Programming Languages and Techniques (CIS120)

Lecture 33

November 20, 2015

Swing I: Drawing and Event Handling

Chapter 29

Announcements

- HW8: Spellchecker
  - Available on the web site
  - Due: Tuesday, November 24th
  - Parsing, working with I/O, more practice with collections
- Next Week: No Lab Sections
- Next Wednesday: Bonus Lecture
  "Consequences of Code as Data"
  - Attendance not required (but encouraged if you are around!)

Poll

Have you started HW 08 (Spellchecker) Yet?

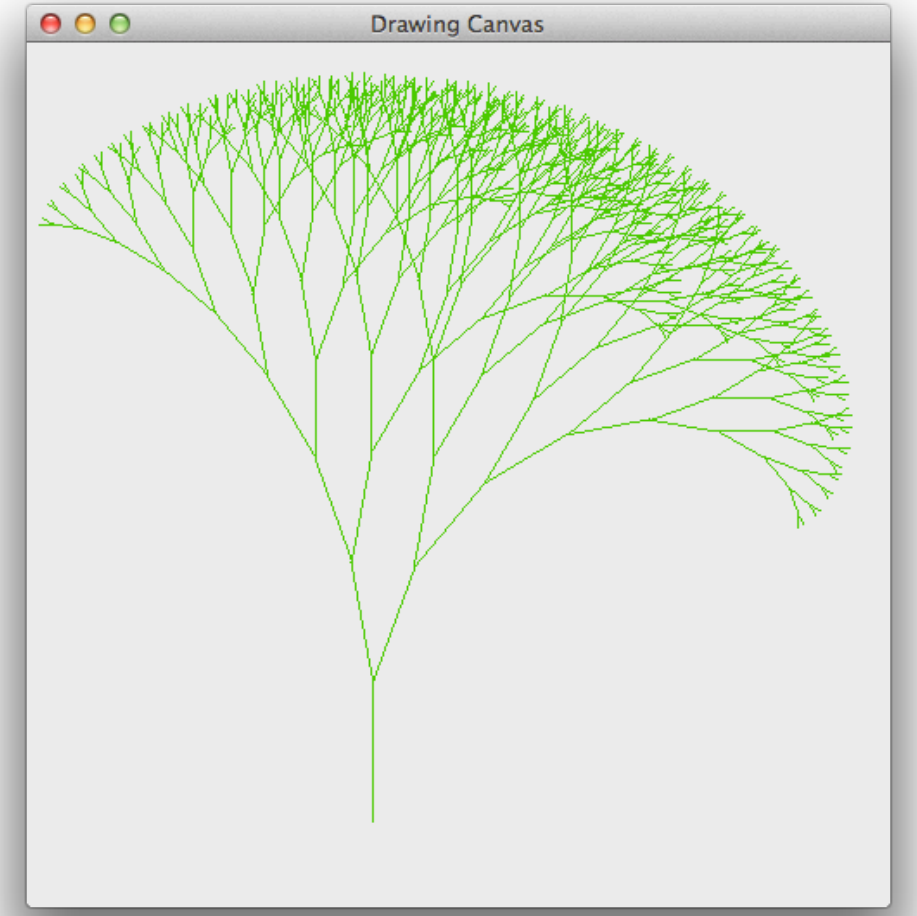
1. Not at all
2. I've downloaded it
3. Partway through
4. Finished!

Simple Drawing

DrawingCanvas.java

DrawingCanvasMain.java

Fractal Drawing Demo



Simple Drawing Component

```
public class DrawingCanvas extends JComponent {

    public void paintComponent(Graphics gc) {
        super.paintComponent(gc);

        // set the pen color to green
        gc.setColor(Color.GREEN);

        // draw a fractal tree
        fractal (gc, 75, 100, 270, 15);
    }

    // get the size of the drawing panel
    public Dimension getPreferredSize() {
        return new Dimension(150,150);
    }
```

How to display this component?

JFrame

- Represents a top-level window
  - Displayed directly by OS (looks different on Mac, PC, etc.)
- Contains JComponents
- Can be moved, resized, iconified, closed

```
public void run() {
    JFrame frame = new JFrame("Tree");

    // set the content of the window to be the drawing
    frame.getContentPane().add(new DrawingCanvas());

    // make sure the application exits when the frame closes
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // resize the frame based on the size of the panel
    frame.pack();

    // show the frame
    frame.setVisible(true);
}
```

User Interaction

Start Simple: Lightswitch Revisited

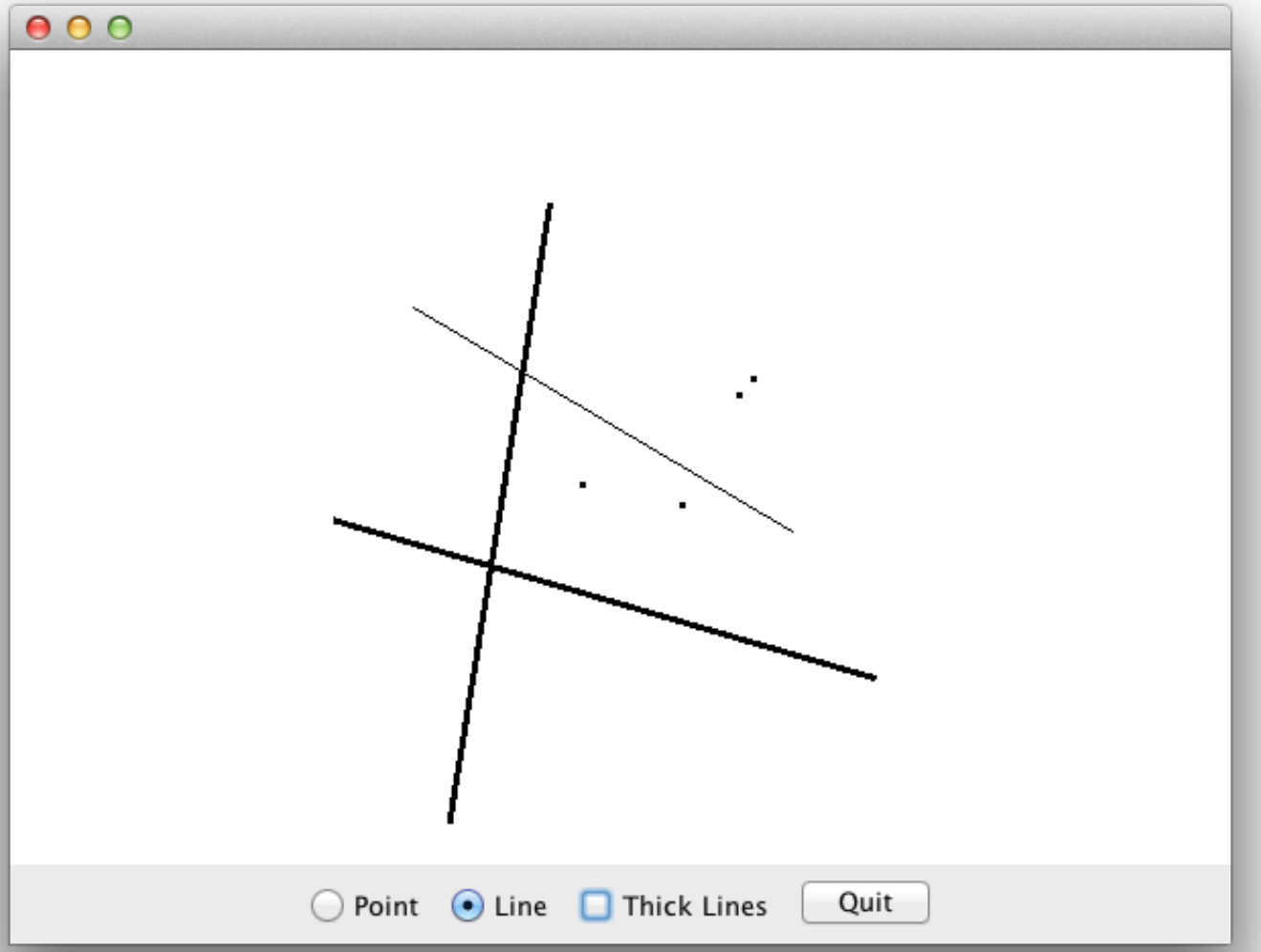
Task: Program an application that displays a button. When the button is pressed, it toggles a “lightbulb” on and off.

OnOffDemo

The Lightswitch GUI program in Swing.

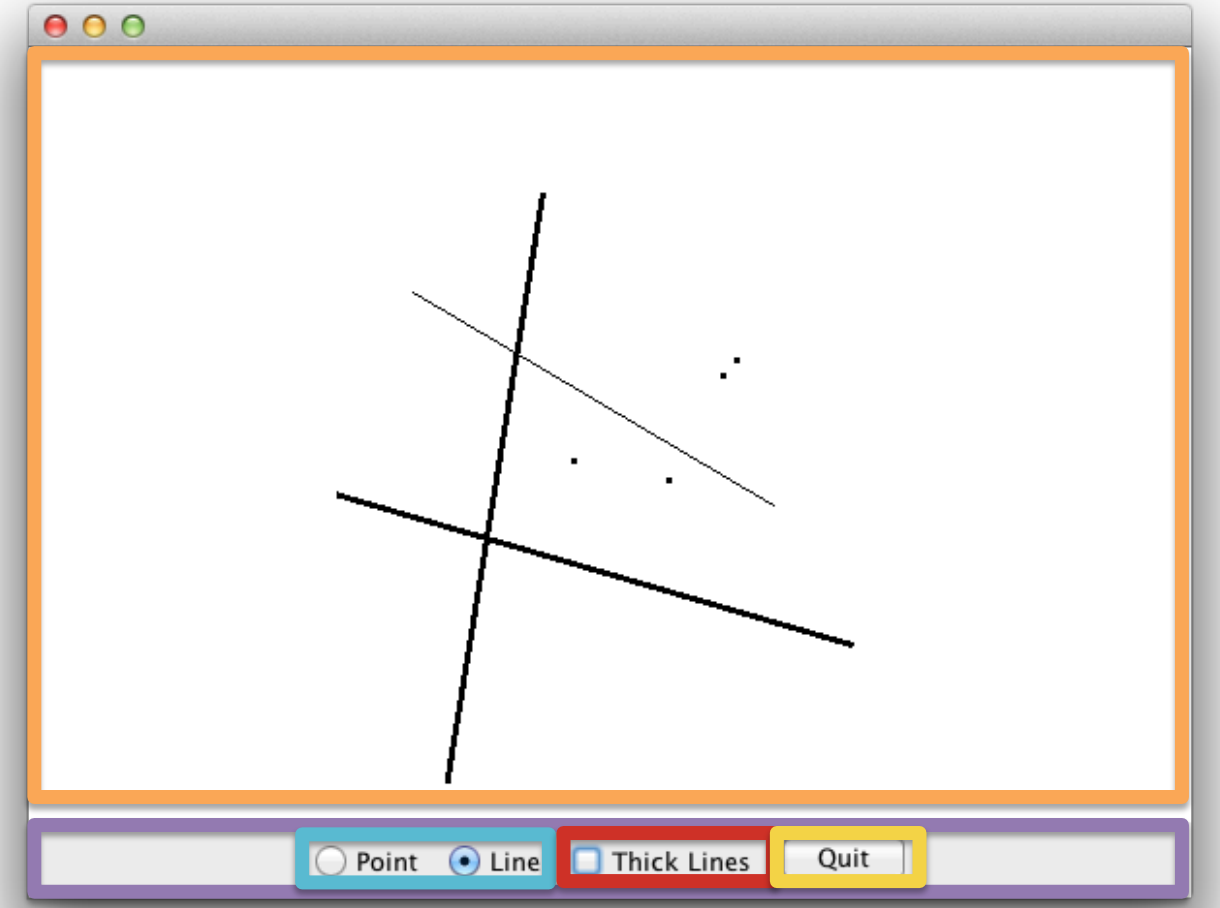
Swing Programming Demo

Layout



What layout would you use for this app? What components would you use?

Canvas
subclass of
JPanel
(canvas)



JPanel
(toolbar)

JRadioButton
(point, line)

JCheckbox
(thick)

JButton
(quit)

Inner Classes

Inner Classes

- Useful in situations where two objects require "deep access" to each other's internals
- Replaces tangled workarounds like "owner object"
  - Solution with inner classes is easier to read
  - No need to allow public access to instance variables of outer class
- Also called "dynamic nested classes"

Basic Example

Key idea: Classes can be *members* of other classes…

```
class Outer {
  private int outerVar;
  public Outer () {
    outerVar = 6;
  }
  public class Inner {
    private int innerVar;
    public Inner(int z) {
      innerVar = outerVar + z;
    }
    public int getInnerVar() {
      return innerVar;
    }
  }
}
```

Name of this class is Outer.Inner
(which is also the static type of objects that this class creates)

Reference from inner class to instance variable bound in outer class

Constructing Inner Class Objects

Based on your understanding of the Java object model, which of the following make sense as ways to construct an object of an inner class type?

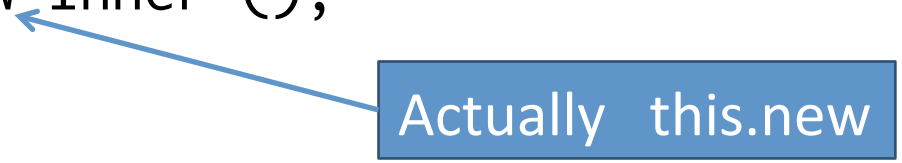
1. Outer.Inner obj = new Outer.Inner()
2. Outer.Inner obj = (new Outer()).new Inner();
3. Outer.Inner obj = new Inner();
4. Outer.Inner obj = Outer.Inner.new ()

CIS 120

Object Creation

- Inner classes can refer to the instance variables and methods of the outer class
- Inner class instances usually created by the methods/constructors of the outer class

```
public Outer () {
    Inner b = new Inner ();
}
```



- Inner class instances *cannot* be created independently of a containing class instance.

```
Outer.Inner b = new Outer.Inner()
```

```
Outer a = new Outer();
Outer.Inner b = a.new Inner();
```

```
Outer.Inner b = (new Outer()).new Inner();
```

Anonymous Inner Classes

- Define a class and create an object from it all at once, inside a method

```
quit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
 });
```

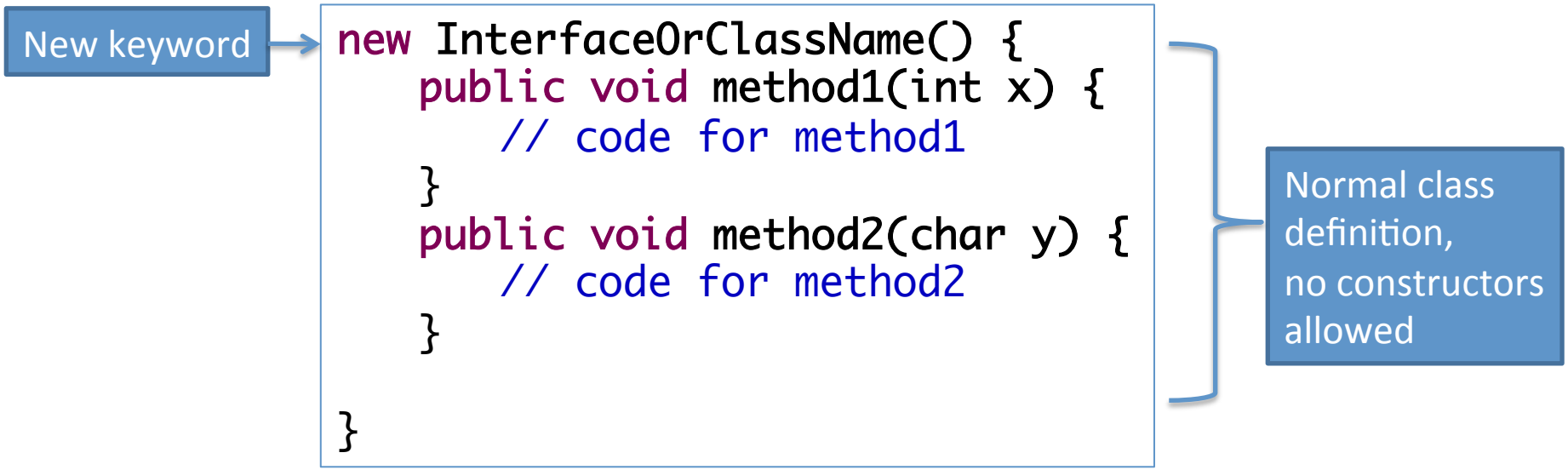
Puts button action right with button definition

```
line.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        shapes.add(new Line(…));
        canvas.repaint();
    }
});
```

Can access fields and methods of outer class, as well as final local variables

Anonymous Inner class

- New *expression* form: define a class and create an object from it all at once



Static type of the expression is the Interface/superclass used to create it

Dynamic class of the created object is anonymous! Can't refer to it.

Like first-class functions

- Anonymous inner classes are the real Java equivalent of Ocaml first-class functions
- Both create "delayed computation" that can be stored in a data structure and run later
  - Code stored by the event / action listener
  - Code only runs when the button is pressed
  - Could run once, many times, or not at all
- Both sorts of computation can refer to variables in the current scope
  - OCaml: Any available variable
  - Java: only instance variables (fields) and variables marked final

Attendance

Did you attend class today.

1. YES