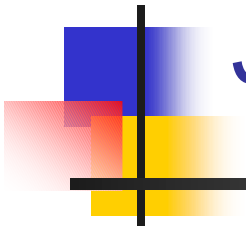
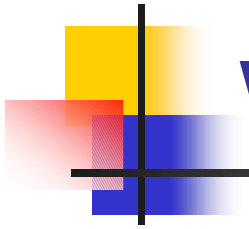


Java 2 For Beginners

Day 3

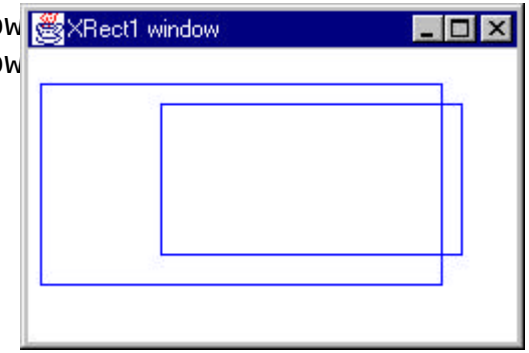


We wrote a 2 Rectangle Program

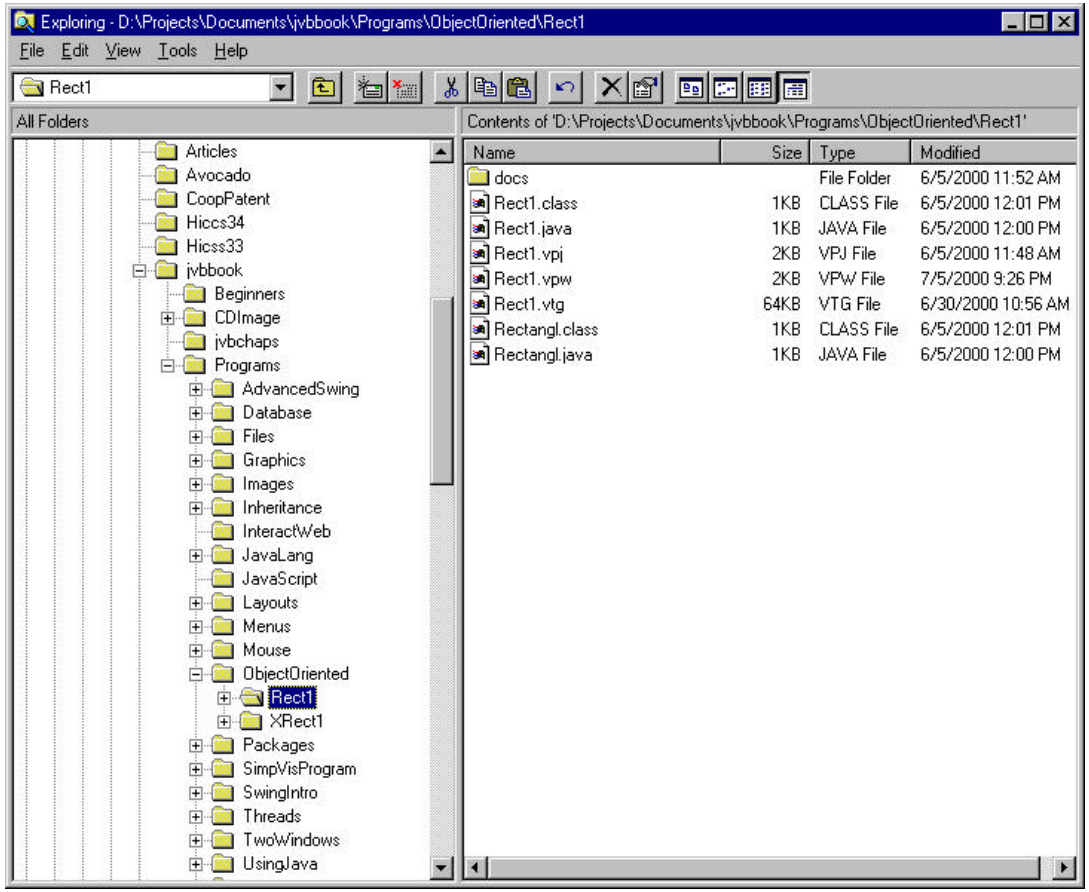
- But it would not exit on the close box click.
- We can make it do so by using Xframe class.
- Written for this class, not in base Java toolkit.
  - You must include a copy of Xframe.java in every program folder to use it.

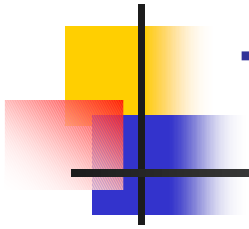
Revised Program

```
//Simple Rectangle drawing example
public class XRect1 extends XFrame {
    private Rectangl rect1;             //two rectangle objects
    private Rectangl rect2;
//-----------------------------------------
    public XRect1() {
        super("XRect1 window");  //create window with title bar
 //Create rectangles and tell them where to draw
        rect1 = new Rectangl(10, 40, 200, 100);
        rect2 = new Rectangl(70, 50, 150, 75);

        setBounds(50, 50, 475, 225);      //size of window
        setVisible(true);                 //display window
    }
//-----------------------------------------
    public void paint(Graphics g) {
        rect1.draw(g);
        rect2.draw(g);
    }
//-----------------------------------------
    public static void main(String args[]) {
        new XRect1();    //create instance of XRect1 class
    }
}
```



All of the code we use is on CD in zip file

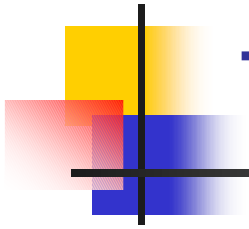




The Integer and Float classes

- Object versions of basic int and float
- Have useful conversion methods

```
Integer intClass = new Integer(12);
   int i = intClass.intValue();
   String si =intClass.toString();
Float fltClass = new Float(12.3);
   float t = fltClass.floatValue();
   String st = fltClass.toString();
```



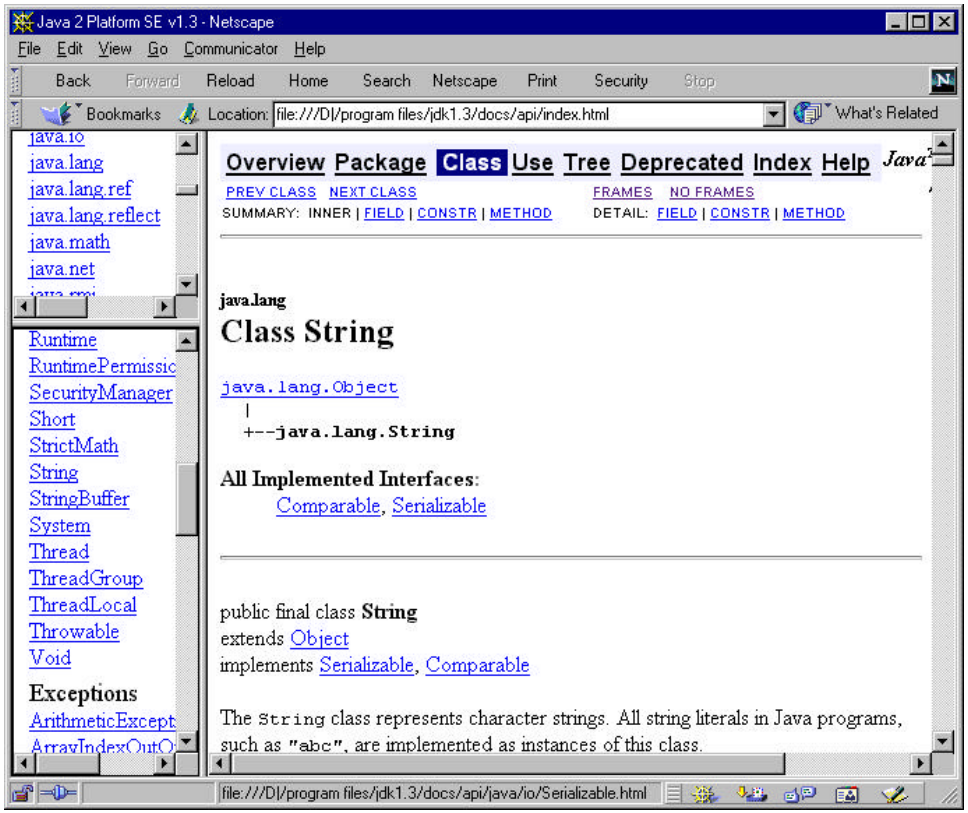
The String class

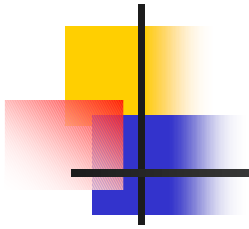
- Contains 0 or more characters

```
String s = “Hello there”;
int i = s.indexOf(“ “);
String hlo = s.substring(0, i);
String thr = s.substring(i + 1);
String Hlo = hlo.toUpperCase();
String Llo = hlo.toLowerCase();
if(thr.equals(“there”))
   System.out.println(“they are equal);
if(Hlo.equalsIgnoreCase(“hello”))
   System.out.println(“foo”);
```

String has many useful methods

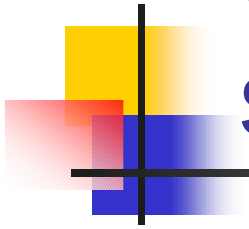
- Easiest to look them up in docs





Inheritance

- This technique works because Java is an object-oriented language where ***inheritance*** is commonly used.
- Java classes can only inherit from one parent class.
  - Multiple inheritance not allowed
- Inheritance allows you to derive new classes from existing ones
  - and only write code for those parts that are changed



Suppose we want to draw squares

- Squares are a special case of Rectangl
- We should only have to write some simple code to do this.

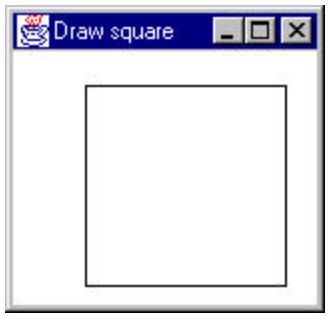
Rectangl and Square

```
import java.awt.*;
import java.applet.*;

public class Rectangl {
    private int width, height;
    private int xpos, ypos;
    private Color color;

    public Rectangl(int sp, int yp,
                    int w, int h)  {
        xpos = yp;         //save posn
        ypos = yp;
        width = w;        //save shape
        height = h;
    }
//-----------------------------------
    public void draw(Graphics g) {
      g.drawRect(xpos,ypos,
          width, height);
    }
}
```

```
public class Square extends Rectangl {
public Square(int xp, int yp, int side) {
        //sides are the same size
        super(xp, yp, side, side);
    }
}
```

Drawing the Square

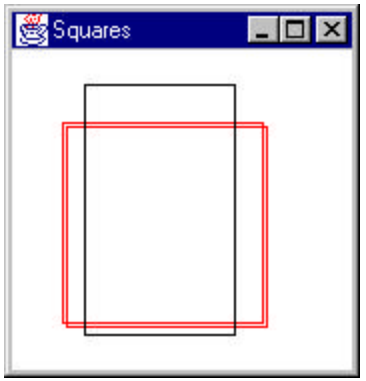
```
public class Sqr extends XFrame {
    Square s1;  //one instance of the square
    public Sqr()    {    //constructor
        super("Draw square");
        setBounds(100, 150, 150, 150);
        s1 = new Square(20, 40, 100); //create square
        setVisible(true);
    }
//-----------------------------------
    public void paint(Graphics g) {
        s1.draw(g);
    }
    static public void main(String argv[]) {
        new Sqr();
    }
}
```

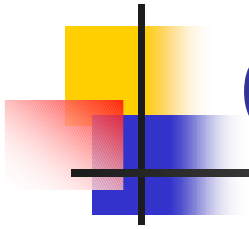


A Square and a Rectangl

- Now very simple

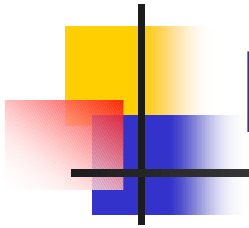
```
public class Sqr2 extends XFrame {
    Square2 s1;  //one instance of the square object
    Rectangl r1;

    public Sqr2()       { //constructor
        super("Squares");
        s1 = new Square2(30, 60, 100);        //create square
        r1 = new Rectangl(40,40, 75,125);
        setBounds(100,100, 300,200);
        setVisible(true);
    }
//---------------------------------
    public void paint(Graphics g) {
        s1.draw(g);
        r1.draw(g);
    }
    static public void main(String argv[]) {
        new Sqr2();
    }
}
```





Classes and Methods

- Every class can have public and private functions.
  - Called ***methods*** in Java parlance
  - public methods called from outside class
  - private methods called only inside class
  - You almost always call a class's methods
    - rather than calling an external function on class data
    - r1.draw; not draw(r1);

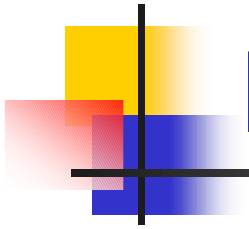


Rectangl public methods

```
public class Rectangl {
    private int width, height;
    private int xpos, ypos;
    private Color color;

    public Rectangl(int x, int y, int w, int h)    {
        xpos = x;         ypos = y;
        width = w;        height = h;
        color = Color.black;   //default color
    }
//-----------------------------------
    public void move(int x, int y) {
        xpos = x;      ypos = y;
    }
//-----------------------------------
    public void setColor(Color c) {
        color = c;
    }
//-----------------------------------
    public void draw(Graphics g) {
        g.setColor(color);     //set to curent color
        g.drawRect(xpos,ypos, width, height);  //draw rectangle
    }
}
```

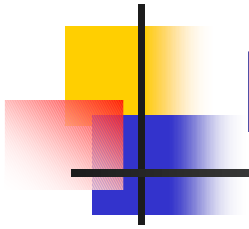
Then our square can be a different width and color

```
public class Square2 extends Rectangl {
    private int xpos, ypos;     //saved copies

//This square class is derived from Rectangl
    public Square2(int x, int y, int side) {
        super(x, y, side, side); //sides are the same size
        xpos = x;
        ypos = y;
        setColor(Color.red);   //all squares are red
    }
//-----------------------------------
    public void draw(Graphics g) {     //overrides Rectangle Draw
        super.draw(g);          //draw in original position
        move(xpos+1, ypos+1);  //transpose one pixel
        super.draw(g);          //draw it again
        move(xpos-1, ypos-1);  //move it back
    }
}
```



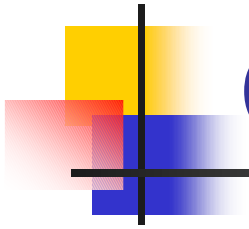
Note that our Square2 class

- Has no move method or setColor method
  - These calls are automatically forwarded to the parent class.
  - We say that the Square2 class *inherits* the methods of the base Rectangl class.



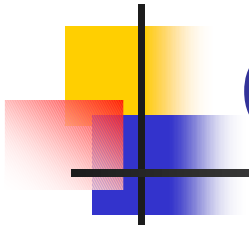
Data inside the class

- Each instance of Rectangl and Square
  - Has x and y position
  - Has a color
  - Has a size
- Each instance could have different values for each these.
- One class,
  - but each instance with separate values



This is a fundamental idea in OO programming

- Classes contain both code and data
- Every instance of a class has its own data.



Overriding methods

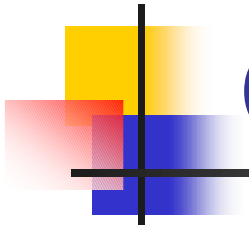
- Rectangl

```
public void draw(Graphics g) {
  g.setColor(color);
  g.drawRect(xpos, ypos,
       width, height);
}
```

- This is called Polymorphism

- Square2

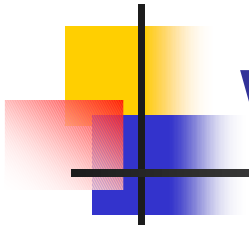
```
public void draw(Graphics g) {
  //overrides Rectangle Draw
  super.draw(g);
  move(xpos+1, ypos+1);
  super.draw(g);
  move(xpos-1, ypos-1);
}
```



Object Oriented Terminology

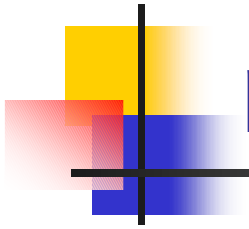
- Inheritance
  - Classes can use methods of parent classes
- Encapsulation
  - Data are held inside each instance of class
- Polymorphism
  - Same method name in derived class may be implemented differently.

This is how our XFrame class works

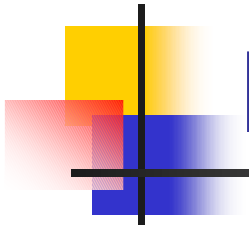


- It inherits all the methods of the Frame class
- We add in a method to catch the Window close box click.
- So while Frame is a very complicated class.
  - XFrame is only 1 lines long.
  - Allows you to reuse your work very efficiently.

Public, private and protected methods



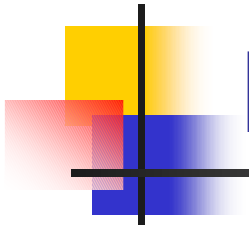
- Public methods are accessible outside a class
- Private methods can only be called inside the class.
- Protected methods can only be called by subclasses.



Public and private variables

- Almost never make the internal variables in a class public.
- Make them private and use getXxx and setXxx methods to access and change them.

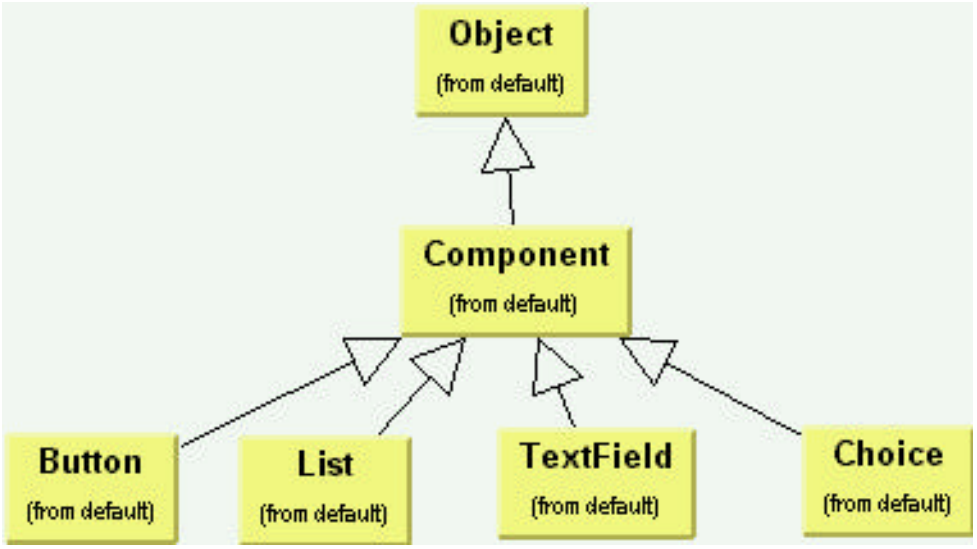
```
public void setColor(Color c) {
  color =  c;
}
```



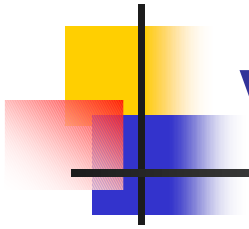
Building Interfaces

- Common window controls
  - TextField
  - TextArea
  - Checkbox
    - also used for Radio buttons
  - Label
- More controls
  - Button
  - List
  - Choice
  - Menus
  - MenuItems
  - Panel

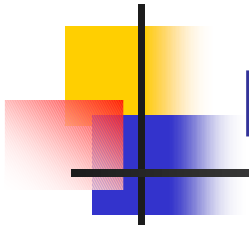
All controls derived from Component



Methods common to most visual Components



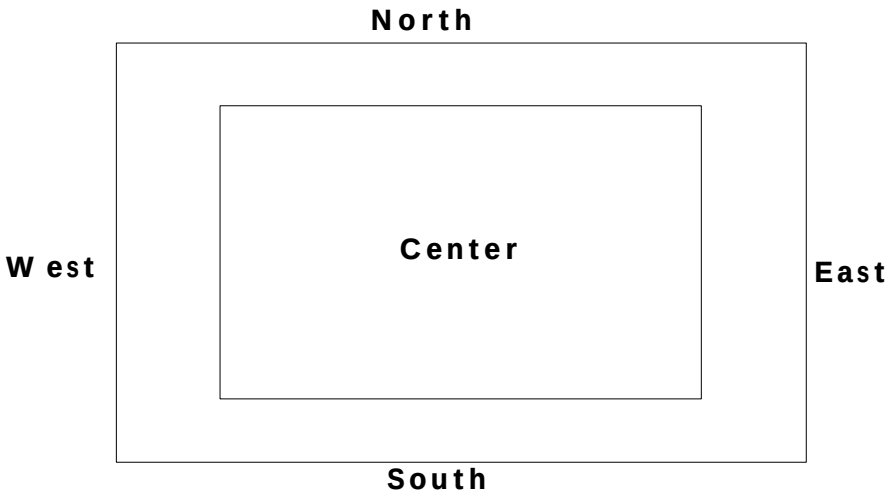
```
paint(Graphics g);
setSize(h, w);
setBounds(x, y, h, w);
setEnable(boolean);
setBackground(Color);
SetVisible(boolean);
repaint();
setFocus();
isEnabled();
getSize();
```



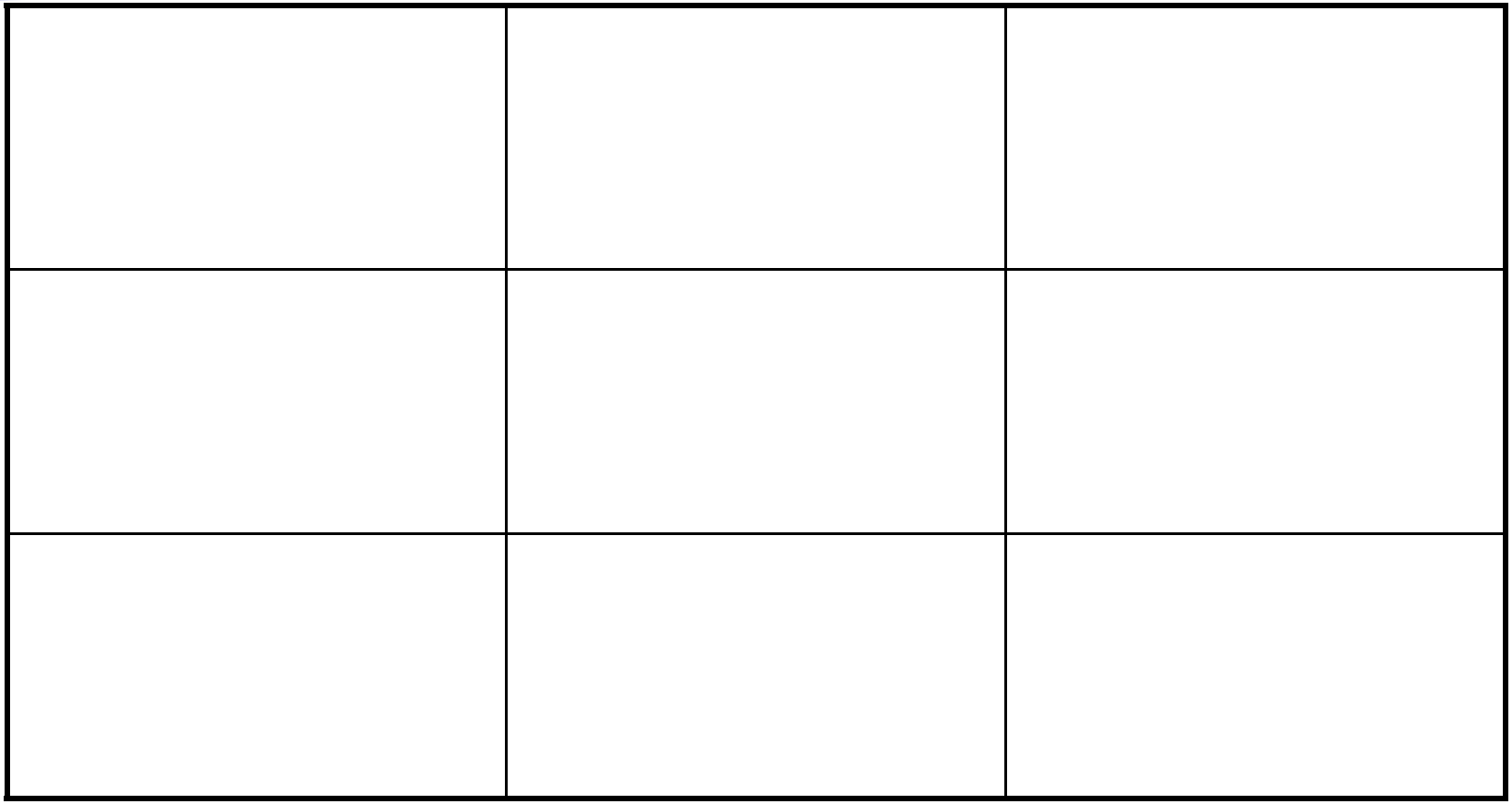
Laying out Components in Window

- Layout Managers
  - Classes that enforce spatial relationships between components
  - Common LayoutManagers
    - BorderLayout
    - FlowLayout
    - GridLayout
    - GridBagLayout
      - others in Swing classes

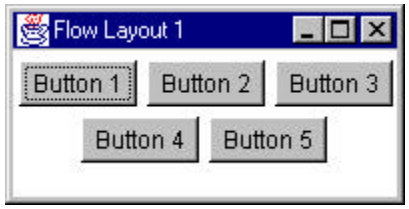
BorderLayout

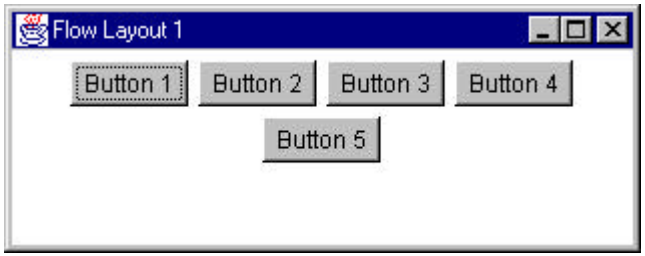


GridLayout

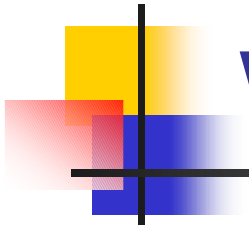


FlowLayout





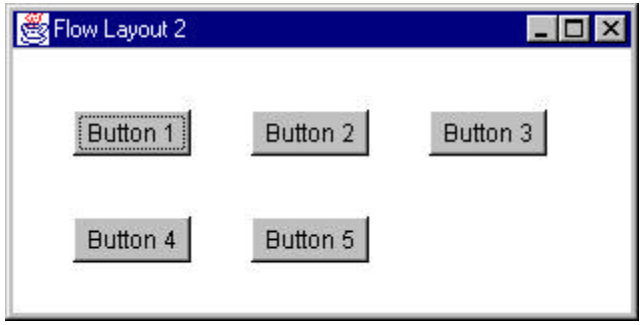
- Components arrange themselves in available space



Writing a Simple FlowLayout

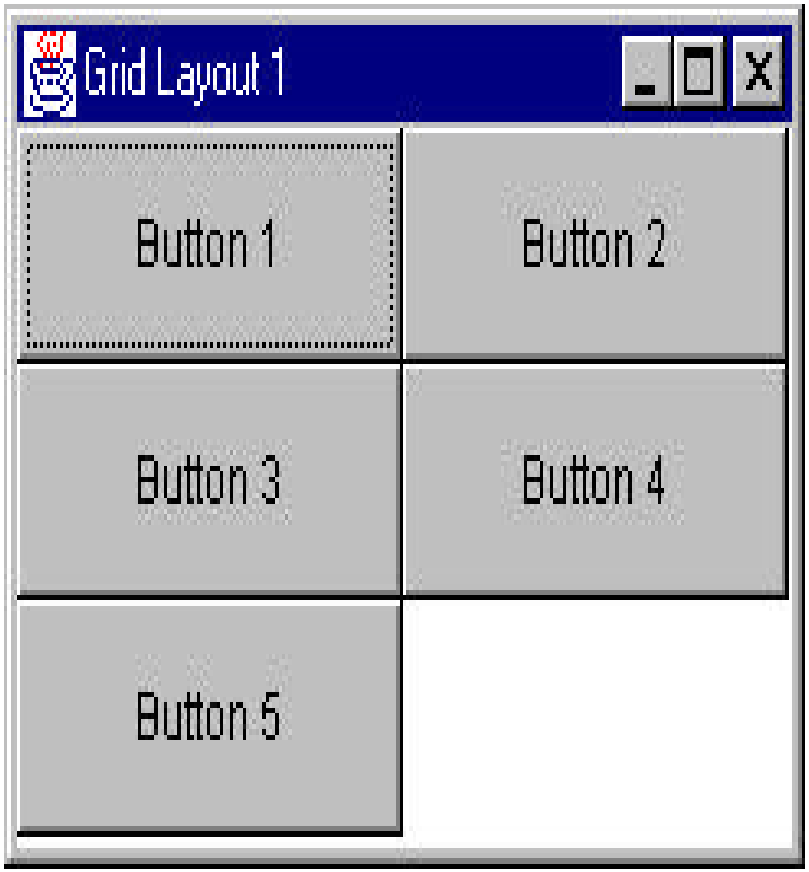
```
public class flow1 extends XFrame {
    public flow1() {
        super("Flow Layout 1");
        setLayout(new FlowLayout());

        add(new Button("Button 1"));
        add(new Button("Button 2"));
        add(new Button("Button 3"));
        add(new Button("Button 4"));
        add(new Button("Button 5"));

        setSize(200,100);
        setVisible(true);
    }
    public static void main(String arg[]) {
        new flow1();
    }
}
```

Also can specify spacing




```
setLayout(new FlowLayout(FlowLayout.LEFT, 30, 30));
```

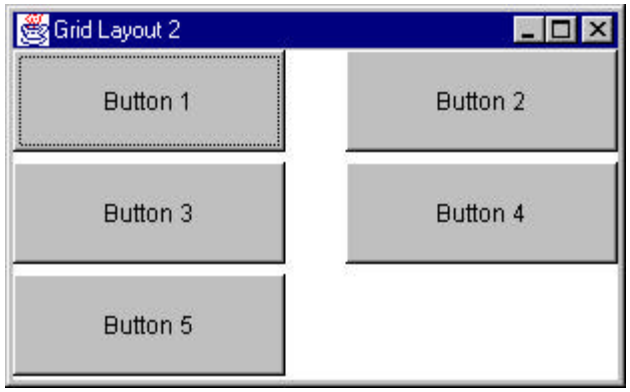
GridLayout




```
super("Grid Layout 1");

setLayout(new GridLayout(3,3));
add(new Button("Button 1"));
add(new Button("Button 2"));
add(new Button("Button 3"));
add(new Button("Button 4"));
add(new Button("Button 5"));
```

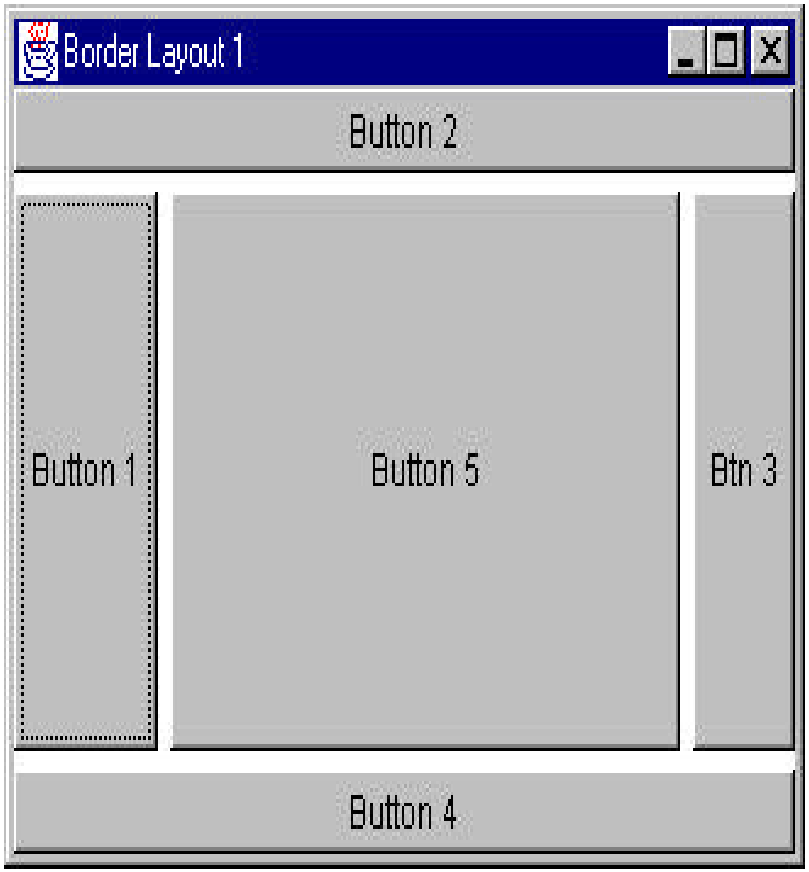
GridLayout with Spacing

```
setLayout(new GridLayout(3, 3, 30, 5));
```



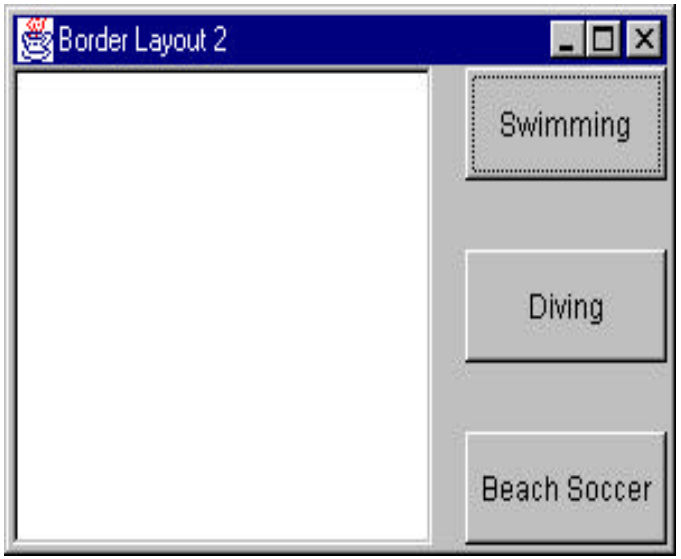
BorderLayout

```
public class border1 extends XFrame {
  public border1() {
    super("Border Layout 1");
    setLayout(new BorderLayout(5, 5));

  add("West",  new Button("Button 1"));
  add("North", new Button("Button 2"));
  add("East",  new Button("Btn 3"));
  add("South", new Button("Button 4"));
  add("Center",new Button("Button 5"));
  setBounds(100,100,300,200);
  setVisible(true);
 }
```



Combine Border and Grid

```
public class border2 extends XFrame {
    public border2() {
        super("Border Layout 2");
        setLayout(new BorderLayout(15, 1));
        setBackground(Color.lightGray );
    //create right panel
        Panel rpanel = new Panel();
        add("East",  rpanel);
    //put List in center
        add("Center", new List(10, false));
    //put rest in right panel
       rpanel.setLayout(new GridLayout(3, 1, 30,25));
       rpanel.add(new Button("Swimming"));
       rpanel.add(new Button("Diving"));
       rpanel.add(new Button("Beach Soccer"));

       setBounds(100,100,300,200);
       setVisible(true);
    }
```

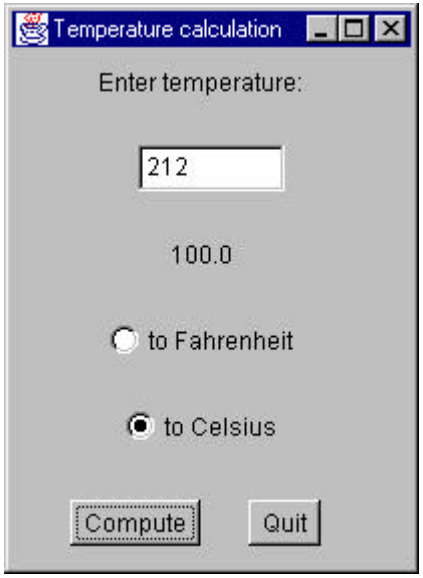


Use Empty Grid Cells for Spacing

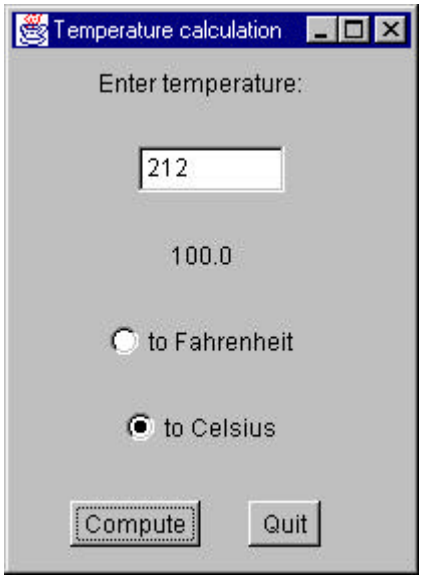
```
add("Center", new List(10, false));
rpanel.setLayout(new GridLayout(5, 1, 30, 5));
rpanel.add(new Panel());
rpanel.add(new Button("Swimming"));
rpanel.add(new Button("Diving"));
rpanel.add(new Button("Beach Soccer"));
rpanel.add(new Panel());
```

Let’s build a useful program

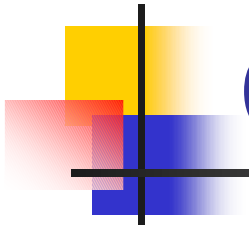
- To convert between temperature scales



We can build this from a grid

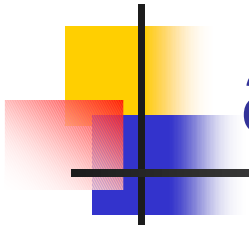


Enter temperature
TextEntry
Label
Checkbox(Radio)
Checkbox(Radio)
Two buttons



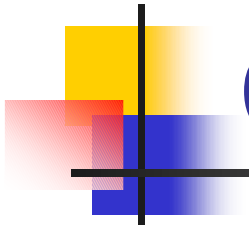
Checkboxes and Radio Buttons

- Checkboxes
  - are square
  - any number can be selected at once
- Radio buttons
  - are round
  - only one of a group can be selected
    - others turned off when one selected



In the Java AWT both kinds are created using checkboxes

- A Checkbox appears as round and singly selectable if
  - it is a member of a CheckboxGroup
- Otherwise it is square and multiply selectable.



Creating Radio Buttons

```
//create the group
CheckboxGroup grpTemper = new CheckboxGroup();
     fahr = new Checkbox("to Fahrenheit", grpTemper, false);
     add(fahr);  //add to layout

     celsius = new Checkbox("to Celsius", grpTemper, false);
     add(celsius); //add to layout
```

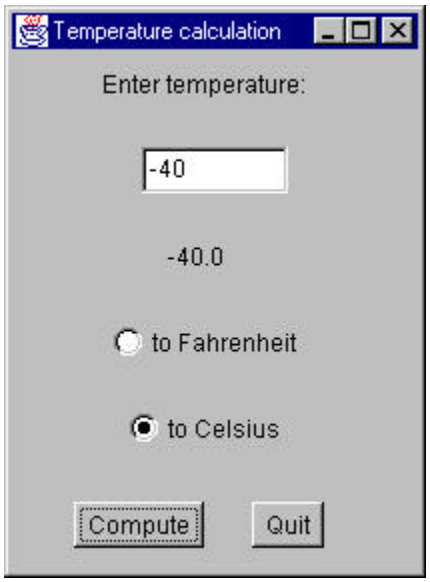
The entire layout

```
setLayout(new GridLayout(6,1));
setBackground(Color.lightGray);

 label1 = new Label("Enter
   temperature:");
 add(label1);

 tempEntry=new TextField(7);
 add(tempEntry);

 result=new Label("            ");
 add(result);

 grpTemper = new CheckboxGroup();
 fahr = new Checkbox("to Fahrenheit",
                     grpTemper, false);
 add(fahr);
```

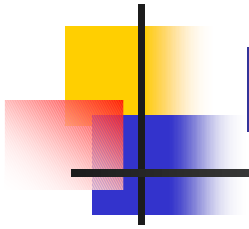
```
celsius = new Checkbox("to Celsius",
                       grpTemper, false);
  add(celsius);

Panel p = new Panel();
add(p);
p.setLayout(new GridLayout(1,2));

compute=new Button("Compute");
Panel lp = new Panel();
p.add(lp);
lp.add(compute);
Panel rp = new Panel();
p.add(rp);
quit = new Button("Quit");
rp.add(quit);
```

Now we have the window

- How do we program it?
- Nearly all components
- receive events from the user
  - Clicks or actions
  - List selections

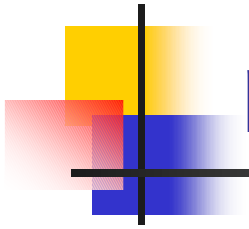


We program our window as a listener

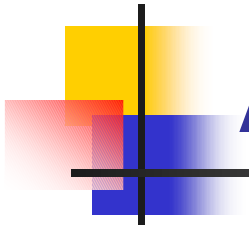


- Kinds of listeners
  - ActionListener
  - ItemListener
  - MouseListener
  - MouseMotionListener
  - WindowListener

Each Listener has one or more methods



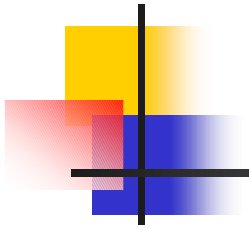
- That are called in response to user actions.
- But since our window extends the Frame class,
  - and since Java only allows single inheritance
  - how can it also extend a Listener class?



An Interface

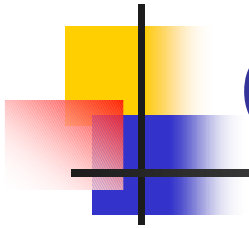
- Is a promise that a class will have certain methods.
- An ActionListener interface must implement the method

```
public void actionPerformed(ActionEvent evt) {
```



If we say that a class

- implements the ActionListener interface
- It *must* have an actionPerfomed method.
- The actionPerformed method is called on
  - Button clicks
  - Enter pressed in a text box
  - Menu items selected

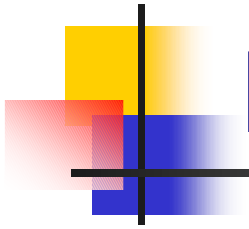


Our TempCalc class

```
import java.awt.*;
import java.awt.event.*;
public class Tempcalc extends Applet implements ActionListener {
//-----------------------------------
public void actionPerformed(ActionEvent evt) {
        Object obj = evt.getSource ();
        if (obj == compute) {
            clickedCompute();   //compute temperature
        }
        else
          if (obj == quit) {
            System.exit(0);     //exit from application
        }
    }
```

You also have to tell each control to listen for actions

```
compute=new Button("Compute");
 Panel lp = new Panel();
 p.add(lp);
 lp.add(compute);
 compute.addActionListener(this);

 Panel rp = new Panel();
 p.add(rp);
 quit = new Button("Quit");
 rp.add(quit);
 quit.addActionListener(this);
```



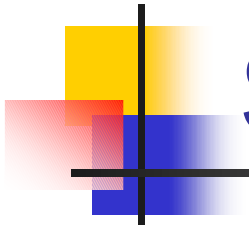
How we compute and display

```
public void clickedCompute() {
        float temp, newtemp;

        temp = new Float(tempEntry.getText()).floatValue();
        if (fahr.getState())
            newtemp = 9 * temp / 5 + 32;
        else
            newtemp = 5 * (temp - 32) / 9;
        result.setText(new Float(newtemp).toString());
    }
```

How our XFrame works

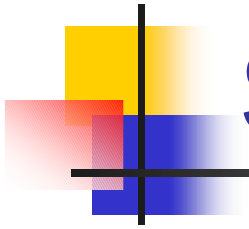
- Xframe is derived from Frame but also implements a WindowListener

```
public class XFrame extends Frame implements WindowListener {
 //  a subclass of Frame that exits on the close box
    public XFrame(String title) {
        super(title);       // put the title in the title bar
        addWindowListener(this);    //listen for close
        setLayout(new BorderLayout());
  }
  //WindowListener interface classes
  public void windowActivated(WindowEvent e){}
  public void windowClosed(WindowEvent e){}
  public void windowDeactivated(WindowEvent e){}
  public void windowIconified(WindowEvent e){}
  public void windowDeiconified(WindowEvent e){}
  public void windowOpened(WindowEvent e){}
  public void windowClosing(WindowEvent e){
      System.exit(0);
      }
}
```



Summary

- Inheritance
  - public and private methods
- Visual components
- Layout managers
- Interfaces
- Event listeners
- Using Xframe



Suggested assignment

- Write a program to add two numbers
  - typed into two text fields
  - add when Add button is clicked