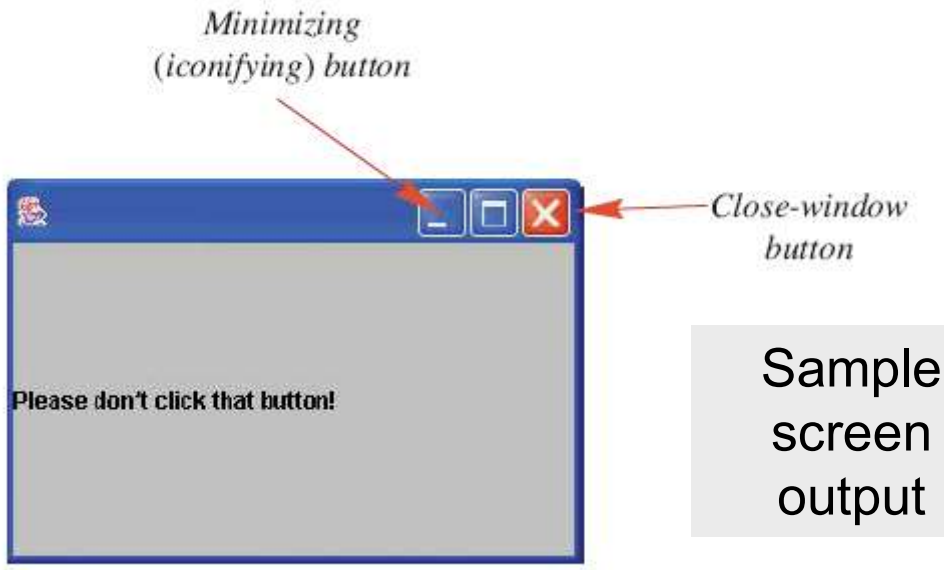
13-2A BASIC SWING DETAILS

- listing 13.1 **`class FirstSwingDemo`**
- Window appears on screen with message

- Uses **`JFrame`** object



Sample screen output

Listing 13.1 `FirstSwingDemo.java`

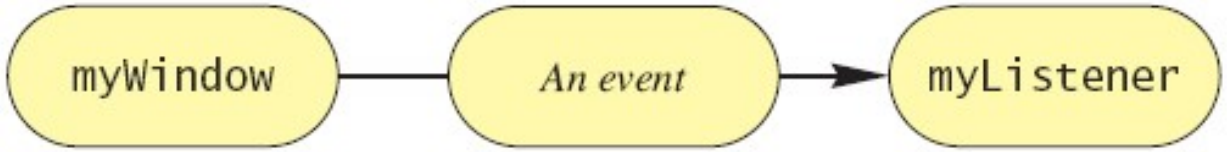
```
import javax.swing.JFrame;
import javax.swing.JLabel;

/**
 A simple demonstration of a window constructed using Swing.
*/
public class FirstSwingDemo
{
   public static final int WIDTH = 300;
   public static final int HEIGHT = 200;

   public static void main(String[] args)
   {
      JFrame myWindow = new JFrame( );
      myWindow.setSize(WIDTH, HEIGHT);
      JLabel myLabel =
            new JLabel("Please don't click that button!");
      myWindow.getContentPane( ).add(myLabel);

      WindowDestroyer myListener = new WindowDestroyer( );
      myWindow.addWindowListener(myListener);

      myWindow.setVisible(true);
   }
}
```

Programming Example

- Note use of
  - » `JLabel` object
  - » Content pane, the inside of the window
  - » Registering a listener
  - » Setting the window to be visible

Programming Example

- Figure 13.2 Clicking the Close-Window button



- listing 13.2 `class WindowDestroyer`

Listing 13.2 `class WindowDestroyer.java`

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

/**
 If you register an object of this class as a listener to any
 object of the class JFrame, the object will end the program
 and close the JFrame, if the user clicks the JFrame's
 close-window button.
*/
public class WindowDestroyer extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}
```

More About Window Listeners

- Derived from class **`WindowAdapter`**
- Figure 13.3 Methods in the Class **`WindowAdapter`**

`public void windowOpened(WindowEvent e)` Invoked when a window has been opened.
`public void windowClosing(WindowEvent e)` Invoked when a window is in the process of being closed. Clicking the close-window button causes an invocation of this method.
`public void windowClosed(WindowEvent e)` Invoked when a window has been closed.
`public void windowIconified(WindowEvent e)` Invoked when a window is iconified. When you click the minimize button in a `JFrame` object, the window is iconified. See Listing 13.1 for the location of the minimize (iconifying) button.

More About Window Listeners

- Figure 13.3 ctd.

`public void windowDeiconified(WindowEvent e)` Invoked when a window is deiconified. When you activate a minimized window, it is deiconified.
`public void windowActivated(WindowEvent e)` Invoked when a window is activated. When you click in a window, it becomes the activated window. Other actions can also activate a window.
`public void windowDeactivated(WindowEvent e)` Invoked when a window is deactivated. When any window is activated, all other windows are deactivated. Other actions can also deactivate a window.
`public void windowGainedFocus(WindowEvent e)` Invoked when a window gains focus. (Focus is not discussed in this text.)
`public void windowLostFocus(WindowEvent e)` Invoked when a window loses focus. (Focus is not discussed in this text.)
`public void windowStateChanged(WindowEvent e)` Invoked when a window changes state.

Size Units for Screen Objects

- Smallest screen area displayed is a pixel
- With Swing,
  - » Both size and position of objects on screen measured in pixels.
- A screen's resolution is a measure of the number of pixels it can display

The `setVisible` Method

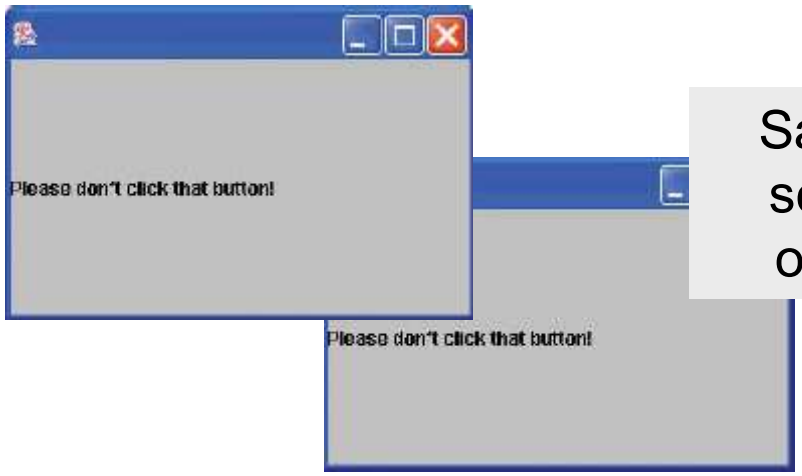
- Takes one argument of type **`boolean`**
- Other objects besides windows can be made visible or invisible
  - » The object calls the method

Syntax:

```
Object_For_Screen.setVisible(Boolean_Expression);
```

Programming Example

- A better version of first Swing program,
  listing 13.3 **`class FirstWindow`**
- A program that uses class **`FirstWindow`**
  listing 13.4 **`class FirstWindowDemo`**



Sample screen output

Listing 13.3 `class FirstWindow`

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

/**
 A simple window class.
*/
public class FirstWindow extends JFrame
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public FirstWindow( )
    {
        super( );

        setSize(WIDTH, HEIGHT);
        JLabel myLabel = new JLabel("Please don't click that button!");
        getContentPane( ).add(myLabel);

        WindowDestroyer listener = new WindowDestroyer( );
        addWindowListener(listener);
    }
}
```

Listing 13.4 `class FirstWindowDemo`

```
/**
 A simple demonstration of using a window class. To see
 both windows you will probably have to move the top window.
*/
public class FirstWindowDemo
{
    public static void main(String[] args)
    {
        FirstWindow window1 = new FirstWindow( );
        window1.setVisible(true);

        FirstWindow window2 = new FirstWindow( );
        window2.setVisible(true);
    }
}
```

Adding Items to a `JFrame` Window

- Requires the following syntax:

Syntax (within a constructor):

```
getContentPane().add(JLabel_Object);
```

Example (within a constructor):

```
JLabel myLabel = new JLabel("Please don't click that button!");
getContentPane().add(myLabel);
```

Programming Example

- listing 13.5 **`class SecondWindow`**
- Note new elements
  - » A title, Second Window
  - » A local variable named **`contentPane`** to reference the content pane of the window
  - » A background color, blue
  - » A new way to add the window listener

Listing 13.5 SecondWindow

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

public class SecondWindow extends JFrame
{
   public static final int WIDTH = 200;
   public static final int HEIGHT = 200;

   public SecondWindow( )
   {
      super( );

      setSize(WIDTH, HEIGHT);

      Container contentPane = getContentPane( );
      JLabel label = new JLabel("Now available in color!");
      contentPane.add(label);

      setTitle("Second Window");
      contentPane.setBackground(Color.BLUE);

      addWindowListener(new WindowDestroyer( ));
   }
```

```
public SecondWindow(Color customColor)
  {
     super( );

     setSize(WIDTH, HEIGHT);

     Container contentPane = getContentPane( );
     JLabel label = new JLabel("Now available in color!");
     contentPane.add(label);

     setTitle("Second Window");
     contentPane.setBackground(customColor);

     addWindowListener(new WindowDestroyer( ));
  }
}
```

A Window with Color

- Note color constants
- Figure 13.4 , the color constants

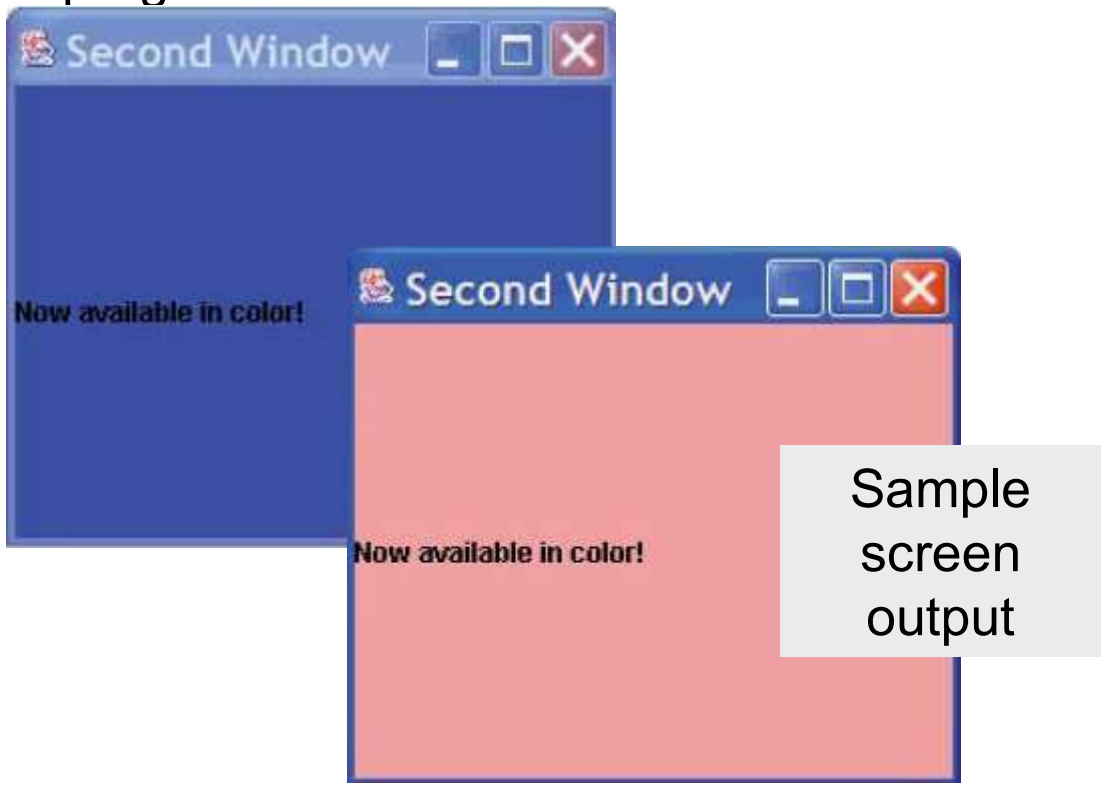
Color.BLACK	Color.MAGENTA
Color.BLUE	Color.ORANGE
Color.CYAN	Color.PINK
Color.DARK_GRAY	Color.RED
Color.GRAY	Color.WHITE
Color.GREEN	Color.YELLOW
Color.LIGHT_GRAY	

- listing 13.6

```
class SecondWindowDemo
```

A Window with Color

- Run of demo program



Sample screen output

Listing 13.6 SecondWindowDemo

```
import java.awt.Color;

public class SecondWindowDemo
{
    /**
     Creates and displays two windows of the class SecondWindow.
    */
    public static void main(String[] args)
    {
        SecondWindow window1 = new SecondWindow( );
        window1.setVisible(true);

        SecondWindow window2 = new SecondWindow(Color.PINK);
        window2.setVisible(true);
    }
}
```

Methods in Class JFrame

- Figure 13.5

`public JFrame()` Creates a new `JFrame` window.
`public JFrame(String title)` Creates a new `JFrame` window with the given title.
`public void add()` This method is inherited from an ancestor class and is basically useless. Adding something to a `JFrame`'s content pane—using `getContentPane().add(`*Item_Added*`)`—involves a different `add` method.
`public void addWindowListener(WindowListener ear)` Registers `ear` as a listener for events fired by the `JFrame` window.
`public Container getContentPane()` Returns the content pane of the `JFrame` window. Note that the content pane returned is of type `Container`.

Methods in Class JFrame

- Figure 13.5 ctd.

`public void setBackground(Color c)` Sets the background color to `c`.
`public void setForeground(Color c)` Sets the foreground color to `c`.
`public void setSize(int width, int height)` Resizes the window to the specified width and height.
`public void setTitle(String title)` Displays the given title on the title bar of the window.
`public void setVisible(boolean isVisible)` Makes the window visible if the argument is `true`, or invisible if the argument is `false`.