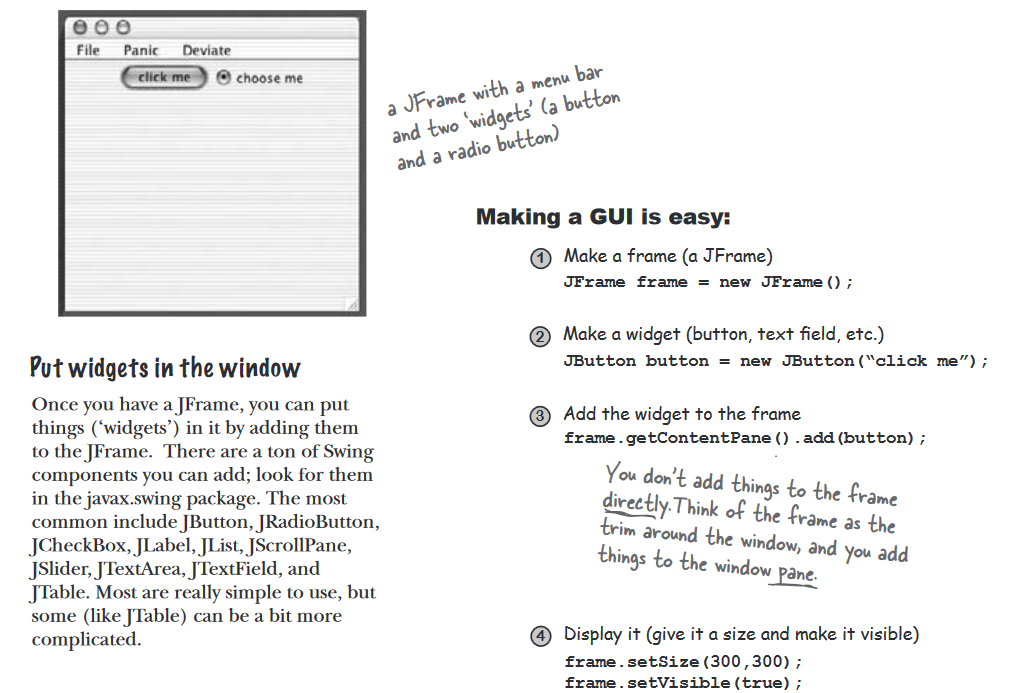
**#** A JFrame is the object that represents a window on the screen. It is where we put all the interface things like boxes, buttons,text fields and so on.

**#**



**#** To use GUI’s, we first need to import the Swing Package using - **import javax.swing.\*;**

**#** First GUI Code-

**import javax.swing.\*;**

**Public class simpleGui1 {**

**public static main void (string[] args) {**

**JFrame frame = new JFrame();**

**Jbutton button = new Jbutton(“click me”);**

**frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**frame.getControlPane().add(button);**

**frame.setsize(300,300);**

**frame.setVisible();**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**#** To get our buttons to do something specific, we need to do two things -

1)A **method** to be called when the user clicks (the thing you want to happen as a result of the button click).

2) A **way** to know when to trigger that method. In other words, a way to know when the user clicks the button!

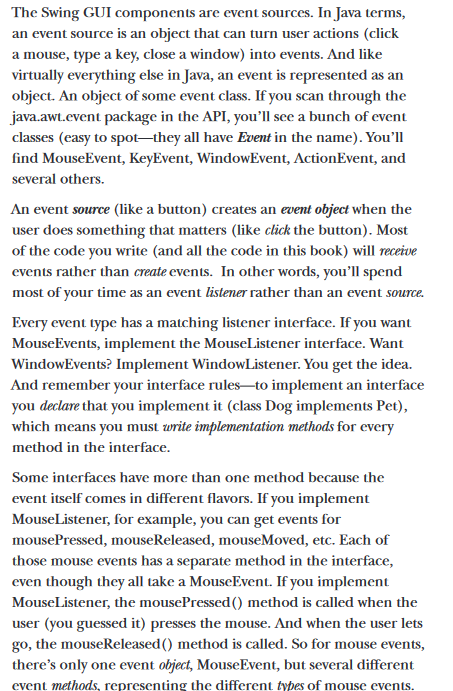
**#** The standard Java Look and Feel is known as **Metal.** While the Mac OS screens use **OS X Aqua**.

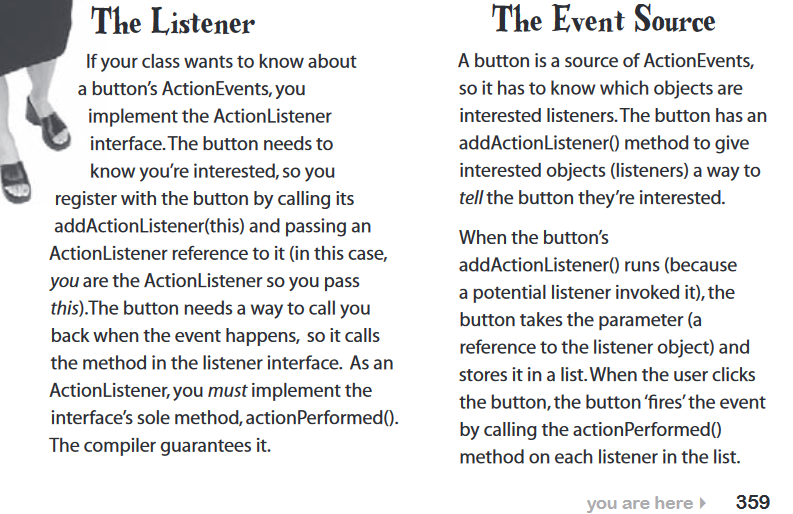
**#** In Java, the process of getting and handling a user-event is known as **Event-Handling**.

For example- when a user presses a button which read “Click me” and now you want to handle it by changing the text to “I have been clicked.

**#** IF you care about the button’s events, **implement an interface** that says, “I’m **listening** for your events .A listener interface is the bridge between the listener (you) and event source (the button).

**#** When you implement a listener interface, you give the button a way to call you back. The interface is where the call-back method is declared.



****

**#** To get a button’s ActionEvent -

1. implement the ActionListener interface
2. Register with the button (tell it you want to listen for events)
3. Define the event-handling method (implement the actionPerformed() method from the ActionListener interface)

**CODE DEMONSTRATION IN NEXT PAGE**



**# Event objects hold DATA about the event, and pass it to the LISTENER.**

**# There are three ways to put things on our GUI:**

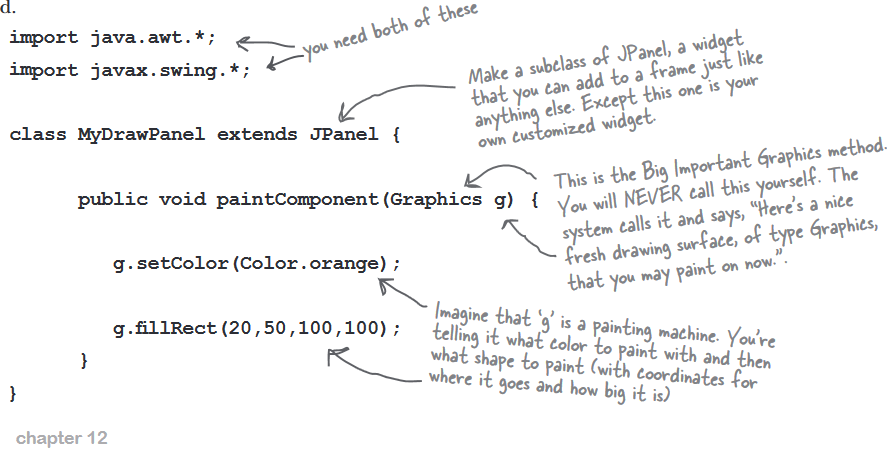
1. **Put Widgets in the frame. (Using previous example, button)**
2. **Draw 2D graphics on a widgetUse a graphics object to paint shapes.**

graphics.fillOval(70,70,100,100);

1. **Put JPEG on a widget.**

graphics.drawImage(myPic,10,10,this);

# To add our own drawing or graphic on our screen, our best bet is to create a panel and draw on it, which is handed by the system ( we never call it)



# In the paintComponent function, we can use it to view our own image by using - Image image = new ImageIcon(“catzilla.jpg”).getImage(); g.drawImage(image,**3,4,**this);

Note- The 3 and 4 parameters the x and y coordinates where the image should be displayed.

**#** The object referenced by the Graphics parameter to paintComponent() is actually an instance of the Graphics2D class. The Graphics 2D class has a variety of methods including: fill3DRect(), draw3DRect(), rotate(), scale(), shear(), transform().

**#** To invoke the Graphics2D methods, you must cast the parameter from a Graphics object to a Graphics2D object:

Graphics2D g2d = (Graphics2D) g; .