**GridLayout**

A simple layout manager is java.awt.GridLayout.Grid layout decorates components in rectangular grid which adds components to a grid in row-major order (i.e. left to right and top to bottom). It arranges component in cells and each cell has the same size.

The class has a choice of three constructors:

1.GridLayout( ) //Build a grid with 1 row and a column for each added component.

2.GridLayout( int r, int c ) //Build a grid with r rows and c columns.

3.GridLayout( int r, int c, int hgap, int vgap )

// Build a grid with r rows and c columns.

//'hgap' is the space between columns and 'vgap' is the space between rows.

**BoxLayout**

javax.swing.BoxLayout adds components to the container either vertically or horizontally. The components do not wrap as with flow layout.

For example, a horizontal arrangement stays always horizontal even if the container is resized. Here is the class constructor: BoxLayout( Container container, int orientation )

For the first argument, pass the content pane of the container that is to use the box layout. The second argument is an integer code indicating vertical or horizontal orientation.

Two of the Orientation Constants for BoxLayout

\*X\_AXIS Lay components out horizontally from left to right.

\*Y\_AXIS Lay components out vertically from top to bottom.

Here given a runnable manual code to describe the layouts…

**package** layout;

**import** javax.swing.JFrame;

**import** java.awt.Container;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.BoxLayout;

**import** javax.swing.JButton;

**public** **class** GridAndBoxLayoutExample

{

JFrame frame;

JButton button,button1, button2, button3,button4, button5;

**public** GridAndBoxLayoutExample()

{

makeGridLayout();

}

**public** **void** makeGridLayout()

{

frame = **new** JFrame("Grid Layout Frame");

frame.setLayout(**new** GridLayout(2, 3, 10, 10)); //row, column, horizontal gap, vertical gap.

button = **new** JButton("Leader");

button1 = **new** JButton("Member 1");

button2 = **new** JButton("Member 2");

button3 = **new** JButton("Member 3");

button4 = **new** JButton("Member 4");

button5 = **new** JButton("See BoxLayout");

frame.add(button);

frame.add(button1);

frame.add(button2);

frame.add(button3);

frame.add(button4);

frame.add(button5);

frame.setSize(500,400);

frame.setVisible(**true**);

//action listener

button5.addActionListener(**new** ActionListener()

{

@Override

**public** **void** actionPerformed(ActionEvent arg0)

{

frame.setVisible(**false**);

frame = **new** JFrame("Box Layout Frame");

Container box = frame.getContentPane( );

box.setLayout( **new** BoxLayout(box, BoxLayout.***Y\_AXIS***) ); //BoxLayout.X\_AXIS used for horizontal line.

// Test

//System.out.println(BoxLayout.Y\_AXIS); //integer value = 1

//System.out.println(BoxLayout.X\_AXIS); //integer value = 0

button = **new** JButton("Teacher");

button1 = **new** JButton("Student 1");

button2 = **new** JButton("Student 2");

button3 = **new** JButton("Student 3");

button4 = **new** JButton("Student 4");

frame.add(button);

frame.add(button1);

frame.add(button2);

frame.add(button3);

frame.add(button4);

frame.setSize(500,400);

frame.setVisible(**true**);

}

});

} // End of the makeGridLayout Method

//Main method to run the code

**public** **static** **void** main(String[] args)

{

**new** GridAndBoxLayoutExample();

}

}