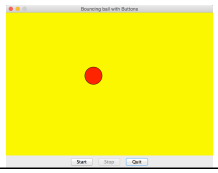
CS102

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

March 29, 2017

Lab recap

1. If you are going to draw an image (like a solid colored circle), include a method called paintComponent.
2. All simple shape drawing must be done with a Graphics object, generated by the system.
3. If you are going to add GUI components, you should attach listeners to those components that generate ActionEvents (JButtons, JTextField, JTextArea, JMenuItems, Timer, and more)



Lab recap

4. If you are going to add JComponents, You need to add them either to a JFrame (the basic window) or to another Container like a JPanel.
5. We add "this", meaning the BouncingBall JPanel sub-class as the actionListener on all objects that generate ActionEvents. There is no need to add a listener for ActionEvents created by the Timer.
6. Adding every component ActionListener as "this" may necessitate complex decision statements in the actionPerformed method:

```
if (evt.getSource() instanceof JButton) {
    if (evt.getActionCommand.equals("Quit") {
        System.exit(0);
```


Lab recap

7. There are different ways to handle events. One way doesn't require implementing the ActionListener interface ("Anonymous inner classes").
8. Layout managers are generally a good idea if your screen is broken into portions that look good. If you you want, you can use a null layout manager, meaning you specify the x,y coordinates and size of every component.

Lab recap

In Lab 6, you wrote an event-driven simulator to make the ball move with every clock tick.

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

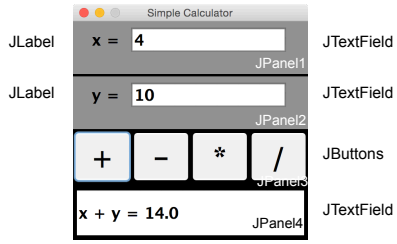
The Timer ActionEvents might interfere with the reaction to other events.

Adding Menus to GUI

1. Create a JMenuBar object and add it to a JPanel.
2. Create a JMenu object.
3. Create JMenuItem objects and add them to the JMenu object.
4. Add the JMenu object to the JMenuBar object.
5. Add JMenuBar object to the content pane.

More Complex GUI

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



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.

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);
```


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);
```


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);
```


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);));
```

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
```

13

```
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;
      }
```

14

```
        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
```

15

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.

16

Main points (cont.)

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

17

Writing JPanels to do both

Uses for JPanel:

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

18

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.

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)
}
```


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.

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);
```


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);
```


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);
```

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);));
```

25

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
```

26

```
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;
    }
```

27

```
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
```

28

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.

29

Main points (cont.)

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

30

Writing JPanels to do both

Uses for JPanel:

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

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.

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)
}
```


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.