Chap 8 - GUIs

- Swing vs AWT
- Objects that make up a GUI are called components.
- This example uses two JLabel components, three JCheckBox components, and a JButton.

1

```
import javax.swing.*;
import java.awt.*;
class SampleAWT {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Sample AWT Components");
    Container display = frame.getContentPane();
    display.setLayout(new FlowLayout());
    display.add(new JLabel(
            "What programming languages do you know?"));
    display.add(new JLabel("Check all that apply:"));
    display.add(new JCheckBox("Java",true));
    display.add(new JCheckBox("C",false));
    display.add(new JCheckBox("C++",false));
    display.add(new JButton("Submit"));
    frame.pack();
    frame.show();
    System.out.println("The end of main().");
  }
}
```

2

The Event Thread

- Notice that although main() ends, the program is still running.
- Creating a JFrame, implicitly creates a separate thread of execution that enters an infinite loop, looking for events.
- This is called an *event driven* program.
- How do we get it to do something when an event occurs?

3

The Event Model

- Swing components are *event sources* - e.g. clicking a JButton creates an ActionEvent.
- To respond to an event you create an *event listener*, and tell the event source about the event listener.
- Event sources have methods like add*Something*Listener(), where *Something* is a particular event type.

4

```
import javax.swing.*;
import java.awt.*;
class SampleAWT2 {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Sample AWT Components");
    Container display = frame.getContentPane();
    // some lines deleted - same as before
    display.add(new JCheckBox("C++",false));
    JButton submit;
    display.add(submit = new JButton("Submit"));
    submit.addActionListener(new Submit());
    frame.pack();
    frame.show();
    System.out.println("The end of main().");
  }
}
```

5

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

class Submit implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("Submitted");
  }
}
```

6

```
import javax.swing.*;
import java.awt.*;
class SampleAWT2 {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Sample AWT Components");
    Container display = frame.getContentPane();
    // some lines deleted - same as before
    display.add(new JCheckBox("C++",false));
    JButton submit;
    display.add(submit = new JButton("Submit"));
    submit.addActionListener(new Submit());
    frame.addWindowListener(new Exiter());
    frame.pack();
    frame.show();
    System.out.println("The end of main().");
  }
}
```

7

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

class Exiter extends WindowAdapter {
  public void windowClosing(WindowEvent e) {
    System.out.println("Exiting.");
    System.exit(0);
  }
}

/*
  This uses a WindowAdapter. We will talk about adapters
  later. We could have implemented this class using
  "implements WindowListener" but it would have required
  a bit more code as will be explained later.
*/
```

8

Listening to a JButton

- Create the button with new `JButton()`
- Get the `Container` for the `JFrame` using `getContentPane()`.
- Add the button to the content pane using `add()`.
- Create an `ActionEventListener` by
  - adding implements `ActionEventListener` to the class declaration and
  - defining an `actionPerformed()` method.
- Add the listener object to the list of listeners for the button by calling `button.addActionListener(listener)`

9

The Event Loop

```
while (true)
  wait for next event;
  determine event type, call it T;
  determine GUI object where event
      occurred, call it O;
  call appropriate method in each T
      listener added to O
end while loop
```

10

```
//TextInput.java
import java.awt.*;
import javax.swing.*;

class TextInput {
  public static void main(String[] args) {
    JFrame frame = new JFrame("TextInput");
    Container pane = frame.getContentPane();
    JTextField input = new
       JTextField("Edit this text then hit <return>");
    Echo listener = new Echo();
    input.addActionListener(listener);
    pane.add(input);
    frame.pack();
    frame.show();
  }
}
```

11

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

class Echo implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    JTextField source = (JTextField)e.getSource();
    String text = source.getText();
    System.out.println(text);
  }
}
```

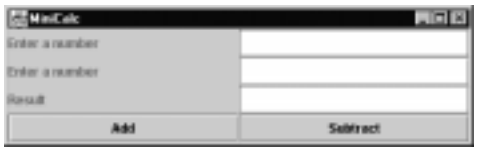
12

Using Several Components

- How do we arrange the GUI components?
  - Using layout managers.
- How do we respond to event from several sources?
  - Create separate listeners, or
  - determine the source of the event with a single listener.

13

A Mini-Calculator



14

```
//MiniCalc.java - demo GridLayout
import java.awt.*;
import javax.swing.*;

class MiniCalc {
  public static void main(String[] args) {
    JFrame frame = new JFrame("MiniCalc");
    Container pane = frame.getContentPane();
    // create the major components
    JTextField firstNumber = new JTextField(20);
    JTextField secondNumber = new JTextField(20);
    JTextField result = new JTextField(20);
    JButton addButton = new JButton("Add");
    JButton subButton = new JButton("Subtract");
```

15

```
    // there will be 4 rows of 2 components each
    pane.setLayout(new GridLayout(4, 2));
    // add all of the components to the content pane
    pane.add(new JLabel("Enter a number"));
    pane.add(firstNumber);
    pane.add(new JLabel("Enter a number"));
    pane.add(secondNumber);
    pane.add(new JLabel("Result"));
    pane.add(result);
    pane.add(addButton);
    pane.add(subButton);
    // setup the listener, listening to the buttons
    DoMath listener =
        new DoMath(firstNumber, secondNumber, result);
    subButton.addActionListener(listener);
    addButton.addActionListener(listener);
    frame.pack();
    frame.show(); }} // ran out of room on the page :(
```

16

```
//DoMath.java - respond to two different buttons
import javax.swing.*;
import java.awt.event.*;

class DoMath implements ActionListener {
  private JTextField inputOne, inputTwo, output;

  DoMath(JTextField first, JTextField second,
         JTextField result)
  {
    inputOne = first;
    inputTwo = second;
    output = result;
  }
```

17

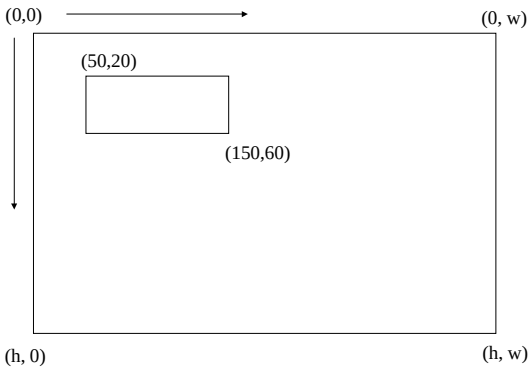
```
  public void actionPerformed(ActionEvent e) {
    double first, second;
    first =
        Double.parseDouble(inputOne.getText().trim());
    second =
        Double.parseDouble(inputTwo.getText().trim());
    if (e.getActionCommand().equals("Add"))
      output.setText(String.valueOf(first + second));
    else
      output.setText(String.valueOf(first - second));
  }
}
```

18

Drawing with Swing

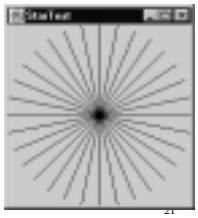
- Drawing is done on an instance of class java.awt.Graphics (or java.awt.Graphics2D).
- Using Graphics, units are always in terms of pixels.
- The origin is in the upper left corner with positive directions being down and to the right.

19



20

```
//StarTest.java - display a starburst
import java.awt.*;
import javax.swing.*;
class StarTest {
  public static void main(String[] args) {
    JFrame frame = new JFrame("StarTest");
    Container pane = frame.getContentPane();
    Star star = new Star();
    pane.add(star);
    frame.pack();
    frame.show();
  }
}
```



21

```
import java.awt.*;
import javax.swing.*;
class Star extends JComponent {
  public void paint(Graphics g) {
    for (double angle = 0; angle < Math.PI;
        angle = angle + Math.PI / 16) {
      double x1, x2, y1, y2;
      // compute coordinates of endpoints of a line
      // cosine and sine range from -1 to 1
      // multiplying by RADIUIS changes the
      // range to be from -RADIUS to RADIUS
      // adding RADIUS gives the final range of
      // 0 to 2 * RADIUS
      x1 = Math.cos(angle) * RADIUS + RADIUS;
      y1 = Math.sin(angle) * RADIUS + RADIUS;
      x2 = Math.cos(angle + Math.PI) * RADIUS +RADIUS;
      y2 = Math.sin(angle + Math.PI) * RADIUS +RADIUS;
      g.drawLine((int)x1, (int)y1, (int)x2, (int)y2);
    }
  }
```

22

```
  // make the JComponent big enough to show the image
  public Dimension getMinimumSize() {
    return new Dimension(2 * RADIUS, 2 * RADIUS);
  }
  public Dimension getPreferredSize() {
    return new Dimension(2 * RADIUS, 2 * RADIUS);
  }
  private static final int RADIUS = 100;
}
```

What happens if these two methods are omitted?

23

FlowLayout

- The first example SampleAWT used a FlowLayout.
- If we resize the window, the components move around.

24

25

```
import javax.swing.*;
import java.awt.*;
class SampleAWT {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Sample AWT Components");
    Container display = frame.getContentPane();
    display.setLayout(new FlowLayout());
    display.add(new JLabel(
            "What programming languages do you know?"));
    display.add(new JLabel("Check all that apply:"));
    display.add(new JCheckBox("Java",true));
    display.add(new JCheckBox("C",false));
    display.add(new JCheckBox("C++",false));
    display.add(new JButton("Submit"));
    frame.pack();
    frame.show();
    System.out.println("The end of main().");
  }
}
```

26

```
import javax.swing.*;
import java.awt.*;
class SampleAWT {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Sample AWT Components");
    Container display = frame.getContentPane();
    display.setLayout(new FlowLayout(FlowLayout.LEFT));
    // code omitted
  }
}
```

We can have the FlowLayout center the components on a line, left justify them, or right justify them.

27

Mouse Input

- One of the earliest examples of a GUI and of using a Mouse, involved drawing on the screen with the mouse.
- Here is a program that lets us draw on the screen with the mouse.

28

```
//SimplePaint.java - a program to draw with the mouse
import java.awt.*;
import javax.swing.*;

class SimplePaint {
  public static void main(String[] args) {
    JFrame frame = new JFrame("SimplePaint");
    Container pane = frame.getContentPane();
    DrawingCanvas canvas = new DrawingCanvas();
    PaintListener listener = new PaintListener();
    canvas.addMouseMotionListener(listener);
    pane.add(canvas);
    frame.pack();
    frame.show();
  }
}
```

29

```
// DrawingCanvas.java - a blank Canvas
import java.awt.*;
import javax.swing.*;

class DrawingCanvas extends JComponent {
  //these methods are needed to give the component a size
  public Dimension getMinimumSize() {
    return new Dimension(SIZE, SIZE);
  }
  public Dimension getPreferredSize() {
    return new Dimension(SIZE, SIZE);
  }
  private static final int SIZE = 500;
}
```

30

```
//PaintListener.java - do the actual drawing
import java.awt.*;
import java.awt.event.*;

public class PaintListener implements MouseMotionListener
{
  public void mouseDragged(MouseEvent e) {
    DrawingCanvas canvas =
                  (DrawingCanvas)e.getSource();
    Graphics g = canvas.getGraphics();
    g.fillOval(e.getX() - radius, e.getY() - radius,
               diameter, diameter);
  }
  public void mouseMoved(MouseEvent e){}
  private int radius = 3;
  private int diameter = radius * 2;
}
```

31

Refreshing the display

- The previous program doesn't "remember" what was drawn.
- If we obscure part of the drawing, it won't come back.
- The following program remedies that problem.

32

```
//DrawingCanvas2.java - remember drawing operations
// using an offscreen image
import java.awt.*;
import javax.swing.*;

class DrawingCanvas2 extends JComponent {
  // transfer the offscreen image to the screen
  public void paint(Graphics g) {
    if (offscreenImage != null)
      g.drawImage(offscreenImage,0,0,SIZE,SIZE,null);
  }
```

33

```
  // return the offscreen image, if one doesn't exist
  // create one
  public Image getOffscreenImage() {
    if (offscreenImage == null)
      offscreenImage = createImage(SIZE, SIZE);
    return offscreenImage;
  }

  public Dimension getMinimumSize() {
    return new Dimension(SIZE, SIZE);
  }
  public Dimension getPreferredSize() {
    return new Dimension(SIZE, SIZE);
  }
  private static final int SIZE = 500;
  private Image offscreenImage;
}
```

34

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

public class PaintListener2 implements MouseMotionListener {
  public void mouseDragged(MouseEvent e) {
    DrawingCanvas2 canvas = (DrawingCanvas2)e.getSource();
    Graphics g = canvas.getGraphics();
    g.fillOval(e.getX() - radius, e.getY() - radius,
               diameter, diameter);
    // duplicate the drawing on the offscreen image
    Image image = canvas.getOffscreenImage();
    g = image.getGraphics();
    g.fillOval(e.getX() - radius, e.getY() - radius,
               diameter, diameter);
  }
  public void mouseMoved(MouseEvent e){}
  protected int radius = 3;
  protected int diameter = radius * 2;
}
```

35

Applets

- An Applet is a special Java program, intended to be executed inside of a Web browser.
- An Applet is a Java class that extends Applet or JApplet.
- An Applet is started when a special HTML tag is encountered in a Web document.

36

```
<html>
<body>
Below is the beginning of an applet calculator. This
calculator can be used only for addition and
subtraction. You enter a number in each of the first
...
<p>
<center>
<applet code="MiniCalcApplet.class"
width=200 height=100>
</applet>
<p>
MiniCalcApplet
</center>
</body>
</html>
```

37

```
// MiniCalcApplet.java
/* <applet code="MiniCalcApplet.class" width=200
height=100></applet> */
import javax.swing.*;
import java.awt.*;

public class MiniCalcApplet extends JApplet {
  public void init() {
    Container pane = getContentPane();
    // create the major components
    JTextField firstNumber = new JTextField(20);
    JTextField secondNumber = new JTextField(20);
    JTextField result = new JTextField(20);
    JButton addButton = new JButton("Add");
    JButton subButton = new JButton("Subtract");
```

38

```
    // there will be 4 rows of 2 components each
    pane.setLayout(new GridLayout(4, 2));
    // add all of the components to the content pane
    pane.add(new JLabel("Enter a number"));
    pane.add(firstNumber);
    pane.add(new JLabel("Enter a number"));
    pane.add(secondNumber);
    pane.add(new JLabel("Result"));
    pane.add(result);
    pane.add(addButton);
    pane.add(subButton);
    // setup the listener, listening to the buttons
    DoMath listener =
        new DoMath(firstNumber, secondNumber, result);
    subButton.addActionListener(listener);
    addButton.addActionListener(listener);
  }
}
```

39

```
//SimplePaintApplet.java - applet version of program
//    that draws with the mouse
/* <applet code="SimplePaintApplet.class"
width=500 height=500> </applet> */
import java.awt.*;
import javax.swing.*;

public class SimplePaintApplet extends JApplet {
  public void init() {
    Container pane = getContentPane();
    DrawingCanvas2 canvas = new DrawingCanvas2();
    PaintListener2 listener = new PaintListener2();
    canvas.addMouseMotionListener(listener);
    pane.add(canvas);
  }
}
```

40

```
/SimplePaintCombined.java - combine all of the
//  features of SimplePaint, PaintListener, and
//  DrawingCanvas into a single class.
//  This style of programming isn't recommended.
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class SimplePaintCombined
  extends JComponent implements MouseMotionListener
{
  private static final int SIZE = 500;
  public static void main(String[] args) {
    JFrame frame = new JFrame("SimplePaint");
    Container pane = frame.getContentPane();
    SimplePaintCombined canvas =
            new SimplePaintCombined();
```

41

```
    canvas.addMouseMotionListener(canvas);
    pane.add(canvas);
    frame.pack();
    frame.show();
  }
  public void mouseDragged(MouseEvent e) {
    SimplePaintCombined canvas =
      (SimplePaintCombined)e.getSource();
    Graphics g = canvas.getGraphics();
    g.fillOval(e.getX() - 3, e.getY() - 3, 6, 6);
  }
  public void mouseMoved(MouseEvent e) {}
  public Dimension getMinimumSize() {
    return new Dimension(SIZE, SIZE);
  }
  public Dimension getPreferredSize() {
    return new Dimension(SIZE, SIZE);
  }
}
```

42

Can a program be both an Applet and a regular application?

Yes, as shown in the following example.

43

```
// MiniCalcBoth.java
/* <applet code="MiniCalcBoth.class" width=200
height=100></applet> */
import javax.swing.*;
import java.awt.*;

public class MiniCalcBoth extends JApplet {
  public void init() {
      buildGUI(getContentPane());
  }
  public static void main(String[] args) {
      JFrame frame = new JFrame("MiniCalc");
      buildGUI(frame.getContentPane());
      frame.pack();
      frame.show();
  }
```

44

```
  private static void buildGUI(Container pane) {
    // create the major components
    JTextField firstNumber = new JTextField(20);
    ...
    JButton subButton = new JButton("Subtract");
    // there will be 4 rows of 2 components each
    pane.setLayout(new GridLayout(4, 2));
    // add all of the components to the content pane
    pane.add(new JLabel("Enter a number"));
    pane.add(firstNumber);
    ...
    pane.add(subButton);
    // setup the listener, listening to the buttons
    DoMath listener =
        new DoMath(firstNumber, secondNumber, result);
    subButton.addActionListener(listener);
    addButton.addActionListener(listener);
  }
}
```

45