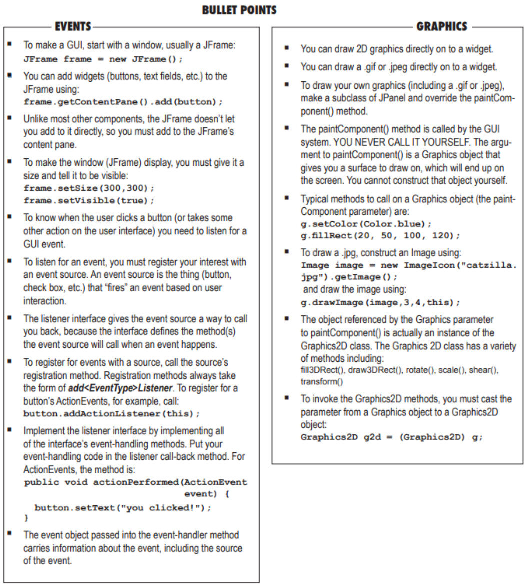
CHPATER -14

A Very Graphic Story



BULLET POINTS

EVENTS

- To make a GUI, start with a window, usually a JFrame:
  `JFrame frame = new JFrame();`
- You can add widgets (buttons, text fields, etc.) to the JFrame using:
  `frame.getContentPane().add(button);`
- Unlike most other components, the JFrame doesn't let you add to it directly, so you must add to the JFrame's content pane.
- To make the window (JFrame) display, you must give it a size and tell it to be visible:

```
frame.setSize(300,300);
frame.setVisible(true);
```

- To know when the user clicks a button (or takes some other action on the user interface) you need to listen for a GUI event.
- To listen for an event, you must register your interest with an event source. An event source is the thing (button, check box, etc.) that "fires" an event based on user interaction.
- The listener interface gives the event source a way to call you back, because the interface defines the method(s) the event source will call when an event happens.
- To register for events with a source, call the source's registration method. Registration methods always take the form of ***add<EventType>Listener***. To register for a button's ActionEvents, for example, call:
  `button.addActionListener(this);`
- Implement the listener interface by implementing all of the interface's event-handling methods. Put your event-handling code in the listener call-back method. For ActionEvents, the method is:

```
public void actionPerformed(ActionEvent
                                 event) {
  button.setText("you clicked!");
}
```

- The event object passed into the event-handler method carries information about the event, including the source of the event.

GRAPHICS

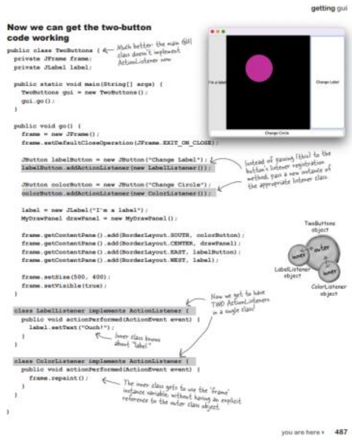
- You can draw 2D graphics directly on to a widget.
- You can draw a .gif or .jpeg directly on to a widget.
- To draw your own graphics (including a .gif or .jpeg), make a subclass of JPanel and override the paintComponent() method.
- The paintComponent() method is called by the GUI system. YOU NEVER CALL IT YOURSELF. The argument to paintComponent() is a Graphics object that gives you a surface to draw on, which will end up on the screen. You cannot construct that object yourself.
- Typical methods to call on a Graphics object (the paintComponent parameter) are:

```
g.setColor(Color.blue);
g.fillRect(20, 50, 100, 120);
```

- To draw a .jpg, construct an Image using:
  `Image image = new ImageIcon("catzilla.jpg").getImage();`
  and draw the image using:
  `g.drawImage(image,3,4,this);`
- 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;`

- **Check Box**:
- itemStateChanged()
- actionPerformed()
- focusGained()
- **Text Field**:
- actionPerformed()
- keyTyped()
- focusGained()
- **Scrolling List**:
- itemStateChanged()
- focusGained()
- **Button**:
- actionPerformed()
- focusGained()
- mousePressed()
- **Dialog Box**:
- windowClosing()
- focusGained()

- **Radio Button**:
- itemStateChanged()
- actionPerformed()
- focusGained()
- **Menu Item**:
- actionPerformed()
- focusGained()



getting gui

Now we can get the two-button code working

```
public class TwoButtons {
  private JFrame frame;
  private JLabel label;

  public static void main(String[] args) {
    TwoButtons gui = new TwoButtons();
    gui.go();
  }

  public void go() {
    frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JButton labelButton = new JButton("Change Label");
    labelButton.addActionListener(new LabelListener());

    JButton colorButton = new JButton("Change Circle");
    colorButton.addActionListener(new ColorListener());

    label = new JLabel("I'm a label");
    MyDrawPanel drawPanel = new MyDrawPanel();

    frame.getContentPane().add(BorderLayout.SOUTH, colorButton);
    frame.getContentPane().add(BorderLayout.CENTER, drawPanel);
    frame.getContentPane().add(BorderLayout.EAST, labelButton);
    frame.getContentPane().add(BorderLayout.WEST, label);

    frame.setSize(500, 400);
    frame.setVisible(true);
  }

  class LabelListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      label.setText("Ouch!");
    }
  }

  class ColorListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      frame.repaint();
    }
  }
}
```

Much better: the main GUI class doesn't implement ActionListener now

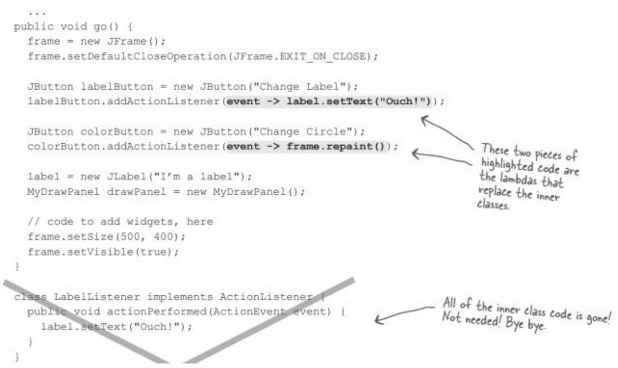
Instead of passing (this) to the button's listener registration method, pass a new instance of the appropriate listener class.

Now we get to have TWO ActionListeners in a single class!

Inner class knows about "label"

The inner class gets to use the 'frame' instance variable, without having an explicit reference to the outer class object.

you are here ▸ 487



```
  ...
public void go() {
  frame = new JFrame();
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JButton labelButton = new JButton("Change Label");
  labelButton.addActionListener(event -> label.setText("Ouch!"));

  JButton colorButton = new JButton("Change Circle");
  colorButton.addActionListener(event -> frame.repaint());

  label = new JLabel("I'm a label");
  MyDrawPanel drawPanel = new MyDrawPanel();

  // code to add widgets, here
  frame.setSize(500, 400);
  frame.setVisible(true);
}

class LabelListener implements ActionListener {
  public void actionPerformed(ActionEvent event) {
    label.setText("Ouch!");
  }
}
```

These two pieces of highlighted code are the lambdas that replace the inner classes.

All of the inner class code is gone! Not needed! Bye bye.

EXERCISE

1.WHO AM I?

I got the whole GUI, in my hands. - JFrame click it.
Every event type has one of these. - Listener interface
The listener's key method.-actionPerformed()
This method gives JFrame its size.-setSize()
You add code to this method but never call it.-paintComponent()
When the user actually does something, it's an ____ .-event
Most of these are event sources.-swing components
I carry data back to the listener.-event object
An addXxxListener( ) method says an object is an ___ .
How a listener signs up.-addXxxListener()
The method where all the graphics code goes.-paintComponent()

I’m typically bound to an instance.-inner class
The “g” in (Graphics g) is really of this class.-Graphics2D
The method that gets paintComponent() rolling.-repaint()
The package where most of the Swingers reside.-javax.swing

2.BE THE COMPILER

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class InnerButton {
   private JButton button;

   public static void main(String[] args) {
      InnerButton gui = new InnerButton();
      gui.go();
   }

   public void go() {
      JFrame frame = new JFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      button = new JButton("A");
      button.addActionListener(new ButtonListener());
      frame.getContentPane().add(BorderLayout.SOUTH, button);
      frame.setSize(200, 100);
      frame.setVisible(true);
   }

   class ButtonListener implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         if (button.getText().equals("A")) {
            button.setText("B");
         } else {
            button.setText("A");
         }
      }
   }
}
```

3.POOL PUZZLE

```
import javax.swing.*;
import java.awt.*;
import java.util.concurrent.TimeUnit;
```

```
}

public class Animate {
   int x = 1;
   int y = 1;

   public static void main(String[] args) {
      Animate gui = new Animate();
      gui.go();
   }

   public void go() {
      JFrame frame = new JFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      MyDrawP drawP = new MyDrawP();
      frame.getContentPane().add(drawP);
      frame.setSize(500, 270);
      frame.setVisible(true);

      for (int i = 0; i < 124; i++, y++, x++) {
         x++;
         drawP.repaint();
         try {
            TimeUnit.MILLISECONDS.sleep(50);
         } catch (Exception ex) {
         }
      }
   }

   class MyDrawP extends JPanel {
      public void paintComponent(Graphics g) {
         g.setColor(Color.white);
         g.fillRect(0, 0, 500, 250);
         g.setColor(Color.blue);
         g.fillRect(x, y, 500 - x * 2, 250 - y * 2);
      }
   }
}
```