An Introduction To Graphical User Interfaces

Part 2: You will learn how to arrange or organize graphical controls within a GUI manually and using a layout manager class.

How To Handle The Layout Of Components

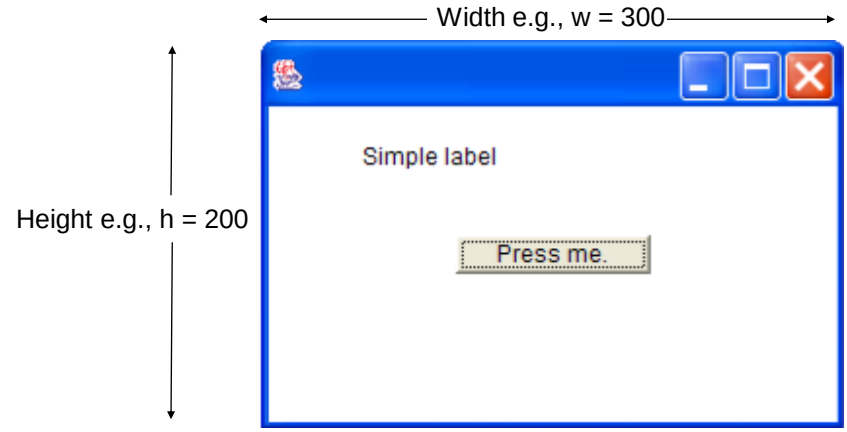
1. Manually set the coordinates yourself
2. Use one of Java's built-in layout manager classes

How To Handle The Layout Of Components

1. **Manually set the coordinates yourself**
2. Use one of Java's built-in layout manager classes

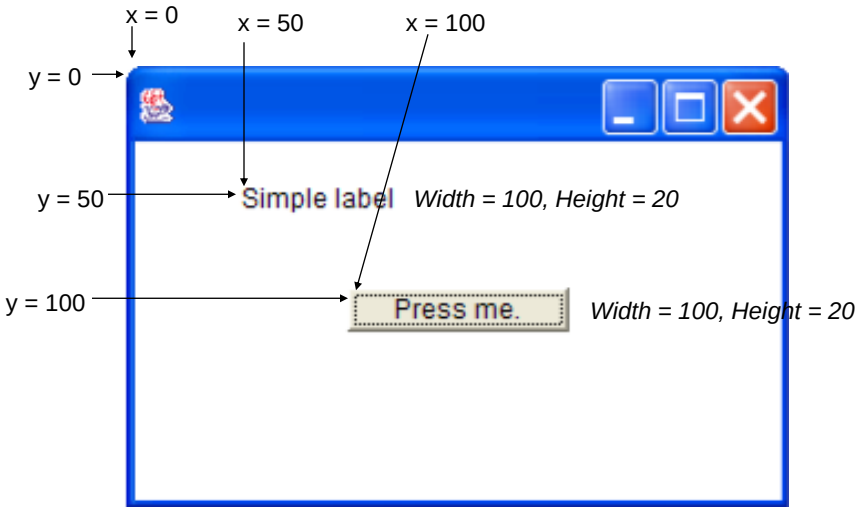
Layout Is Based On Spatial (X,Y) Coordinates

e.g.
```
MyFrame my =new MyFrame();
my.setSize(300,200);
```



Layout Is Based On Spatial Coordinates

Coordinates Of Components: Relative To The Container



Pitfall 2: Invisible `Component`

- Don't forget that coordinates (0,0) are covered by the title bar of the frame.
- `Components` added at this location may be partially or totally hidden by the title bar.

A Example With Manual Layout

- **Name of the folder containing the complete example**: `4manualLayout`

An Example With Manual Layout: The `Driver` Class

```
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JFrame;

public class Driver {
     public static final int WIDTH_FRAME = 300;
     public static final int HEIGHT_FRAME = 300;
     public static final int X_COORD_BUTTON = 100;
     public static final int Y_COORD_BUTTON = 100;
     public static final int WIDTH_BUTTON = 100;
     public static final int HEIGHT_BUTTON = 20;
     public static final int X_COORD_LABEL = 50;
     public static final int Y_COORD_LABEL = 50;
     public static final int WIDTH_LABEL = 100;
     public static final int HEIGHT_LABEL = 20;
```

An Example With Manual Layout: The `Driver` Class (2)

```
     public static void main(String [] args) {
         JFrame aFrame = new JFrame();
         aFrame.setLayout(null);
         aFrame.setSize(WIDTH_FRAME,HEIGHT_FRAME);
         JButton aButton = new JButton("Press me.");
         aButton.setBounds(X_COORD_BUTTON,
                           Y_COORD_BUTTON,
                           WIDTH_BUTTON,
                           HEIGHT_BUTTON);
         JLabel aLabel = new JLabel("Simple label");
         aLabel.setBounds(X_COORD_LABEL,
                          Y_COORD_LABEL,
                          WIDTH_LABEL,
                          HEIGHT_LABEL);
         aFrame.add(aButton);
         aFrame.add(aLabel);
         aFrame.setVisible(true);
     }
}
```

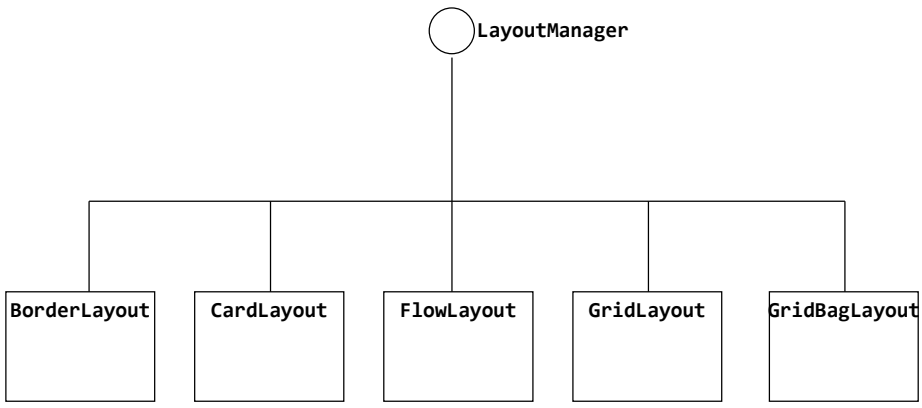
- **To manual layout you make the l manager (null**
- **Don't use automatic manager (manually s**

How To Handle The Layout Of Components

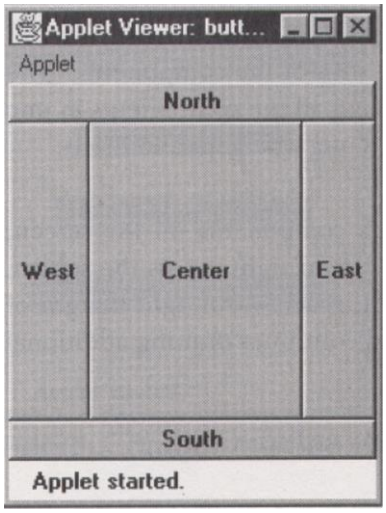
1. Manually set the coordinates yourself
2. **Use one of Java's built-in layout manager classes**

Java Layout Classes

- There are many implementations (this diagram only includes the original classes that were implemented by Sun).

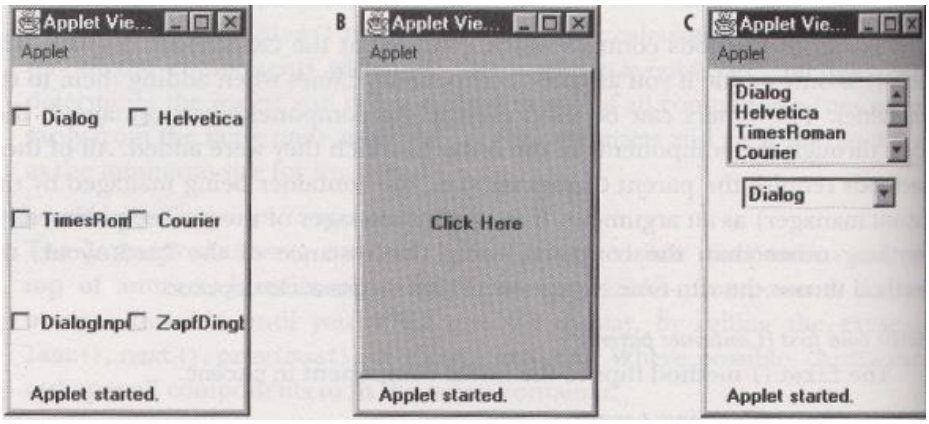


BorderLayout ("Compass Directions")



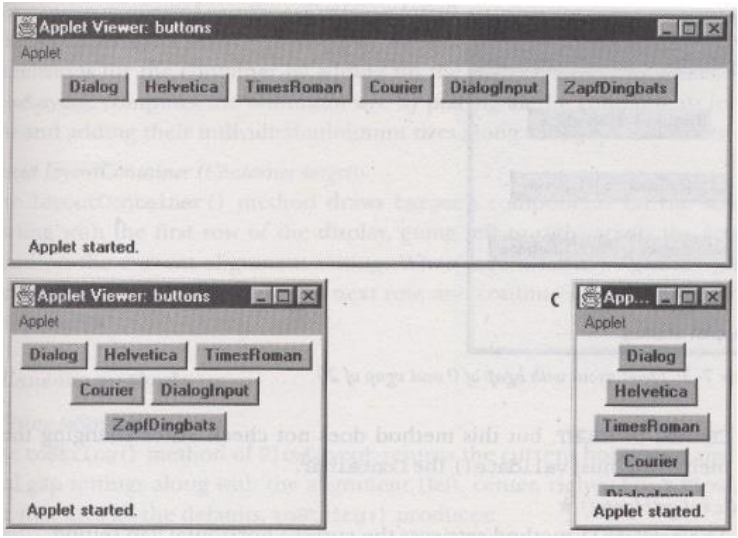
From Java: AWT Reference p. 256

CardLayout ("Tab-Like")



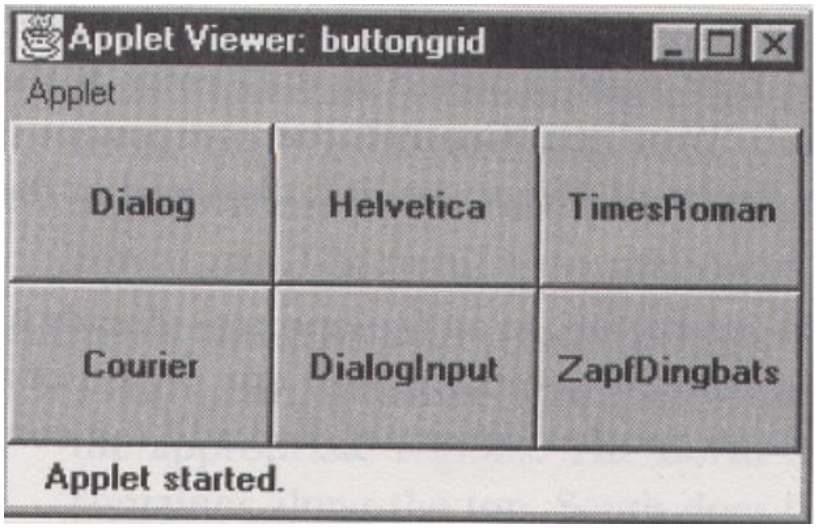
From Java: AWT Reference p. 264

`FlowLayout` (Adapts To Resizing "Web-Like")



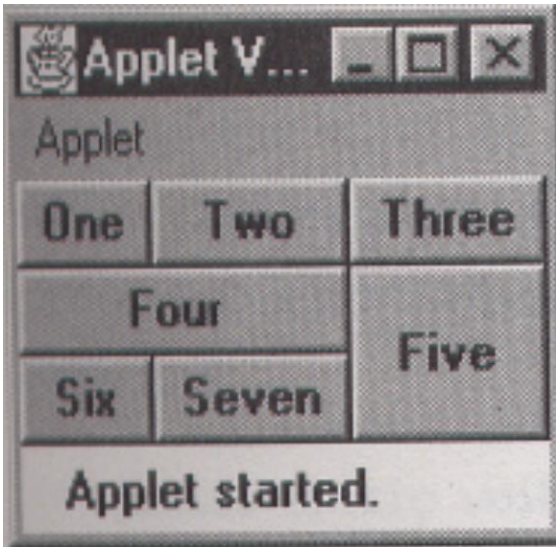
From Java: AWT Reference p. 253

`GridLayout`



From Java: AWT Reference p. 260

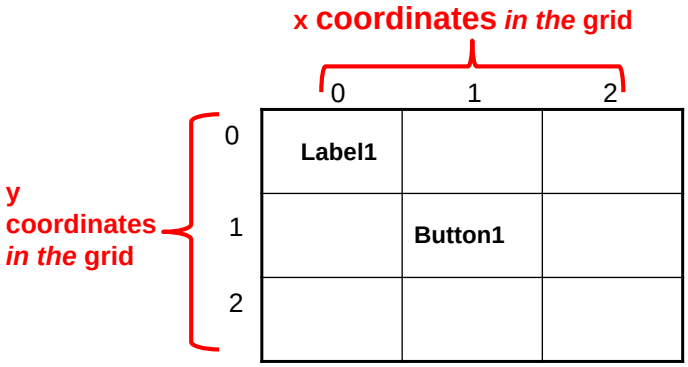
GridBagLayout



From Java: AWT Reference **p. 269**

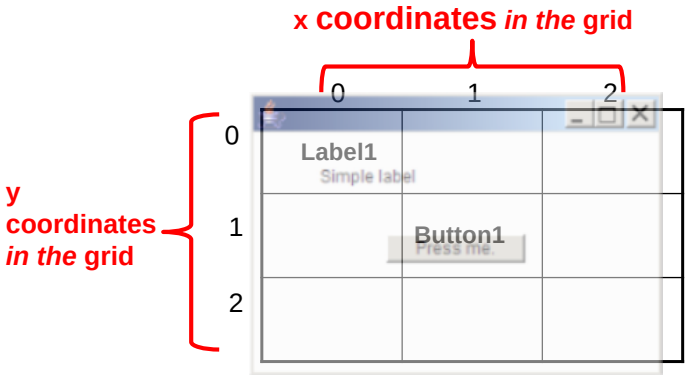
Implementing A GUI When Using The GridBagLayout

- Use graph paper or draw out a table.



Implementing A GUI When Using The GridBagLayout

- Use graph paper or draw out a table.



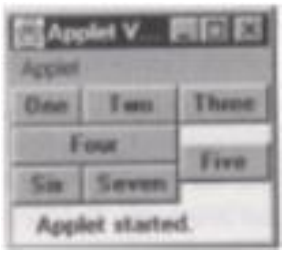
GridBagConstraints

- Goes with the GridBagLayout class.
- Because the GridBagLayout doesn't know 'how' to display components you also need GridBagConstraints to constrain things (determine the layout).
- GridBagConstraints indicates how components should be displayed for a particular GridBagLayout.
- For more complete information see:
  - https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/java/awt/class-use/GridBagConstraints.html

Some Important Parts Of The `GridBagConstraints` Class

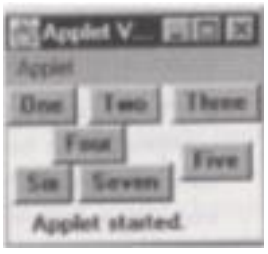
```
public class GridBagConstraints
{
  // Used in conjunction with the constants below to determine
  // the resize policy of the component
  public int fill;

  // Apply only if there is available space.
  // Determine in which direction (if any) that the component
  // expands to fill the space.
  public final static int NONE;
  public final static int BOTH;
  public final static int HORIZONTAL;
  public final static int VERTICAL;
```

`GridBagContraints`: Fill Values



Horizontal

Vertical



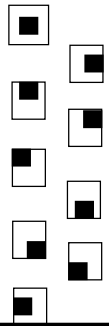
None

Some Important Parts Of The `GridBagConstraints` Class (2)

```
// Position within the grid
public int gridx;
public int gridy;

// Number of grid squares occupied by a component
public int gridwidth;
public int gridheight;
```

Some Important Parts Of The `GridBagConstraints` Class (3)

```
// Used in conjunction with the constants below to determine
// that the component drift if the space available is larger
// than the component.
public int anchor;

// Only if the component is smaller than the available space.
// Determine the anchor direction
public final static int CENTER;
public final static int EAST;
public final static int NORTH;
public final static int NORTHEAST;
public final static int NORTHWEST;
public final static int SOUTH;
public final static int SOUTHEAST;
public final static int SOUTHWEST;
public final static int WEST;
```



Some Important Parts Of The `GridBagConstraints` Class (4)

```
// With a particular 'cell' in the grid this attribute
// specifies the amount of padding around the component
// to separate it from other components.
// Usage:
// insets = new Insets(<top>,<left>,<bottom>,<right>);
// Example (Set top, left, bottom, and right)
// insets = new Insets(0, 0, 0, 0); // No padding (default)
public insets;
```

Insets = 0: no padding

Insets = 10: many spaces/padding

An Example Using The `GridBagLayout`

- **Name of the folder containing the complete example**: : `5gridbaglayout`

An Example Using The GridBagLayout: The Driver Class

```
public class Driver
{
     public static final int WIDTH = 400;
     public static final int HEIGHT = 300;
     public static void main(String [] args)
     {
         MyFrame aFrame = new MyFrame ();
         aFrame.setSize(WIDTH,HEIGHT);
         aFrame.setVisible(true);
     }
}
```

An Example Using The GridBagLayout: Class MyFrame

```
public class MyFrame extends JFrame {
    private JButton left;
    private JButton right;
    private JLabel aLabel;
    private GridBagLayout aLayout;
    GridBagConstraints aConstraint;

    public MyFrame() {
        MyWindowListener aWindowListener = new MyWindowListener();
        addWindowListener(aWindowListener);
        aConstraint = new GridBagConstraints();
        Scanner in = new Scanner(System.in);
        System.out.print("Buffer size to pad the grid: ");
        int padding = in.nextInt();
```

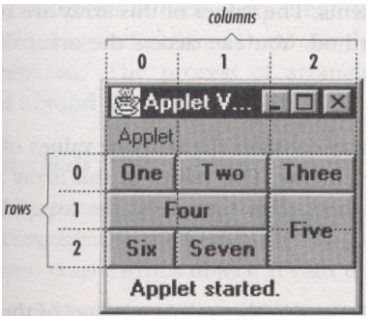
An Example Using The GridBagLayout: Class MyFrame (2)

```
        left = new JButton("L: Press me");
        right = new JButton("R: Press me");
        MyButtonListener aButtonListener = new MyButtonListener();
        left.addActionListener (aButtonListener);
        right.addActionListener (aButtonListener);
        aLabel = new JLabel("Simple label");
        aConstraint.insets = new
           Insets(padding,padding,padding,padding);
        aLayout = new GridBagLayout();
        setLayout(aLayout);     // Calling method of super class.
        addWidget(aLabel, 0, 0, 1, 1);
        addWidget(left, 0, 1, 1, 1);
        addWidget(right, 1, 1, 1, 1);
    }
```

An Example Using The GridBagLayout: Class MyFrame (3)

```
    public void addWidget(Component widget,
                          int x,
                          int y,
                          int w,
                          int h)
    {   // aConstraint: refers to instance of
        // GridBagConstraints
        aConstraint.gridx = x;
        aConstraint.gridy = y;
        aConstraint.gridwidth = w;
        aConstraint.gridheight = h;
        aLayout.setConstraints(widget, aConstraint);
        add(widget);        // Calling method of super class.
    }
} // End of definition for class MyFrame
```

Advanced Uses Of `GridBagLayout`



Button	gridx (col)	gridy (row)	grid-width	grid-height
One	0	0	1	1
Two	1	0	1	1
Three	2	0	1	1
Four	0	1	2	1
Five	2	1	1	2
Six	0	2	1	1
Seven	1	2	1	1

From Java: AWT Reference p. 269

Layout Of GUI Components

- JT's note (and opinion): learning how to layout GUI components manually will teach you "how things work".
  - That's because you have to handle many details yourself (either manually or by using a layout class).
  - Except when writing small programs with a simple GUI (assignment) doing things manually is just too much of a hassle.
    - The programmer focuses on the wrong details (how do I get the programming language to 'do stuff' as opposed to how do I create a GUI that is 'user-friendly').
  - In other cases ('real life programs') an IDE is used.
  - Some examples:
    - Sun's NetBeans IDE: http://docs.oracle.com/javase/tutorial/uiswing/learn/index.html
    - IBM's Eclipse IDE: http://www.ibm.com/developerworks/opensource/library/os-ecvisual/

After This Section You Now Know

- How to manually layout graphical components within a GUI (as specified by the (x, y) pixel coordinates.
- How to use the layout manager class `GridBagLayout` in conjunction with `GridBagConstraints` to specify layout on a grid and to automated update layout when the container is resized.

James Tam

Copyright Notice

- Unless otherwise specfied, all images were produced by the author (James Tam).

James Tam