Class 17 Notes

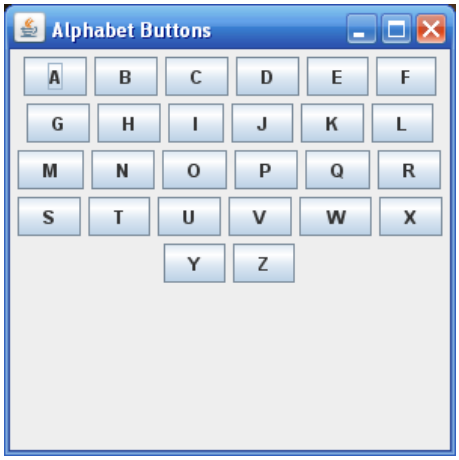
So you know how BorderLayout places components in a frame. Now we look at a different layout manager -- FlowLayout. FlowLayout places components in a frame one after the other, left to right. There is no NORTH, SOUTH, EAST, WEST, or CENTER. And, FlowLayout can place as many components in the frame as will fit.

So here goes…Introducing FlowLayout our new frame decorator.
Named after Flo..the Progressive lady

FlowLayout

The FlowLayout manager arranges components horizontally in a container (e.g. JFrame), left to right, row by row, in the order in which they are added to the container.

Here 26 buttons have been added and the FlowLayout manager just placed one after the other row by row.



The FlowLayout class has three constructors:

- FlowLayout()
  instantiates a FlowLayout layout manager object that center aligns components in a container.
  The buttons above are center aligned

- FlowLayout(int Alignment)

  instantiates a FlowLayout layout manager with the specified alignment: FlowLayout.LEFT, FlowLayout.CENTER, or FlowLayout.RIGHT.

For example, new FlowLayout(FlowLayout.LEFT)
This just like using Word where text is left aligned, right aligned or centered. It is the same with components…They can be left aligned, right aligned or centered. In the picture above they are centered. **CENTER is the default.**

- FlowLayout(int Alignment, int horizontalSpace, int verticalSpace)
  instantiates a FlowLayout layout manager with the specified alignment. Parameters horizontalSpace and verticalSpace specify horizontal and vertical space, in pixels, between components.
  Example: new FlowLayout(Flowlayout.LEFT, 10,10). Each component will be separated by 10 pixels

Now we saw that when you add components to a frame, BorderLayout is the default. Suppose that we do not want BorderLayout to place our components but instead we want to add components using the FlowLayout manager. OK we can just fire the BorderLayout manager and hire the FlowLayout manager.

In the frame constructor include the statements:

FlowLayout flow = new FlowLayout(); // create a FlowLayout manager named flow
setLayout(flow) // makes flow the manager of the frame (fires BorderLayout :( )
or
this.setLayout(flow)

You can also do the same in one statement:
setLayout(new FlowLayout()) ;

If you want a particular alignment you can use
FlowLayout flow = new FlowLayout(FlowLayout.LEFT), for example

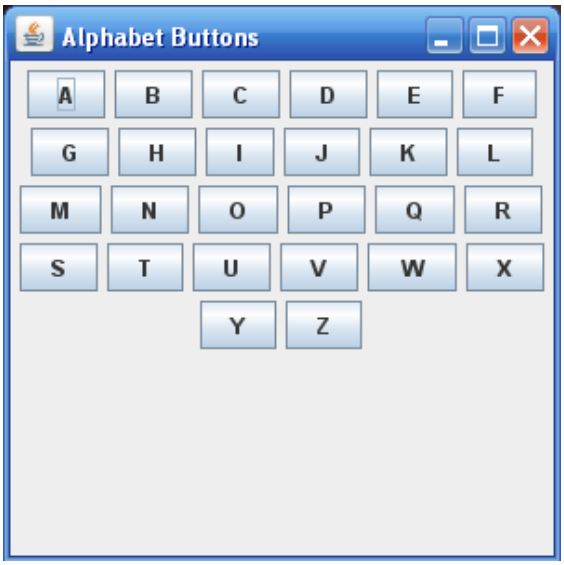
Example
Create a class **AlphabetFrame that extends JFrame**. The AlphabetFrame is a container that holds 26 buttons labeled with the letters of the alphabet. See the picture below

IMPORTANT:

- **A JButton can display a String or a picture.**
- **You cannot place a char or int on a JButton.**
- **The JButton constructor takes a String or an image as an argument.**
- **The ASCII value for 'A' is 65 and for 'Z' is 90. This will be used in the following program**

```
1.  import java.awt.*;
2.  import javax.swing.*;

3.  public class AlphabetButtons extends JFrame
4.  {
5.      public AlphabetButtons()  // this is the constructor
6.      {
7.          super("Alphabet Buttons");       // call one argument constructor of JFrame (parent)

8.          FlowLayout flow = new FlowLayout(); // make a new manager
9.          setLayout(flow)      // layout manager is now FlowLayout—BorderLayout  is fired

10.         setBounds(0,0, 300, 300);     // frame is placed at (0,0) and is 300 x 300
11.         for (int i = 'A' i < ='Z'; i++)    // loop through the ascii value from A(65)  to Z(90)
12.         {
13.             char  letter = (char)(i );                    // cast to a char 65 is A, 66—B…90--Z
14.             JButton button = new JButton(letter+"");   // Make a button. Notice the ""  makes a  cast to STRING
15.             add(button);                                 // place the button…you can also say  this.add(button)
16.         }
17.         setVisible(true);  // VERY IMPORTANT
```

```
18.
19.     }


20.         public static void main(String[] args)
21.         {
22.             JFrame frame = new AlphabetButtons();
23.
24.         }
25.     }
```

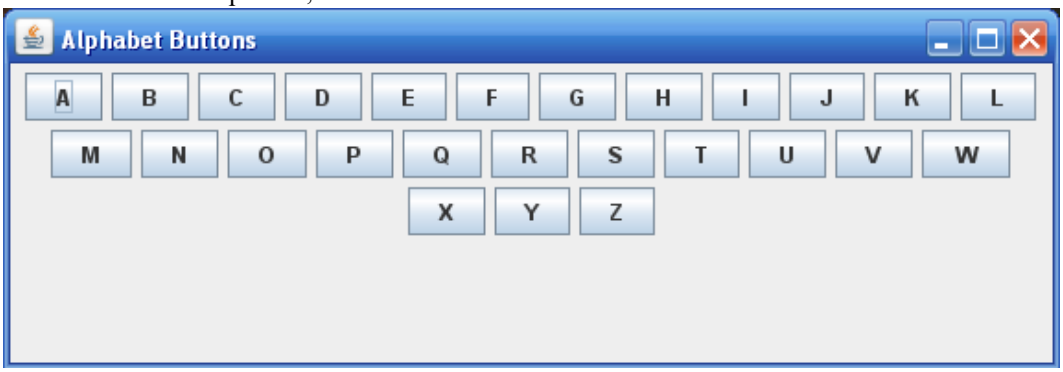


Twenty-six buttons placed with *FlowLayout*

The JButton constructor requires a **String (or image)** parameter
Notice the loop variable i goes from 65 (ASCII 'A') to 90 ('ASCII 'Z').
So if i is 65 then letter = (char)i is 'A', if i is 66 then letter = (char)i is 'B' etc.
and **letter +"" is a STRING**…which is a valid button label

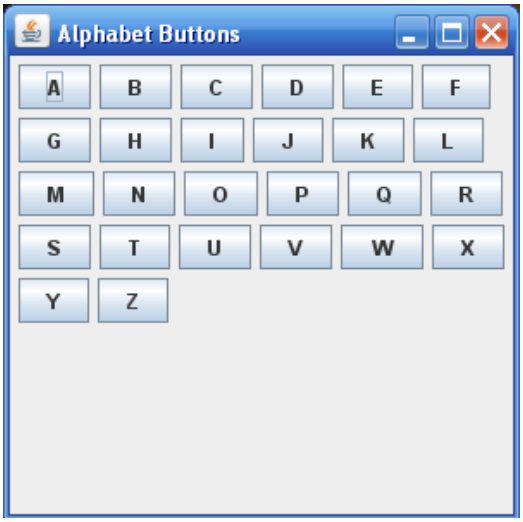
An *AlphabetButtons* frame of size 100 by 100 some are hidden

When the frame is expanded, all buttons are visible in three rows.



A full width frame that fills the screen.

If we use setLayout(newFlowLayout(FlowLayout.**LEFT**);
the buttons in each row are left justified and the frame appears as



***FlowLayout* with LEFT alignment**

(The alignment feature may not work on a Mac. Another quirk!)

You should enter this program and run it. Then drag the frame around the screen making it larger and smaller see how the buttons are placed.
Make sure you understand every line of code before moving on. Again, you have to instantiate a JButton with a String or an image. You can cast an int or char to a String by adding the empty string : 456+ "".

GridLayout Manager

BorderLayout and FlowLayout both have their own way of decorating or placing components in a frame. A third layout manager is GridLayout. The GridLayout manager thinks of the frame as a two dimensional array and places components row by row.

> The GridLayout layout manager arranges the components of a frame in a grid of specified dimensions, left to right, top to bottom, row by row….**like a two dimensional** array

The constructors of GridLayout are:

- GridLayout(int rows, int columns)
  where rows and columns specify the number of rows and columns in the grid.
  For Example GridLayout grid = new GridLayout(5,7). The frame will have 5 rows and 7 columns

- GridLayout(int rows, int columns, int horizontalSpace, int verticalSpace)
  where rows and columns specify the number of rows and columns in the grid and horizontalSpace and verticalSpace are the horizontal and vertical gaps between components.

- GridLayout()
  creates a grid with a single row and a column for each component. Not particularly useful

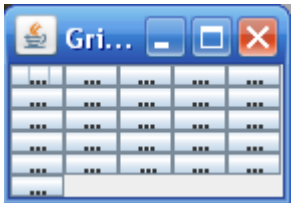
A JFrame uses BorderLayout by default but you can replace BorderLayout with GridLayout
The following example uses GridLayout to place 26 alphabet buttons in a frame.
It is pretty much identical to the last example but with a different layout manager—just 2 lines are changed

Example
Place 26 "alphabet buttons" in a frame using GridLayout. The grid should have 6 rows and 5 columns.
The program is identical to the last one except for the layout manager….grid vs flow
But the picture is different

```
1.  import java.awt.*;
2.  import javax.swing.*;
3.  import java.util.*;

4.  public class GridAlphabetButtons extends JFrame
5.  {
6.      public GridAlphabetButtons ()        // default  constructor
7.      {
8.          super("Grid Layout  Alphabet Buttons");

9.          GridLayout grid = new GridLayout(6, 5);    // make a GridLayout manager
10.         setLayout(grid );      // set the layout manager to grid ( and fire BorderLayout)

11.         setBounds(0,0, 300,300);

12.         for (int i = 'A' i <='Z'; i++)  // same as the last example
13.         {
14.             char letter  = (char)i;
15.             JButton button = new JButton(letter+"");
16.             add(button);
17.         }
18.          setVisible(true);
19.
20.     }
21.     public static void main(String[] args)
22.     {
23.         JFrame frame = new GridAlphabetButtons ();
24.
25. }
```

A frame created with *GridLayout*

In contrast to the FlowLayout, a frame of size100 by 100 resizes and displays all 26 buttons but not very nicely. The letters in the buttons do not resize and are too small to be viewable, but all the buttons do appear. **You will often have to play around with the size to get the picture that you want.**



***GridLayout* frame of size 100 by 100**

You should type this one in and run it.

Placing Components in a Frame Without a Layout Manager

OK ..Border, Flow, Grid…They each have their own way of decorating a frame.
However,---

- **You don't need to use a layout manager at all to place components.**
- **You can place a component anywhere in a frame by designating the frame coordinates of where you want place a component.**
- **If you place a button at position (50,75) the upper left corner of the button is at position (50,75) of your frame.**
- **Of course, you have to disable (fire!) the BordeLayout manager to do this.**

By default, a frame uses the BorderLayout layout manager. To disable the default layout manager and place components in a frame without any assistance,

set the layout manager of the frame to null, using setLayout(null).

Now there will be no layout manager associated with the frame. (So you're too cheap to hire a decorator!)

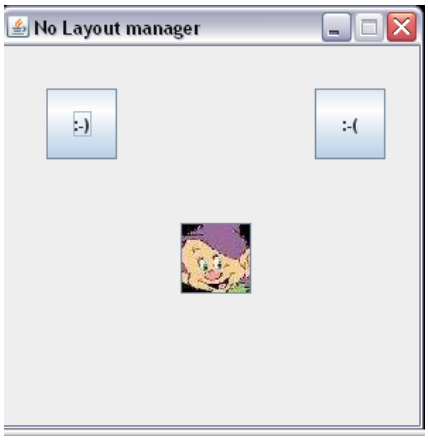
Example Here we use no layout manager but place three buttons at specified locations. See the picture below

Place three buttons, each of size 50 by 50, in a frame of size 300 by 300 such that:

- the first button is placed at position (30,30),
- the second button is placed at (220,30), and
- the third button is placed at (125,125).

```
1.  import javax.swing.*;
2.  import java.awt.*;

3.  public class NoLayoutManager extends JFrame
4.  {

5.      public NoLayoutManager()
6.      {
7.          super("No Layout manager");   //  call  one argument constructor of parent JFrame

8.          setLayout(null);                  // no layout manager – disable BorderLayout ( or this.setLayout(null)

9.          setBounds(0,0, 300,300);       // position and size

10.         // create the three buttons
11.         JButton picture = new JButton( new ImageIcon("dopey.jpg"));
12.         JButton smile  = new JButton (":-)");
13.         JButton frown = new JButton (":-(");

14.         // set the position and size of each button notice we use setBounds
15.         picture.setBounds(125,125,50,50);          //  position in the frame is (125,125) size is 50 x 50
16.         smile.setBounds(30,30, 50,50) ;            //  position in the frame is (30,130) size is 50 x 50;
17.         frown.setBounds(220,30,50,50) ;            //  position in the frame is (220,30) size is 50 x 50

18.         // add each button to the frame
19.         add(picture);// or this.add(picture)
20.         add(smile);
21.         add(frown);
22.         setResizable(false);  // cannot resize the frame
23.         setVisible(true);
24.     }
```

```
25.     public static void main(String[] args)
26.     {
27.         JFrame frame = new NoLayoutManager();
28.     }
29. }
```



A frame created without a layout manager

The frame cannot be resized. . If the frame were, in fact, resizable, i.e., setResizable(true), then after expanding the frame, the three buttons would not resize and their positions in the expanded frame would remain the same and in the same positions .

A resizeable *NoLayoutManager* frame maximized

Labels

A ***JLabel*** is a component that displays text and/or an image. In contrast to a button-- which can be "clicked" -- a label is a component that is used primarily to *display* a string or an image.

A JLabel does nothing but display information—a String or a Picture.

The constructors are:

- **JLabel()**
  // creates a label with nothing displayed

- **JLabel (String s)**
- // creates a label with a string
  JLabel label = new JLabel("I am a label");

- **JLabel(newImageIcon (String s));**
- // creates a JLabel with a picture
  JLabel picture = new JLabel(new ImageIcon("Homer.jpg"));

- **JLabel (String s, horizontalAllignment)**
  // how we want the text on the label aligned
  JLabel label = new JLabel("I am a label", **SwingConstants.LEFT**);
  or
  JLabel label = new JLabel("I am a label", **SwingConstants.RIGHT**);
  or
  JLabel label = new JLabel("I am a label", **SwingConstants.CENTER);**

Java has provided many methods for JLabel but two very useful are:
SetText(String s) and setIcon(ImageIcon x)
For example

```
JLable label = new JLabel("ABC")      // creates a label with "ABC"
Label.setText("DEFG");                // now the label displays "DEFG"
```

Adding labels to a JFrame is really no different than adding JButtons.

Here is a previous example , Alphabuttons.java, rewritten so it adds JLabels rather than JButtons. I just substituted JLabel for JButton. That was the only change.

```
1.  import java.awt.*;
2.  import javax.swing.*;

3.  public class AlphabetLabels extends JFrame
4.  {

5.      public AlphabetLabels()
6.      {
7.          super("Alphabet Labels");                  // cal1one argument constructor of JFrame
8.          FlowLayout flow = new FlowLayout();        // make a new manager
9.          setLayout(flow)                            // layout manager is now FlowLayout
10.         setBounds(0,0, 300, 300);                  // placed at (0,0) and 300 x 300
11.
12.         for (int i = 65; i < =90; i++)             // ascii valuesfrom A(65)  to Z(90)
13.         {
```

```
14.             char  letter = (char)(i );                  // cast to a char 65 –A, 66—B…90—Z

15.             JLabel label = new JLabel(letter+"");       // Notice the "" is  a cast to STRING
16.             add(label);                                  // or this.add(label)

17.         }
18.         setVisible(true);
19.
20.     }

21.         public static void main(String[] args)
22.         {
23.             JFrame frame = new AlphabetLabels();
24.
25.         }
26.     }
```

The frame looks bit different since the labels are sized smaller. They are sized to fit the text
Here is the frame:

If, after line 17 I added,

```
JLabel numbers = new JLabel("0123456789");
add(numbers);
```

The frame would appears as

Here the last label is sized to hold the whole string "0123456789"

You can make the labels a different or fixed size (e.g. 50 x 50) by adding the statement

label.setPreferredSize(new Dimension(50,50)); // width x height

after line 15. It will not change the font size but it will change the label size:

Later you will see how to change the font size.
So adding a JLabel is really no different that adding a JButton. The frame may look different but the method is pretty much the same.