

Programming Languages and Techniques (CIS120)

Lecture 33

April 8, 2013

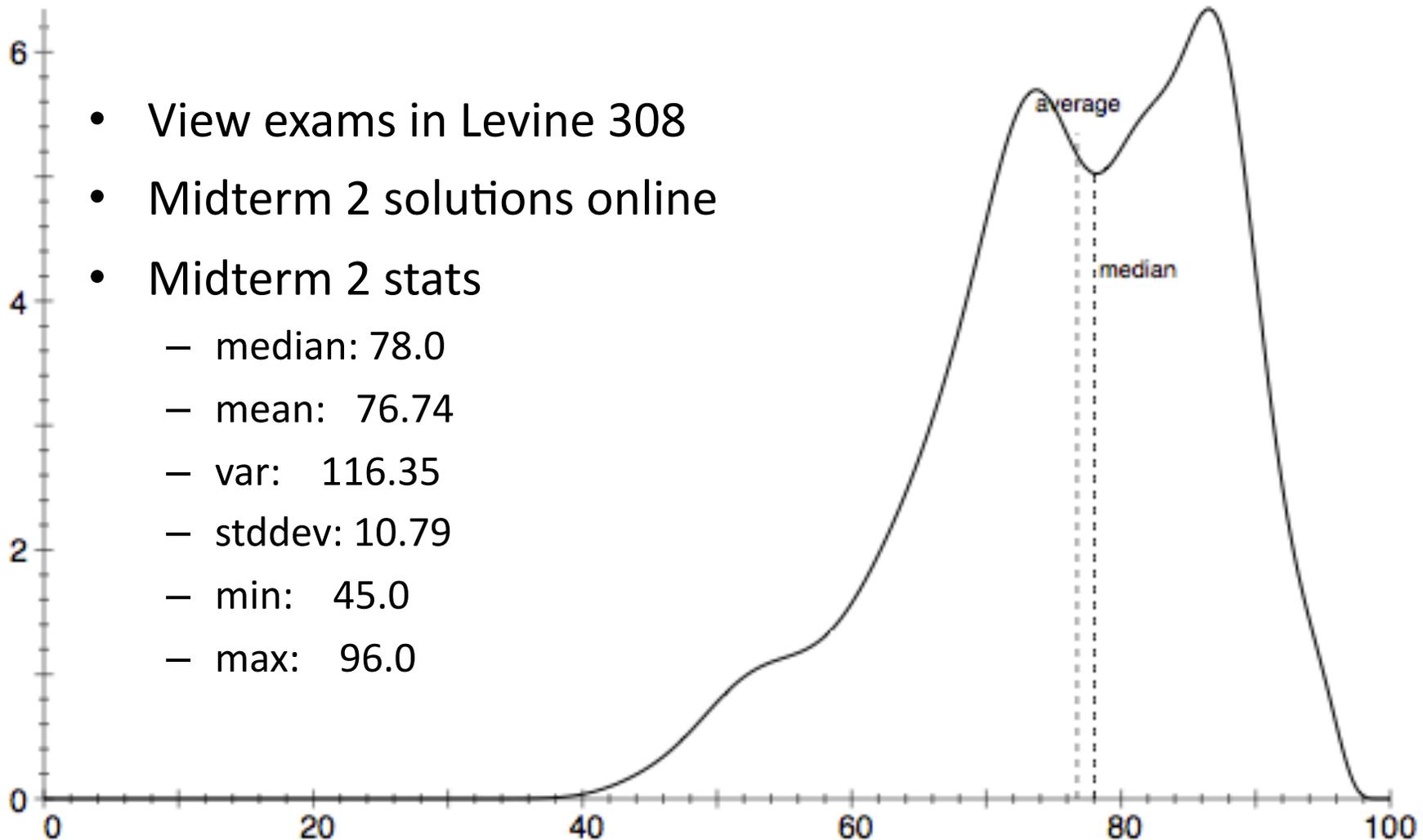
Swing I

Announcements

- HW 09 Due tomorrow at midnight

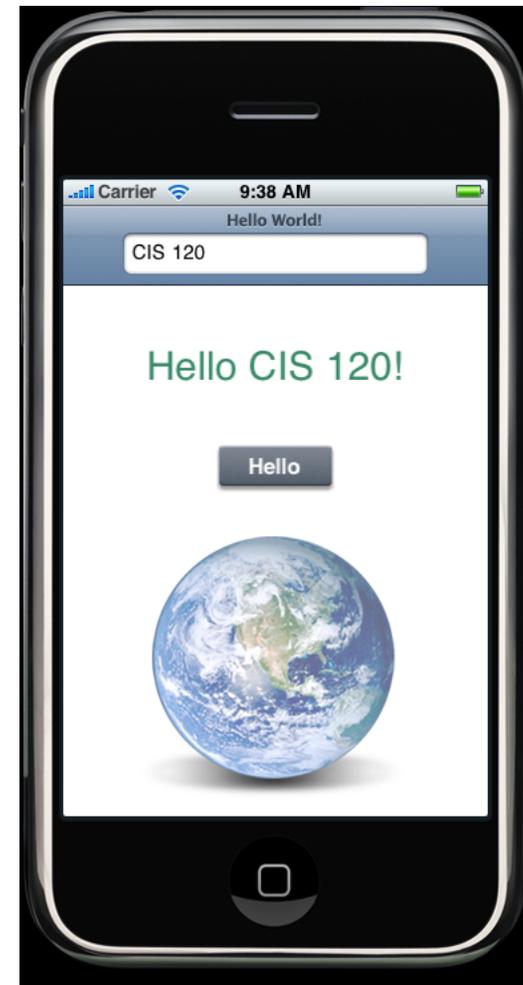
- View exams in Levine 308
- Midterm 2 solutions online
- Midterm 2 stats

- median: 78.0
- mean: 76.74
- var: 116.35
- stddev: 10.79
- min: 45.0
- max: 96.0



Why study GUIs (again)

- Most common example of event-based programming
- Heavy and effective use of OO inheritance
 - Nice opportunity to compare our “hand-rolled objects” in OCaml with those supported by Java’s rich object system
- Experience using references for communication
- Case study of library organization
- Fun!



OCaml GUI review

- Graphics Context (gctx.ml)
 - Provides drawing operations
 - Translates coordinates so that they are relative to each widget
 - Keeps track of state necessary for drawing (pen color, line thickness)
- Widgets (widget.ml)
 - Abstract type for "things" on the screen
 - In OCaml, a record of three first-class functions

```
type t = { repaint : Gctx.t -> unit,  
           size    : Gctx.t -> int,  
           handle  : event -> unit }
```

- basic widgets: buttons, canvas, scrollbars, labels, checkboxes, radiobuttons
 - container widgets: border, hpair, vpair, hlist, vlist
- Event Listeners
 - Functions that execute when events happen
 - Update the state of the application
 - Widgets reflect changes when they are redrawn

Comparison overview

	OCaml	Java
Graphics Context	Gctx.t	Graphics
Widget type	Widget.t	JComponent
Basic Widgets	button label checkbox	JButton JLabel JCheckBox
Container Widgets	hpair, vpair	JPanel, Layouts
Events	event	ActionEvent MouseEvent KeyEvent
Event Listener	mouse_listener mouseclick_listener (any function of type event -> unit)	ActionListener MouseListener KeyListener

CIS 120 Concepts from OCaml GUI assignment have analogues in Java swing library

Simple Drawing

DrawingCanvas.java

DrawingCanvasExample.java

How do we draw a picture?

- In OCaml, create a widget where the repaint function uses the graphics context to draw an image

```
let w_draw =  
{  
  repaint = (fun (gc:Graphics.t) ->  
             Graphics.draw_line gc (0, 0) (100, 100);  
             Graphics.draw_point gc (3,4)) ;  
  
  size      = (fun (gc:Graphics.t) -> (200,200));  
  
  handle    = (fun () -> ())  
}
```

- In Java, extend from class JComponent....

Fundamental class: JComponent

- Analogue to Widget.t (*Terminology*: widget == component)
- Subclasses *override* methods
 - paintComponent (like repaint, displays the component)
 - getPreferredSize (like size, calculates the size of the component)
 - (Events handled by subclasses)
- Much more functionality available
 - minimum/maximum size
 - font
 - foreground/background color
 - borders
 - what is visible
 - many more...

Simple Drawing Component

```
public class DrawingPanel 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);  
    }  
}
```

Instead of a record with first-class functions, we use an object with methods

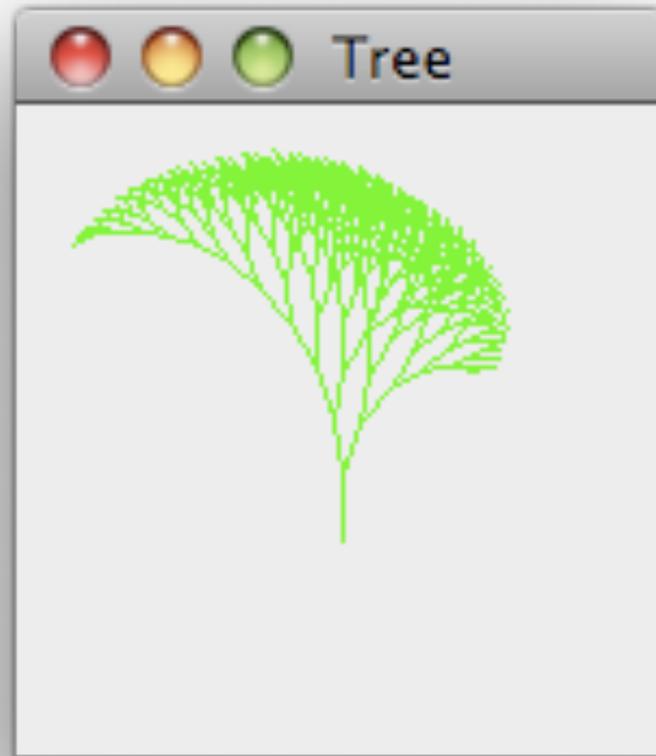
How to display this component?

JFrame

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

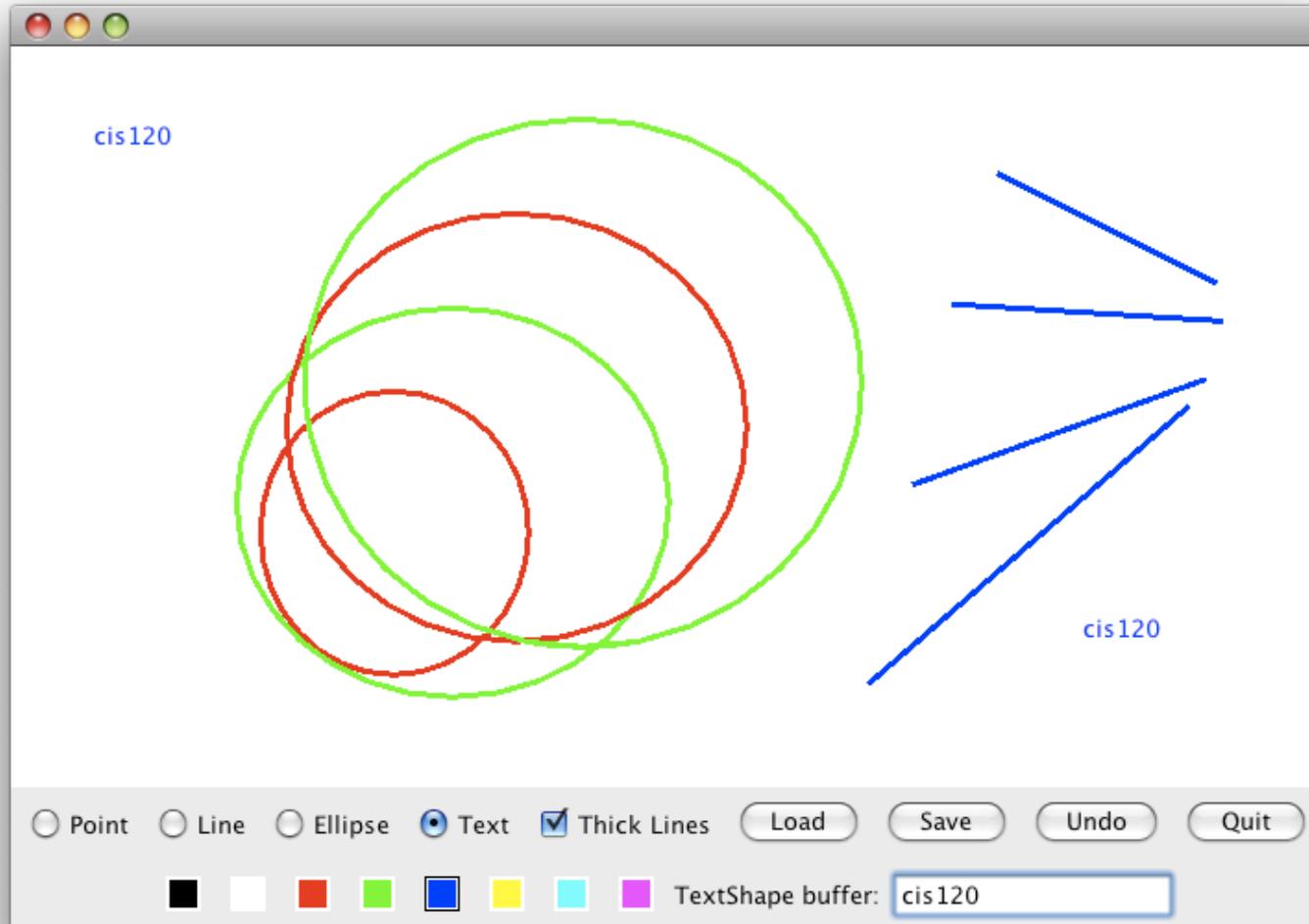
```
public void run() {  
    JFrame frame = new JFrame("Tree");  
    // set the content of the window to be the drawing  
    frame.getContentPane().add(new DrawingPanel());  
  
    // make sure the application exits when the frame closes  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    // resize the frame based on the size of the panel  
    frame.pack();  
  
    // show the frame  
    frame.setVisible(true);  
}
```

Fractal Drawing Demo



User Interaction

JavaPaint?



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 Java.

Swing practicalities

- Java library for GUI development
 - javax.swing.*
- Built on existing library: AWT
 - java.awt.*
 - If there are two versions of something, use Swing's. (e.g., java.awt.Button vs. javax.swing.JButton)
 - The “Jxxx” version is usually the one you want, rather than “xxx”.
- Portable
 - Communicates with OS's native window system
 - Same Java program looks different when run on PC, Linux and Mac
- Components as Containers
 - Use JPanel to organize and position other components (like vpair, hpair)