

## Objectives

- Programming Paradigms
- Introduction to GUIs in Java

Nov 4, 2011

Sprenkle - CSCI209

1

## Review

- What is the Liskov Substitution Principle?

Nov 4, 2011

Sprenkle - CSCI209

2

## Assignment 10 Notes

- What was the code smell we identified?
- How are we going to make the game more extensible?
- Focus on Extensibility
- But, handle other code smells as well
- Due next Wednesday

Nov 4, 2011

Sprenkle - CS209

3

## PROGRAMMING PARADIGMS

Nov 4, 2011

Sprenkle - CSCI209

4

## Programming Paradigms

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

Blurred lines  
between paradigms,  
Not completely  
independent

Nov 4, 2011

Sprenkle - CSCI209

5

## GUIs IN JAVA

Nov 4, 2011

Sprenkle - CSCI209

6

## Java GUI Libraries: AWT & Swing

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

Nov 4, 2011

Sprenkle - CSCI209

7

## Swing & AWT

- Swing does not completely replace AWT
- Using the Swing graphics programming model
  - Improves performance
  - Allows more efficient development of GUIs
- We will use Swing mostly
  - Leverage AWT

Nov 4, 2011

Sprenkle - CSCI209

8

## Swing: Made up of Components

- Top-level components
  - ~Hold GUI elements
  - Examples: `JFrame`, `JWindow`, `JDialog`, `JApplet`
- GUI Elements
  - ~Things user interacts with
  - Examples: `JButton`, `JLabel`, `JMenuBar`

Nov 4, 2011

Sprenkle - CSCI209

9

## JFRAMES AND PARENT CLASSES

Nov 4, 2011

Sprenkle - CSCI209

10

## Frames

- **Frame**: Most basic unit of graphics programming
- Example of a *container*
  - A *container* contains other UI components
- A top-level window
  - Not contained within another window
- Swing's `JFrame` class implements a frame



Nov 4, 2011

Sprenkle - CSCI209

11

## 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);
    }
}
```

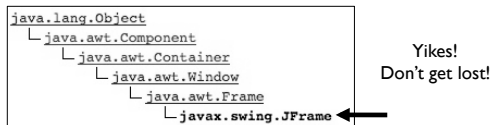
Nov 4, 2011

Sprenkle - CSCI209

12

## Frame Inheritance

### • Class hierarchy



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

Nov 4, 2011

Sprenkle - CSCI209

13

## Components & Containers

### • Component

- **Abstract** class
- Everything you [see](#) is a component
  - All nonmenu-related AWT components
- Many methods
  - Some deprecated: be careful

### • Container

- **Concrete** implementation of Component
- Base class of many classes



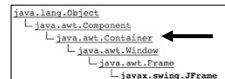
Nov 4, 2011

Sprenkle - CSCI209

14

## Container Methods

- `add(Component c)`
- `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`



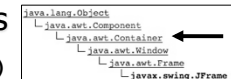
Nov 4, 2011

Sprenkle - CSCI209

15

## Container Methods

- `remove(Component c)`
- `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



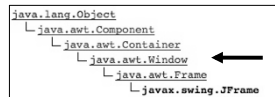
Nov 4, 2011

Sprenkle - CSCI209

16

## 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



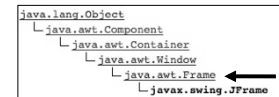
Nov 4, 2011

Sprenkle - CSCI209

17

## Frame's Methods

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



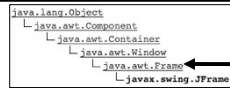
Nov 4, 2011

Sprenkle - CSCI209

18

## Frame Methods

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



Nov 4, 2011

Sprenkle - CSCI209

19

## Screen Resolution

- Screens have various resolutions
- How do you determine how big to make a frame?
  - Determine the screen resolution
  - Obtain system information, such as screen resolution, using a `Toolkit` object
    - `Toolkit's getScreenSize()`
      - Returns screen resolution as a `Dimension` object
  - `Toolkit`, `Dimension`: part of `java.awt` package

Nov 4, 2011

Sprenkle - CSCI209

20

## Screen Resolution

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

```

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

Nov 4, 2011

Sprenkle - CSCI209

21

## Example

What will this Frame look like?

```

class MyFrame extends JFrame {
    public MyFrame() {
        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);

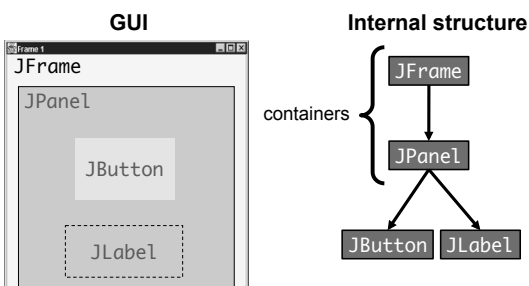
        setTitle("My Frame Title");
    }
}
  
```

Nov 4, 2011

Sprenkle - CSCI209

22

## Anatomy of an Application GUI



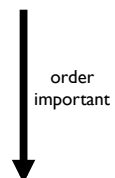
Nov 4, 2011

Sprenkle - CSCI209

23

## Implementing 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 4, 2011

Sprenkle - CSCI209

24

## Implementing a GUI Component

1. Create it  
 `JButton b = new JButton();`
2. Configure it  
 `b.setText("press me");`  
 `b.setForeground(Color.blue);`
3. Add it to parent  
 `panel.add(b);`
4. Listen to it  
➤ Events: Listeners

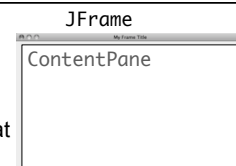
Nov 4, 2011

Sprenkle - CSCI209

25

## JFrame

- Contains `ContentPane`
  - A `Container` object that holds components you add, placing them in the frame
  - The part of the frame that holds UI components



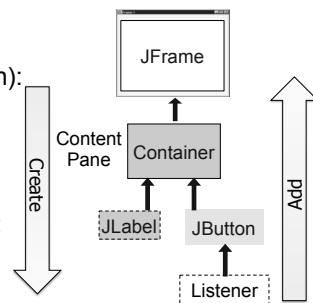
Nov 4, 2011

Sprenkle - CSCI209

26

## Building a GUI

1. Create (top down):
  - Frame
  - Container
  - Components
  - Listeners
2. Add (bottom up):
  - Listeners into components
  - Components into panel
  - Panel into frame



Nov 4, 2011

Sprenkle - CSCI209

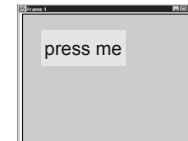
27

## Example Code

```
// create the components
JFrame f = new JFrame("title");
Container pane = f.getContentPane();
JButton b = new JButton("press me");

// add button to panel
pane.add(b);

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



Nov 4, 2011

Sprenkle - CSCI209

28

## DRAWING

Nov 4, 2011

Sprenkle - CSCI209

29

## JPanel

- Implements a panel
- A panel has a surface on which you can draw
- A panel is a `Container`
  - Can add components to a panel
- Useful in designing layouts

Nov 4, 2011

Sprenkle - CSCI209

30

## To Draw on a Panel

- Define a new class that *extends* JPanel
- In derived class, override paintComponent (Graphics g)
- Graphics object
  - Collection of settings for drawing images and text, e.g., colors and fonts
  - Abstract class
    - Implementation different for each platform
  - All drawing in Java goes through a Graphics object

Nov 4, 2011

Sprenkle - CSCI209

31

## Drawing on a Panel

```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        // code for drawing goes here
    }
}
```

Nov 4, 2011

Sprenkle - CSCI209

32

## paintComponent

- System calls paintComponent *automatically* whenever container needs to be redrawn
  - Do *not* call this method yourself
  - It will be called when it needs to be
- If need to force repainting the screen, call repaint()
  - Calls paintComponent for all needed components with appropriate Graphics objects

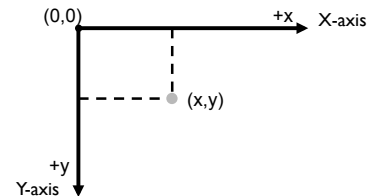
Nov 4, 2011

Sprenkle - CSCI209

33

## Drawing on a Panel: Graphics

- Measurements on a Graphics object is in pixels, as an offset from the top-left corner
  - (0,0) coordinates represent top-left corner of the container on which you are drawing



Nov 4, 2011

Sprenkle - CSCI209

34

## Rendering Text

- Displaying text is a special type of drawing, called *rendering text*
- To render text on a panel, call drawString()

```
class HelloWorldPanel extends JPanel {
    public static final int MESSAGE_X = 75;
    public static final int MESSAGE_Y = 100;

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

        g.drawString("Hello World.",
            MESSAGE_X, MESSAGE_Y);
    }
}
```

Nov 4, 2011

Sprenkle - CSCI209

35

## Drawing on a Panel

- Notice we call superclass's (JPanel) paintComponent method
- JPanel has its own idea on how to draw/ paint the panel
  - Fills in the background color
- To make sure background color gets filled, call superclass's paintComponent
  - Every JPanel should color its background

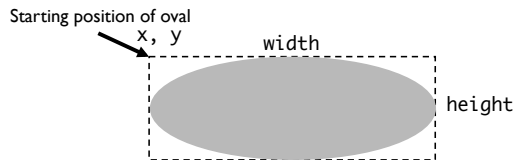
Nov 4, 2011

Sprenkle - CSCI209

36

## Drawing Lines, Rectangles, Ovals

- Draw ovals, rounded rectangles within bounding rectangle



- Filled or outlined (e.g., `fillRect` vs `drawRect`)
- Can also draw arcs, polygons, polylines

Nov 4, 2011

Sprenkle - CSCI209

37

## Colors

- Colors made up of three components
  - Red, Green, Blue component
  - RGB values
    - Components: either 0 to 255 or 0.0 to 1.0
- `Color` class defines 13 color constants
  - black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, and yellow
  - Also defined in all caps
  - See API

[http://en.wikipedia.org/wiki/List\\_of\\_colors](http://en.wikipedia.org/wiki/List_of_colors)

Nov 4, 2011

## Using Graphics object

1. Set the color/font
2. Draw the string/shape

Nov 4, 2011

Sprenkle - CSCI209

39

## Different Implementation Approach

```
public static void main(String args[]) {
    JFrame frame = new JFrame("Using colors");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    ColorJPanel colorJPanel = new ColorJPanel();
    frame.add(colorJPanel);

    frame.setSize(500, 180);
    frame.setVisible(true);
}
```

Extends JPanel

ColorJPanel.java

Nov 4, 2011

Sprenkle - CSCI209

40

## FONTS

Nov 4, 2011

Sprenkle - CSCI209

41

## Changing the Text Font

- Previous `drawString` code drew text using default system font
- Can change the font
- Need to determine which fonts are installed on machine running the program

Nov 4, 2011

Sprenkle - CSCI209

42

## Determining Available Fonts

- **GraphicsEnvironment**
  - Represents the system's graphical environment
  - Call `getAvailableFontFamilyNames()`
    - Returns an array of Strings
    - Each String is the name of a font installed on the system
- Your program can look through fonts to see if font(s) it wants is available on system
- Five fonts are always available, mapped to some font on machine
  - SansSerif, Serif, Monospaced, Dialog, DialogInput

Nov 4, 2011

Sprenkle - CSCI209

43

## Determining the Available Fonts

- To list all fonts installed on system:

```
import java.awt.*;

public class ListFonts {

    public static void main(String[] args) {
        String[] fontNames = GraphicsEnvironment
            .getLocalGraphicsEnvironment()
            .getAvailableFontFamilyNames();
        for (int i=0; i < fontNames.length; i++)
            System.out.println(fontNames[i]);
    }
}
```

Nov 4, 2011

Sprenkle - CSCI209

44

## Creating a Font Object

- **Font** object represents font on the system
- **Font** constructor takes 3 arguments:
  - a String with the font name
  - a constant (defined in the `Font` class) that describes the font style (plain, **bold**, *italic*, or **bold italic**)
  - an integer for the point size

Nov 4, 2011

Sprenkle - CSCI209

45

## Creating a Font Object

```
Font sansbold14 = new Font("SansSerif", Font.BOLD, 14);
Font helvi12 = new Font("Helvetica", Font.ITALIC, 12);
```

- You can change the font that the `Graphics` object uses by calling `setFont()`
- For example...

```
Font sansbold14 = new Font("SansSerif", Font.BOLD, 14);
g.setFont(sansbold14);
g.drawString("Hello, there in SansSerif.", 75, 100);
```

Nov 4, 2011

Sprenkle - CSCI209

Game.java

46

## More GUI Components

- **Choice**
  - Drop-down list
- **FileDialog**
  - Opening and saving files
- **List**
  - Scrollable
  - Allows multiple selections
- **ScrollPane**
  - scrollbars
- **TextField**
  - Single line of text
- **TextArea**
  - Multiple lines of text

Nov 4, 2011

Sprenkle - CS209

47

## Menus

- **MenuBar**
  - Thing across top of frame
  - Frame: `setMenuBar(MenuBar mb);`
- **Menu**
  - The dropdown part
  - A sequence of `MenuItems`
  - `Menu` is a subclass of `MenuItems`, so can have submenus

Nov 4, 2011

Sprenkle - CS209

48



## Practice: Combining Components

- Create a panel with three buttons on it

ButtonPanel.java

Nov 4, 2011

Sprenkle - CS209

49

## Placement of Components

- How does the panel know where to place a button?
- How does the panel know where to place the next button?
- How does the panel know where to place *any* component that is added to it?

Nov 4, 2011

Sprenkle - CS209

50

## Looking Ahead

- Today: 12:30 p.m., Women's Resource Room
  - Professor Stough's talk
  - In Sakai:
    - 3 most important points of his talk
    - most surprising thing he mentioned
    - at least one question that you wondered during the talk
    - discuss at least one CS topic/technology that this work provides a foundation for (beyond those mentioned in the talk) or one use that you see possible
- Next Wednesday: Roulette Refactoring due
- Next Friday: 2<sup>nd</sup> Exam
  - Focus: Python vs. Java, collections, testing, coverage, design principles (tradeoffs), metrics, GUIs
  - Terminology

Nov 4, 2011

Sprenkle - CSCI209

51