JFrame Class Diagram

javax.swing.JFrame

- int *x*
- int *y*
- int *width*
- int *height*
- Color *backColor*

«constructor»
+ **JFrame**(String)

«update»
+ void **add**(java.awt.Component, int)
+ void **remove**(java.awt.Component)
+ void **repaint**( )
+ void **setBackground**(java.awt.Color)
+ void **setBounds**(int, int, int, int)
+ void **setLayout**(java.awt.LayoutManager)
+ void **setVisible**(boolean)
. . .

Class Specifications

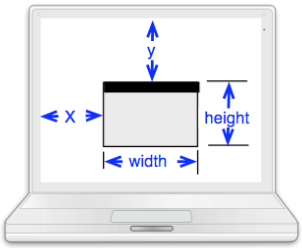
JFrame Class Specifications

Invariant

A `JFrame` object …

- is a rectangular window placed upon a computer display.
- is positioned so that its upper left corner is *x* pixels from the left and *y* pixels from the top of the display
- has a visible region that is ***width*** pixels wide, ***height*** pixels high, and with a background color of ***backColor***.

(Below is a JFrame with ***backColor*** `of` gray on a white display.)



JFrame Class Specifications (continued)

Constructor Methods

`public JFrame(String s)`

post: a new `JFrame` (window) object is created
and *s* is displayed in the window's title bar

note: this method call needs to be followed by calls to `setBounds` and `setVisible`

Update Methods

`public void add(java.awt.Component pic, int j)`

pre: *j* == 0 for best results

post: image *pic* will be drawn upon this JFrame

`public void remove(java.awt.Component pic)`

post: image *pic* will be removed from this JFrame (assuming it was previously added)

`public void repaint()`

post: this JFrame is marked to be redrawn as soon as possible

`public void setBounds(int newX, int newY, int w, int h)`

pre: *w* >= 0 ***and*** *h* >= 0

post: *x* == *newX* ***and*** *y* == *newY*
and *width* == *w* ***and*** *height* == *h*

JFrame Class Specifications (continued)

Update Methods

`public void setBackground(java.awt.Color c)`

post: *backColor* == *c*

`public void setLayout(java.awt.LayoutManager m)`

pre: *m* == `null` (for our purposes)

post: added objects are rearranged via *m*

`public void setVisible(boolean b)`

post: *b* == *true* ***implies*** this JFrame is made visible and brought to the front

An Example

```
import java.awt.Color;
import javax.swing.JFrame;

public class Driver {
  private JFrame blackWin, greenWin;

  public Driver() {
    blackWin = new JFrame("Mine");
    blackWin.setBounds(10, 10, 100, 200);
    blackWin.setLayout(null);
    blackWin.setBackground(Color.black);
    blackWin.setVisible(true);


    greenWin = new JFrame("Yours");
    greenWin.setBounds(150, 100, 100, 50);
    greenWin.setLayout(null);
    greenWin.setVisible(true);
    greenWin.setBackground(Color.green);
  }
}
```

`String` constants require double quotes.

Code Pattern for `JFrame` initialization

1) instantiate: `new JFrame`
2) set dimensions: `setBounds`
3) `setLayout(null)`
4) `setVisible(true)`

the add method

method specification

```
public void add(java.awt.Component pic, int j)
```

pre: $j == 0$ for best results

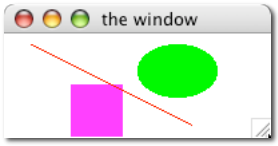
post: image *pic* will be drawn upon this JFrame

The options for Component arguments include...

`java.awt.Label`

author-written { `Line`, `Oval`, `Rectangle` }

and many others



java.awt.Label

java.awt.Label

- int *x*
- int *y*
- int *width*
- int *height*
- Color *backColor*
- Color *foreColor*

«constructor»
+ **Label**(String)
«update»
+ void **repaint**( )
+ void **setBackground**(java.awt.Color)
+ void **setForeground**(java.awt.Color)
+ void **setBounds**(int, int, int, int)
. . .

Code Pattern for `Label` use

1) instantiate: `new Label`
2) set x, y & dimensions: `setBounds`
3) `add` to some canvas
4) `repaint( )`

Example

imports both Color & Label

```
import java.awt.*;
import javax.swing.JFrame;
public class Driver {
  private JFrame win;
  private Label wiLabel;

  public Driver() {
    win = new JFrame("the window");
    win.setBounds(10, 10, 200, 200);
    win.setLayout(null);
    win.setVisible(true);

    wiLabel = new Label("Wisconsin");
    wiLabel.setBounds(10, 40, 120, 20);
    wiLabel.setForeground(Color.blue);
    wiLabel.repaint();
    win.add(wiLabel, 0);
  }
}
```


Rectangle

Rectangle

- int *x*
- int *y*
- int *width*
- int *height*
- Color *backColor*

«constructor»
+ **Rectangle**(int, int, int, int)
«update»
+ void **add**(java.awt.Component, int)
+ void **repaint**( )
+ void **setBackground**(java.awt.Color)
+ void **setLocation**(int, int)
+ void **setSize**(int, int)
. . .

Code Pattern for `Rectangle` use

1) instantiate: `new Rectangle`
2) `setBackground`
3) `add` to some canvas
4) `repaint( )`

Example

```
import java.awt.Color;
import javax.swing.JFrame;
public class Driver {
  private JFrame win;
  private Rectangle box;

  public Driver() {
    win = new JFrame("the window");
    win.setBounds(10, 10, 200, 200);
    win.setLayout(null);
    win.setVisible(true);

    box = new Rectangle(50, 50, 10, 10);
    box.setBackground(Color.magenta);
    box.repaint();
    win.add(box, 0);
  }
}
```

Note: no import required for Rectangle, but Rectangle.class file must be in same folder as Driver.class.

Oval

Oval

- int *x*
- int *y*
- int *width*
- int *height*
- Color *backColor*

«constructor»
+ **Oval**(int, int, int, int)

«update»
+ void **add**(java.awt.Component, int)
+ void **repaint**( )
+ void **setBackground**(java.awt.Color)
+ void **setLocation**(int, int)
+ void **setSize**(int, int)
. . .

The only differences between Oval and Rectangle is the name of the constructor and their appearance.

Code Pattern for `Oval` use

1) instantiate: `new Oval`
2) `setBackground`
3) `add` to some canvas
4) `repaint( )`

Example

```
import java.awt.Color;
import javax.swing.JFrame;
public class Driver {
  private JFrame win;
  private Oval dot;

  public Driver() {
    win = new JFrame("the window");
    win.setBounds(10, 10, 200, 200);
    win.setLayout(null);
    win.setVisible(true);

    dot = new Oval(100, 50, 60, 30);
    dot.setBackground(Color.green);
    dot.repaint();
    win.add(dot, 0);
  }
}
```


the add method

A graphical object (excepting `JFrame`) only becomes visible if it is `add`ed to some other visible object (call the underlying object the *container*).

Drawing Rules for *container*`.add(` *object*`, 0 );`

1) A *container* might be a `JFrame`, `Oval`, `Rectangle`, (others).

2) Each *object* can be added to no more than one *container*.

3) Last `add` to the same *container* appears in front.

4) An `add`ed *object* is **clipped** to the bounding border of its *container*.

3.1.10

the repaint method

A call to `repaint()` is needed (sometimes) to cause an object to become visible.

Example

```
...
square = new Rectangle(30, 40, 50, 50);
window.add(square, 0);
square.repaint();
circle = new Oval(0, 0, 20, 20);
square.add(circle, 0);
circle.repaint();
```

or

```
...
square = new Rectangle(30, 40, 50, 50);
window.add(square, 0);
circle = new Oval(0, 0, 20, 20);
square.add(circle, 0);
window.repaint();
```


Example

```
import java.awt.Color;
import javax.swing.JFrame;
public class Driver  {
   private JFrame win, bottomWindow;
   private Rectangle  grayRect;
   private Oval redOval, whiteDot;

   public Driver()  {
      win = new JFrame("the window");
      win.setBounds(10, 10, 200, 100);
      win.setLayout(null);
      win.setBackground( Color.lightGray );
      win.setVisible(true);

      grayRect = new Rectangle(40, 40, 40, 50);
      grayRect.setBackground( Color.darkGray );
      win.add(grayRect, 0);
      whiteDot = new Oval(10, 10, 80, 80);
      whiteDot.setBackground( Color.white );
      grayRect.add(whiteDot, 0);

      redOval = new Oval(60, -20, 100, 50);
      redOval.setBackground( Color.red );
      win.add(redOval, 0);
      win.repaint();
   }
}
```