


## CS102

Introduction to  
data structures, algorithms,  
and object-oriented  
programming

March 27, 2017


1



## Intro to GUI Programming

- **Definition:** GUI means "Graphical User Interface"
  - Fact: All GUI components know how to draw themselves.
  - You can give them additional properties/behaviors
  - We have already used one of the graphics components of a GUI – the JOptionPane class.

2




## Imports

For ease of programming and readability, GUI programs use the following import statements:

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

3



## JFrame

Fundamental component – a window.

- Can be opened, closed, and resized.
- Has "title" displayed in top bar.
- Doesn't have any content...you create content as shown in slides that follow.

```
JFrame window = new JFrame("title");
```


4

Some properties must be set before making the JFrame visible, inside either main, an instance method, or a constructor:

```
window.setContentPane(content);
window.setSize(250,100);
window.setLocation(100,100);
window.setDefaultCloseOperation
    (JFrame.EXIT_ON_CLOSE);
window.setPreferredSize(new Dimension(250,100));
window.pack();
window.setVisible(true);
```

The content variable in the 1<sup>st</sup> line is usually a JPanel that must have been instantiated before the 1<sup>st</sup> line executes.

5



## JPanel

Uses for JPanel:

1. **Draw something.**
2. Hold other components.

To make the entire class a JPanel, extend JPanel in the class signature or use an inner class that extends JPanel.

If you are using a JPanel as a drawing window, you should override the paintComponent method:

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    (drawing operations with g) }
```

6

If the JPanel is to be used for drawing:

1. Write drawing method **paintComponent** that is passed a Graphics object by the system when execution starts.
2. Pass the Graphics object to another method if any substantial coding need be done (e.g., drawing multiple shapes, using decisions, loops).
3. Re-call paintComponent by adding a call to repaint(). The repaint method is not written in the class you write...it is a call to make the system call the paintComponent method.

7

## Simple "drawing" GUI

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/**
 * HelloWorldGUI - This class draws a greeting using Graphics
 */
public class HelloWorldGUI extends JPanel {
    JPanel content; // JPanel to hold this drawing surface
    public static void main(String[] args) {
        HelloWorldGUI displayPanel = new HelloWorldGUI();
    } // end main
    public void paintComponent(Graphics g) {
        super.paintComponent(g); // overrides parent's method
        g.drawString("Hello World!", 20, 30);
    } // end paintComponent
```

8

```
/**
 * Physical layout of GUI is set up in constructor
 * or an instance method.
 */
public HelloWorldGUI() {
    content = new JPanel();
    content.setSize(250,100);
    content.setLayout(new BorderLayout());
    // 'this' is a HelloWorldGUI object added to middle
    content.add(this, BorderLayout.CENTER);

    JFrame window = new JFrame("GUI Test");
    window.setBackground(Color.PINK);
    window.setContentPane(content);
    window.setLocation(100,100);
    window.setPreferredSize(new Dimension(250,100));
    window.pack();
    window.setVisible(true);
} // end of constructor
} // end of class
```

9

## Layout Managers

A **BorderLayout** positions items in a container, arranging and resizing its components to fit in five regions: NORTH, SOUTH, EAST, WEST and CENTER.

```
JPanel p = new JPanel();
p.setLayout(new BorderLayout());
p.add(new JButton("Okay"), BorderLayout.SOUTH);
```

The **GridLayout** class is a layout manager that positions a container's components in a rectangular grid. The container is divided into equal-sized rectangles, and one component is placed in each rectangle. Good for completely filling an area with a set of components. Arguments to constructor are at least the number of rows and number of columns

10

## JPanel

Uses for JPanel:

1. Draw something.
2. Hold other components that create events.

11

## JPanel (cont.)

Components must be added to a container before they are visible. In the code sample below (from the method where the GUI is set up), a JButton is created and added to the JPanel.

```
(quitButton and content are instance variables)
quitButton = new JButton("Quit");
quitButton.addActionListener(this);
content = new JPanel();
content.setLayout(new BorderLayout());
content.add(quitButton, BorderLayout.SOUTH);
```

12

## JPanel (cont.)

General technique for setting up the components of a GUI:

1. Create a JFrame as a window to hold everything.
2. Instantiate a JPanel as a container in the window.
3. Assign a layout manager to container.
4. Instantiate components and add them to the container.
5. Set the container as the content pane of the window.

13

## Events and Listeners

The structure of containers and components make up the physical appearance of the GUI, but do not set up the behavior.

GUIs are event-driven—the program responds to events with event-handling methods:

```
public void actionPerformed(ActionEvent e) {
    System.exit(0);
}
```

... from within method setting up components as listeners:  
quitButton.addActionListener(listener);

14

## Programming behavior of GUI

General technique for setting up behavior of GUI:

1. Write event-handling methods. For ActionEvents, the class signature includes "implements ActionListener" and the event-handling method is actionPerformed.
2. Create objects and register them as listeners on components that generate ActionEvents.
3. When events occur, the appropriate listener is notified and its actionPerformed method is executed.

15

## Simple "Component" GUI

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/**
 * ComponentGUI
 * This class shows one JButton that closes the window
 */
class ComponentGUI extends JPanel implements ActionListener {
    JPanel content; // JPanel to hold components inside JPanel
    JButton quitButton; // component: a push button

    public static void main(String[] args) {
        ComponentGUI cGUI = new ComponentGUI();
    } // end main

    public void actionPerformed(ActionEvent evt) {
        System.exit(0);
    }
}
```

16

```
public ComponentGUI() {
    // instantiate JButton component and add ActionListener
    quitButton = new JButton("Quit");
    quitButton.addActionListener(this);

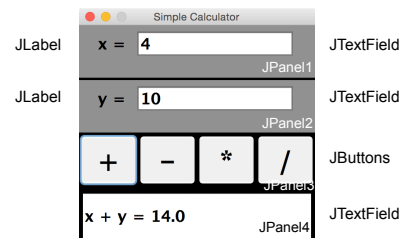
    // instantiate JPanel component, set layout, and add JButton
    content = new JPanel();
    content.setLayout(new BorderLayout());
    content.add(quitButton, BorderLayout.CENTER);

    // instantiate window and add JPanel and its contents
    JFrame window = new JFrame("GUI with JButton");
    window.setContentPane(content);
    window.setSize(250,100);
    window.setLocation(100,100);
    window.setVisible(true);
}
}
```

17

## More Complex GUI

The GUI we'll look at next has a main JPanel with 4 nested JPanels.



18



## Putting a GUI class together

Writing a GUI can be done in many different ways. I will cover the setup for the simple calculator on the last slide:

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

/**
 * This class uses nested JPanels to create a simple calculator.
 */
public class SimpleCalculator extends JPanel implements ActionListener {
```

The SimpleCalculator class is-a JPanel and is-of ActionListener type.

19

After the class signature, declare all the class instance variables (you will often add these as you discover they are needed in the code).

```
JPanel calcPanel, panelX, panelY, buttonPanel, resultPanel;
JButton plus, minus, mult, div;
JLabel xEqual, yEqual;
JTextField enterX, enterY, answer;

public static void main(String[] args) {
    SimpleCalculator sgs = new SimpleCalculator();
} // end main

public SimpleCalculator() {
    JFrame bigPane = new JFrame("Simple Calculator");
    bigPane.setLayout(null);
    bigPane.setBackground(Color.BLACK);
    bigPane.setLocation(100,50);

    calcPanel = new JPanel(); // panel to hold all others
    calcPanel.setLayout(new GridLayout(4,1,3,3));
    bigPane.setContentPane(calcPanel);
```

20

Each JPanel is instantiated in the constructor, the layout manager is set up for each one, and other properties are set. Here, panelX and panelY are instantiated.

```
panelX = new JPanel();
panelX.setBackground(Color.GRAY);
panelX.setLayout(new FlowLayout());
enterX = new JTextField("0", 10);
Font bigText = new Font("SansSerif",Font.BOLD,20);
enterX.setFont(bigText);

panelY = new JPanel();
panelY.setBackground(Color.GRAY);
panelY.setLayout(new FlowLayout());
enterY = new JTextField("0", 10);
enterY.setFont(bigText);
```

21

panelX and panelY each contain a JLabel and a JTextField that are instantiated and added to each panel. Then each panel is added to calcPanel.

```
xEqual = new JLabel("x = ");
xEqual.setFont(bigText);
yEqual = new JLabel("y = ");
yEqual.setFont(bigText);
panelX.add(xEqual);
panelX.add(enterX);
panelY.add(yEqual);
panelY.add(enterY);

calcPanel.add(panelX);
calcPanel.add(panelY);
```

22

Instantiate each JButton and add "this" as the action listener

```
Font biggerText = new Font("SansSerif",Font.BOLD,36);
plus = new JButton("+");
plus.setFont(biggerText);
plus.addActionListener(this);
minus = new JButton("-");
minus.setFont(biggerText);
minus.addActionListener(this);
mult = new JButton("*");
mult.setFont(biggerText);
mult.addActionListener(this);
div = new JButton("/");
div.setFont(biggerText);
div.addActionListener(this);
```

23

Finish the constructor by adding buttons to buttonPanel (after the layout manager is specified. Add all JPanels to calcPanel and finish setting up JFrame.

```
buttonPanel = new JPanel();
buttonPanel.setLayout(new GridLayout(1,1));
buttonPanel.add(plus);
buttonPanel.add(minus);
buttonPanel.add(mult);
buttonPanel.add(div);

bigPane.setPreferredSize(new Dimension(300,300));
calcPanel.add(panelX);
calcPanel.add(panelY);
calcPanel.add(buttonPanel);
calcPanel.add(resultPanel);

bigPane.pack();
bigPane.setLocation(100,50);
bigPane.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
bigPane.setResizable(false);
bigPane.setVisible(true);
} // end constructor
```

24

```

public void actionPerformed(ActionEvent evt){
    double x, y;
    String xStr, yStr;
    // first, get the text from the JTextFields
    try {
        xStr = enterX.getText();
        x = Double.parseDouble(xStr);
    } catch (NumberFormatException nfe) {
        answer.setText("Illegal data for x.");
        enterX.requestFocusInWindow();
        return;
    }
    try {
        yStr = enterY.getText();
        y = Double.parseDouble(yStr);
    } catch (NumberFormatException nfe) {
        answer.setText("Illegal data for y.");
        enterY.requestFocusInWindow();
        return;
    }
}

```

25

```

String op = evt.getActionCommand();
if (op.equals("+"))
    answer.setText("x + y = " + (x+y));
else if (op.equals("-"))
    answer.setText("x - y = " + (x-y));
else if (op.equals("*"))
    answer.setText("x * y = " + (x*y));
else if (op.equals("/")) {
    if (y == 0)
        answer.setText("Can't divide by zero.");
    else
        answer.setText("x / y = " + (x/y));
} // end if
} // end actionPerformed
} // end class

```

26

## Main points in General GUI

1. Write class that implements all Listener interfaces needed.
2. Decide which JComponents you need and declare them as instance variables.
3. Write a main method that creates an object of its own type, calling a zero-parameter constructor.
4. Inside the constructor, create a JFrame to hold all JComponents. Instantiate all JComponents in constructor. Add a Listener to any JComponent that will generate an Event. Add all JComponents to their appropriate containers.

27

## Main points (cont.)

5. Write an actionPerformed method to respond to any Events generated (in this case, only the JButtons generate ActionEvents).

28

## Writing JPanels to do both

Uses for JPanel:

1. Can add other components.
2. Draw something.

29

1. Write class that extends JPanel and implements ActionListener interface.
2. Decide which JComponents you need and declare them as instance variables.
3. Write a main method that creates an object x of its own type. Instead of using the constructor to set up the window, call an instance method and pass x into the method.
4. Inside the instance method you created in step 3, create a JFrame to hold all JComponents. Instantiate and set up all JComponents in constructor. Add all JComponents to their appropriate containers.

30

6. Add a Timer object to the method that sets up the window to generate ActionEvents for continuous motion:

```
Timer frameTimer = new Timer(20, this);
frameTimer.start();
```

7. override the method:

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    (call drawing method, passing in g)
}
```

31

8. Write an actionPerformed method that will be sent an ActionEvent for every clock tick.
9. Write a method that takes a Graphics object as an argument and uses it to create any shapes you need on the window.

32