

Objectives

- Good enough design
- Introduction to GUIs in Java

Nov 3, 2008

Sprenkle - CS209

1

Wrapup of Lendables

- Comparison of Metrics:

Metric	Original	Refactored
Lines of Code	526	511
# Classes	7	8
# Methods	48	45
# Attributes	15	17

→ Decreased lines of code while increasing flexibility, extensibility

- Warning metric

➤ # of Parameters/Method

- Lendable constructor with 6 parameters

Nov 3, 2008

Sprenkle - CS209

2

Assignment 11 Discussion

- Last time: discussed how to handle 0 and 37 (which represent 0 and 00, which are green)
 - Talked about adding constants to the Wheel class and perhaps methods
 - Came up with ideas?

Nov 3, 2008

Sprenkle - CS209

3

Questions about Assignment 11?

Nov 3, 2008

Sprenkle - CS209

4

Refactoring Summary

- Write code and then rewrite code
 - Eye toward extensibility, flexibility, maintainability, and readability
 - Maintain correctness
- Reading/understanding other people's code can be difficult
 - Make your code readable, understandable
- Probably impossible to design/write "correctly" the first time
 - A lot harder to get the logic right, make sure you're not creating bugs, know/check the right answer...
 - Could cause yourself headaches coding this way first

Nov 3, 2008

Sprenkle - CS209

5

Good-Enough Design in the Large Discussion

Perfect Design

- ✓ Follows all design principle
- OCP, Single Responsibility, no code smells, ...
- ✓ May not be possible
- Infinite refactoring, development
- Code never released

Good-enough Design

- Not everyone agrees on design
- Maintenance requires changes to a few places
- ✓ Code gets released to customers

Similar tradeoffs in testing

Nov 3, 2008

Sprenkle - CS209

6

Grading Testing Project

- If a test case passes, what does that mean?
- If a test case fails, what does that mean?

Nov 3, 2008

Sprenkle - CS209

7

Grading Testing Project

- If a test case passes, what does that mean?
 - Code is correct (positive)
 - Code has a bug that test case doesn't reveal (false negative)
- If a test case fails, what does that mean?
 - Code has a bug (negative)
 - Or doesn't follow your specification
 - Code is correct but test case is incorrect (false positive)
- Are there any other possibilities?

Nov 3, 2008

Sprenkle - CS209

8

Grading Testing Project

		Test Case Result	
		Pass	Fail
Application	Correct	True Positive	False Negative
	Faulty	False Positive	True Negative

- Which is worse: false positives or false negatives?

Nov 3, 2008

Sprenkle - CS209

9

My Process

Nov 3, 2008

Sprenkle - CS209

10

PROGRAMMING PARADIGMS

Nov 3, 2008

Sprenkle - CS209

11

Programming Paradigms

- Our focus has been Object-oriented and Procedural paradigms
- Other paradigms
 - Event-driven
 - GUIs, Web applications
 - Distributed
 - Web applications, Grid
 - Concurrent
 - Parallel
 - Aspect-oriented

Blurred lines
between paradigms

Nov 3, 2008

Sprenkle - CS209

12

GUIs IN JAVA

Nov 3, 2008

Sprenkle - CS209

13

AWT & Swing

- AWT: Abstract Windowing Toolkit
 - Original GUI toolkit
 - Relies on operating system to render GUIs
 - Match look and feel of platform
 - Classes in `java.awt.*`
- Swing: new with Java2
 - Classes in `javax.swing.*`
 - Extends AWT
 - Provides Java look and feel for applications
 - But can plug in other look & feels

Nov 3, 2008

Sprenkle - CS209

14

Swing & AWT

- Swing does not completely replace AWT
- Using the Swing graphics programming model
 - Improves performance
 - Allows more efficient development of GUIs
- We will write graphics code that uses Swing
- We may return to AWT later
 - what AWT offers that Swing does not

Nov 3, 2008

Sprenkle - CS209

15

Swing: Made up of Components

- Top-level components
 - `JFrame`, `JWindow`, `JDialog`, `JApplet`
- GUI Elements
 - `JButton`, `JLabel`, `JMenuBar`, ...

Nov 3, 2008

Sprenkle - CS209

16

Frames

- Most basic unit of graphics programming
- Example of a container
 - A container contains other UI components
- A window that is not contained within another window
 - i.e., a top-level window
- `JFrame` Swing class implements a frame

Nov 3, 2008

Sprenkle - CS209

17

Example Frame

```
public class Game extends JFrame implements
KeyListener {

    public static void main(String[] args) {
        Game session = new Game();
        session.init();
    }

    public void init() {
        // Top-left corner is (0,0)
        // width/height: XBOUND, YBOUND
        setBounds(0, 0, XBOUND, YBOUND);
        // Shows the window
        setVisible(true);
    }
}
```

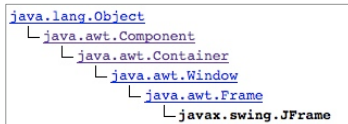
Nov 3, 2008

Sprenkle - CS209

18

Frame Inheritance

- JFrame is derived from `java.awt.Frame`
 - `Frame` class is derived from `Container` class
 - Container: anything that can contain UI components
- Class hierarchy



Yikes!
Don't get lost!

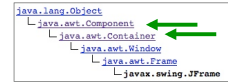
Nov 3, 2008

Sprenkle - CS209

19

Components & Containers

- Component
 - Abstract class
 - Everything you see is a component
 - Superclass of Container
 - Many methods
 - Some deprecated: be careful
- Container
 - Concrete implementation of Component
 - Base class of many classes
 - Can add and remove components to container



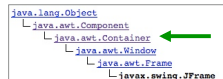
Nov 3, 2008

Sprenkle - CS209

20

Container Methods

- `setSize()`
 - Sets size of frame in pixels
- `setLocation()`
 - Sets location of frame
 - Coordinates of top-left corner
- `setBounds()`
 - Sets both size and location of frame
 - Provides information needed for `setSize()` and `setLocation()`



Nov 3, 2008

Sprenkle - CS209

21

Container Methods

- `getSize()`
 - Returns size of frame
- `getLocation()`
 - Returns current location of frame, relative to enclosing container
- `getLocationOnScreen()`
 - Returns current location of frame, using absolute screen coordinates



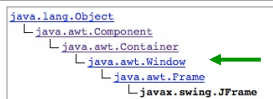
Nov 3, 2008

Sprenkle - CS209

22

Window Methods

- Top-level window
- No borders
- No Menu Bar
- `dispose()`
 - Closes window and reclaims resources associated with it
- `toBack()`
 - Sends window to back, may lose focus/activation
- `toFront()`
 - Bring to front, make this the focused window



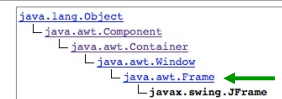
Nov 3, 2008

Sprenkle - CS209

23

Frame's Methods

- Top-level window with title and borders
- `setTitle()`
 - Sets title of frame (displayed in title bar)
- `setResizable()`
 - Can the user resize the frame?



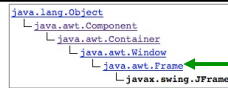
Nov 3, 2008

Sprenkle - CS209

24

Frame Methods

- `getExtendedState()`
- `setExtendedState(int state)`
- States (defined constants):
 - `NORMAL`
 - `ICONIFIED`
 - `MAXIMIZED_HORIZ`
 - `MAXIMIZED_VERT`
 - `MAXIMIZED_BOTH`



Nov 3, 2008

Sprenkle - CS209

25

Screen Resolution

- Since people use various screen resolutions, how can a programmer determine how big to make the frame?
 - Determine the screen resolution
 - Obtain system-information, such as screen resolution, using a `Toolkit` object
 - `Toolkit` has `getScreenSize()` that returns the screen resolution as a `Dimension` class object
 - `Toolkit`, `Dimension`: part of `java.awt` package

Nov 3, 2008

Sprenkle - CS209

26

Screen Resolution

- `Dimension` object holds a width and height value, in pixels
 - public instance fields

```

Toolkit kit = Toolkit.getDefaultToolkit();
Dimension screenSize = kit.getScreenSize();
int screenWidth = screenSize.width;
int screenHeight = screenSize.height;
  
```

Nov 3, 2008

Sprenkle - CS209

27

Example

What will this Frame look like?

```

class CenteredFrame extends JFrame {

    public CenteredFrame() {
        Toolkit kit = Toolkit.getDefaultToolkit();
        Dimension screenSize = kit.getScreenSize();
        int screenHeight = screenSize.height;
        int screenWidth = screenSize.width;

        setSize(screenWidth / 2, screenHeight / 2);
        setLocation(screenWidth / 4, screenHeight / 4);

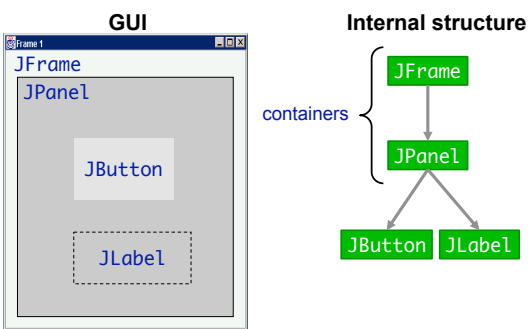
        setTitle("My Centered Frame");
    }
}
  
```

Nov 3, 2008

Sprenkle - CS209

28

Anatomy of an Application GUI



Nov 3, 2008

Sprenkle - CS209

29

Using a GUI Component

1. Create it
2. Configure it
3. Add children (if container)
4. Add to parent (if not `JFrame`)
5. Listen to it



Nov 3, 2008

Sprenkle - CS209

30