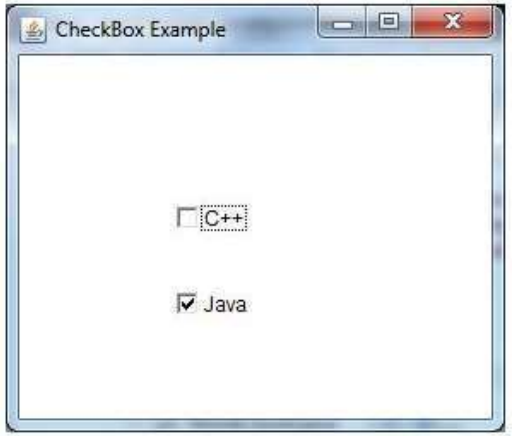
```
import java.awt.*;
class LabelExample{
public static void main(String args[]){
    Frame f= new Frame("Label Example");
    Label l1,l2;
    l1=new Label("First Label.");
    l1.setBounds(50,100, 100,30);
    l2=new Label("Second Label.");
    l2.setBounds(50,150, 100,30);
    f.add(l1); f.add(l2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Output:

HOW TO USE CHECKBOXES?

```
import java.awt.*;
public class CheckboxExample
{
    CheckboxExample(){
      Frame f= new Frame("Checkbox Example");
       Checkbox checkbox1 = new Checkbox("C++");
       checkbox1.setBounds(100,100, 50,50);
       Checkbox checkbox2 = new Checkbox("Java", true);
       checkbox2.setBounds(100,150, 50,50);
       f.add(checkbox1);
       f.add(checkbox2);
       f.setSize(400,400);
       f.setLayout(null);
       f.setVisible(true);
    }
public static void main(String args[])
{
   new CheckboxExample();
}
}
```

Output:



HOW TO USE CHOICES?

```
public class TestChoice extends Frame {
   public TestChoice(String title){
      super(title);

      Choice choice = new Choice();
      choice.add("Item 1");
      choice.add("Item 2");
      choice.add("Item 3");
      choice.add("Item 4");
      choice.add("Item 5");
      add(choice);
   }
```

HOW TO USE TEXTFIELD & TEXTAREA

```
public class TestText extends Frame{
    public TestText(){
        TextField textField = new TextField(20);
        TextArea textArea = new TextArea(5, 20);
        textField.setText("Welcome to Javatpoint");
        textArea.setText("AWT Tutorial");
        add(textField,"North");
        add(textArea,"South");
    }
…
}
```

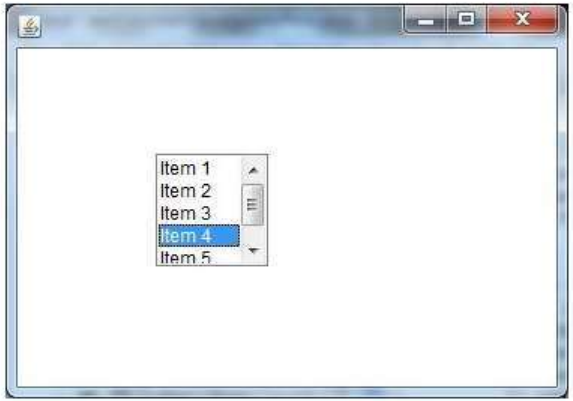
Output:

HOW TO USE LISTS?

```
public class TestList extends Frame {
   public TestList(){
      List l = new List(5, true);
      l.add(“Item1");
      l.add(“Item 2");
      l.add(“Item 3");
      l.add(“Item 4");
      l.add(“Item 5");

   }
```

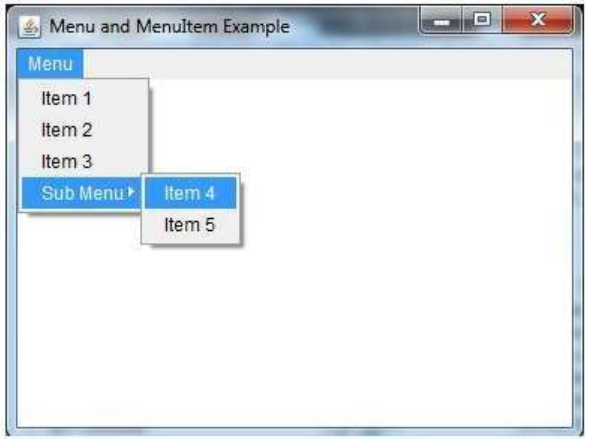
Output:



HOW TO USE MENUS?

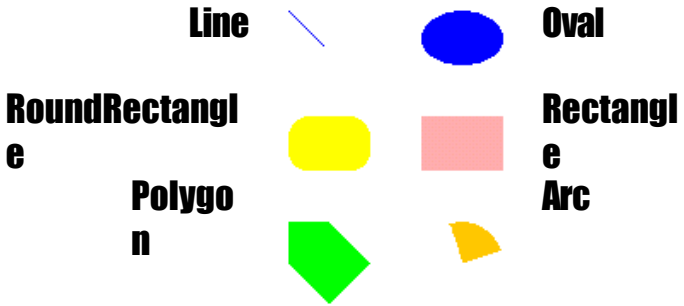
```
import java.awt.*;

 class MenuExample
 {
     MenuExample(){
        Frame f= new Frame("Menu and MenuItem Example");
        MenuBar mb=new MenuBar();
        Menu menu=new Menu("Menu");
        Menu submenu=new Menu("Sub Menu");
        MenuItem i1=new MenuItem("Item 1");
        MenuItem i2=new MenuItem("Item 2");
        MenuItem i3=new MenuItem("Item 3");
        MenuItem i4=new MenuItem("Item 4");
        MenuItem i5=new MenuItem("Item 5");
        menu.add(i1);
        menu.add(i2);
        menu.add(i3);
        submenu.add(i4);
        submenu.add(i5);
        menu.add(submenu);
        mb.add(menu);
        f.setMenuBar(mb);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new MenuExample();
    }
    }
```



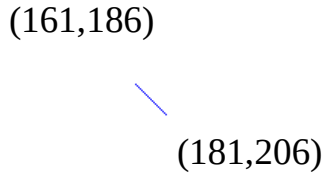
HOW TO USE CANVASES AND GRAPHICS PRIMITIVES?

- FOR DRAWING GEOMETRIC SHAPES, TEXTS, AND IMAGES
- AN ABSTRACT CLASS
  - THE EXTENDED CLASS MUST OVERRIDE **PAINT()**



DRAWLINE(X1,Y1,X2,Y2)

```
class MyCanvas extends Canvas
{
 public void paint(Graphics g)
{
g.setColor(Color.blue);
                        int x1 = 161,
                             y1 = 186,
                             x2 = 181,
                             y2 = 206;
    g.drawLine(x1,y1,x2,y2); }
```



FILLOVAL(X,Y,W,H)
DRAWOVAL(X,Y,W,H)

```
g.setColor(Color.blue);
{
    int x = 239,
        y = 186,
        w = 48,
        h = 32;
      g.fillOval(x,y,w,h);
}
```

FILLPOLYGON(INT[] XS, INT[] YS)
DRAWPOLYGON(INT[] XS, INT[] YS)

```
 g.setColor(Color.green);
{
        int xpoints[] = {25, 145, 25, 145, 25};
        int ypoints[] = {25, 25, 145, 145, 25};
        int npoints = 5;
         g.fillPolygon(xpoints, ypoints, npoints);
}
```

FILLRECT(X,Y,W,H)
DRAWRECT(X,Y,W,H)

```
g.setColor(Color.pink);
{
  int x = 239,
      y = 248,
      w = 48,
      h = 32;
  g.fillRect(x,y,w,h);
}
```

FILLROUNDRECT(X,Y,W,H,RW,RH)
DRAWROUNDRECT(X,Y,W,H,RW,RH)

```
g.setColor(Color.yellow);
{
  int x = 161,
      y = 248,
      w = 48,
      h = 32,
      roundW = 25,
      roundH = 25;
  g.fillRoundRect(x,y,w,h,roundW,roundH);
}
```

FILLARC(X,Y,W,H,SA,A)
DRAWARC(X,Y,W,H,SA,A)

```
g.setColor(Color.orange);
{
  int x = 239,
      y = 310,
      w = 48,
      h = 48,
      startAngle = 20,
      angle = 90;
  g.fillArc(x,y,w,h,startAngle,angle);
}
```

DRAWSTRING, FONT, & FONTMETRICS

```
class MyCanvas extends Canvas {

 public void paint(Graphics g)
 {
  g.setFont(new Font("Dialog",0,20));
  FontMetrics fm = g.getFontMetrics();
 int x = 100;
 int y = 100;
   g.drawString("Hello",x,y);
   y = y+fm.getHeight();
   g.drawString("World",x,y);
 }
}
```

Summary

Reference

1. Herbert Schildt " The Complete Reference Java 2, 8th edition , Tata McGraw Hill, 2011
2. Ralph Bravaco, Shai Simonson, "Java Programming: From the Ground up Tata McGraw Hill, 2012
3. https://www.javatpoint.com