Beginning GUI Programming: More Adventures in Inheritance and Aggregation (and packages)

Slides derived from the work of
Dr. Amy McGovern and Dr. Deborah Trytten

Administrivia

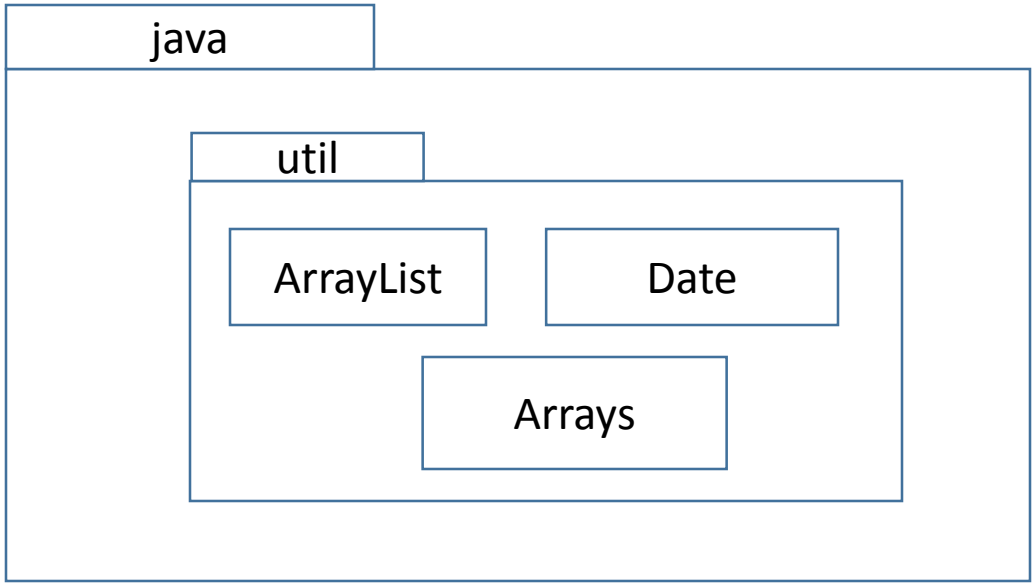
- Today is the deadline for project 2 code review arrangements
- Project 3
- Lab 9 is coming soon

Packages

- Packages are used to group classes together into meaningful units
- Examples
  - java.util
  - java.io
- Very simple to create a package
  - When creating a new class, enter the package name in the dialog box
  - Eclipse adds to top of file:
    - package packageName;

Packages in UML

Packages can nest



Package Naming Conventions

- Top level domain of organization first
  - Subdomains in reverse order
  - Then, name of package
- Done to preserve uniqueness of package names
- Example:
  - edu.ou.game

Access Modification and Packages

- Protected data is shared when classes share a package
- If you do not give methods and data an access modifier (public, protected, private) it will have package level access

Access Levels

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

Handy dandy chart:
http://download.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

Basic Elements of GUIs

- **Components** (sometimes called widgets)
  - Buttons, Checkboxes, etc.
- **Containers** collect Components together into a single unit

Basic Elements of GUIs II

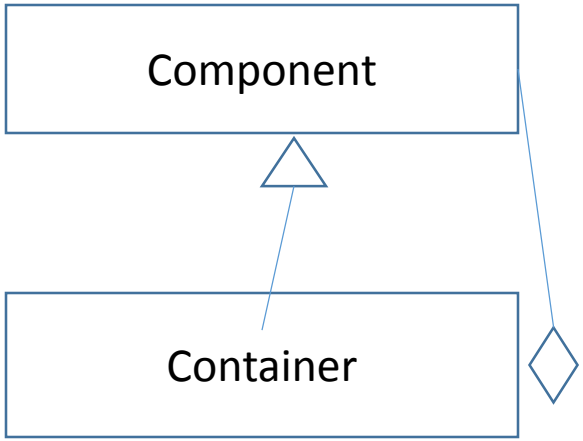
- **Layout Managers**
  - Determine how the components are organized on a display
- Look for examples in PowerPoint
  - Resizing can help identify layout management

Two Kinds of Components

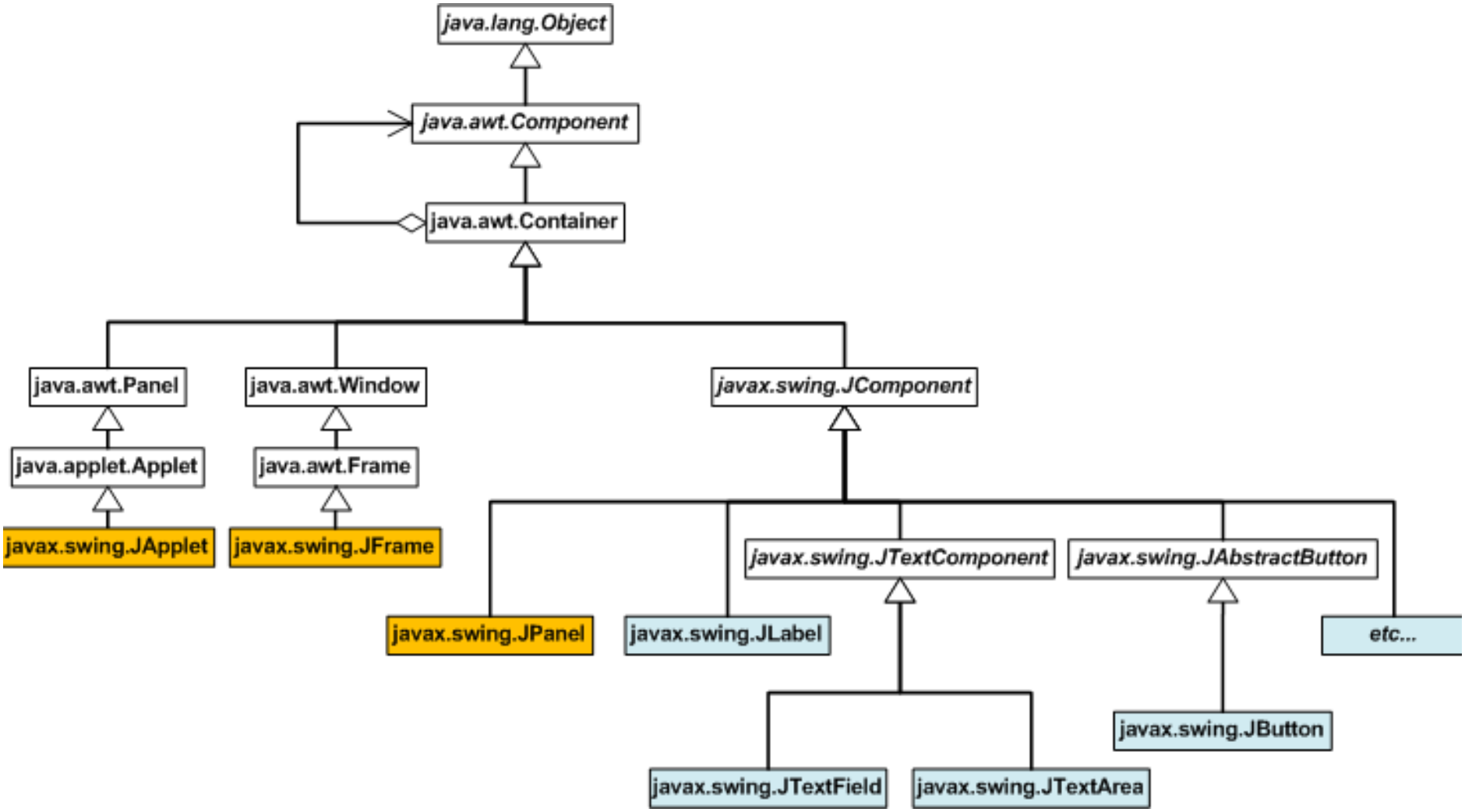
- Abstract Windowing Toolkit (java.awt)
  - Designed to use components from platform (OS/window manager)
  - Good idea, but was problematic
    - Have to allow options for all platform options
    - Resulted in heavyweight components
- Swing (javax.swing)
  - Components start with J (e.g. JButton)
  - Only lowest level components are from platform

Container and Component …

- Container inherits from Component
  - Container is-a Component
- Container also aggregates Component



Java GUI Hierarchy UML



https://www.clear.rice.edu/comp310/JavaResources/GUI/

How to create a window

Key window pieces:

```
JFrame frame = new JFrame("Title");

// Default is to do nothing, so this is necessary
frame.setDefaultCloseOperation(
                JFrame.EXIT_ON_CLOSE);

// Default is upper left corner of screen
frame.setLocationRelativeTo(null); // centers window

// Default is size zero so only decorations will be shown
frame.setSize(100, 100);

// Won't see anything if you forget this
frame.setVisible(true);
```

Example

- Make a window titled “Foo Window”

Add Stuff (Components) to the Window

Frames have a content pane that holds components

- JButton ok = new JButton(“OK”);
  - Construct new button
- frame.add(ok);
  - Put it in the frame’s content pane
- What happens to the button when the window is resized?
- What happens if we try to put in a second button?
- Add a tooltip
  - componentName.setToolTipText(“Explanation”)

Example

- Make a window titled “Save Window”
- Add a button that says “Save first half”
- Can we add a button that says “Save second half”?

Another Implementation Option

Allow class to inherit from JFrame directly, rather than having a JFrame as a component

- What does the code look like?
- Constructor creates the components and containers and hooks them together … rather than doing this in main()

Additional Components

- JLabel – labels on the screen
- Fields for users to enter text
  - JTextField: single line
  - JTextArea: multiple lines
- Binary choices
  - JRadioButton
  - JCheckBox
- Lots of other choices (see classes that start with J in API)

Layout Managers

- The configuration of components within a content pane is determined by the Layout Manager
  - Layout managers are in the java.awt package
- Layout Management is rarely done with absolute pixel positions
  - Too many different monitors, platforms, window sizes

Layout Managers

To change layout managers

- Construct the new manager (e.g., use FlowLayout)
- Use setLayout() to pass the newly constructed manager to the container

FlowLayout

Components are placed in the container from left to right, top to bottom

- Order of calls to add() determines placement
- Resizing rarely works well

Administriva

- Project 3 due Friday
  - Code review slots start next week
- Project 4 coming soon

GridLayout

- Used for rectangular arrangements of widgets
- All components are equal size (limits use of layout)
- Components added left to right, top to bottom
  - These could be stored in an array
- Order of calls to add determines placement

GridLayout

- GridLayout constructor takes two parameters: number of rows and columns
- Weirdness with size = 0 for rows/cols:
  - Used to flag that the other dimension should dominate and this one should be as small as possible.
- More complicated layout manager available: GridBagLayout

BorderLayout

- Really nice layout manager
- Divides screen into five areas (constants)
  - Four compass directions
  - Center
  - Some areas can be left empty
- Resizing occurs at center
  - PowerPoint does this
- Adding: add(component, placement)
  - What happens when multiple components are added to the same placement?

Containers Revisited

- To add more than one component to the middle of a BorderLayout need another container
  - JPanel used to organize components
- Common for hierarchies to get deep
  - Necessitates careful design and documentation of components and containers
  - Resizing is cleaner
- Can add borders to JPanels:
  - TitledBorder (line around JPanel containing description)
  - LineBorder(line around JPanel)

Events

Set up: We register a Listener with some Component

- addActionListener(new ActionListener(){…})

Events

When the component decides that the important event has happened:

- Component creates an Event object to describe what happened
  - ActionEvent
- Component then calls the event handler in the Listener:
  - actionPerformed(ActionEvent e)

Color

- In java.awt
  - RBG: 0-255
  - Doesn't work like grade school paints
    - red + green = yellow
- Can change background and foreground colors
  - setBackground()
  - setForeground()
- Lots of standard name constants available
  - BLACK, BLUE, CYAN, etc.
- Remember: artists are your friends

Font

- Problematic in commercial systems
  - Can be proprietary
  - Can be system dependent
- In java.awt
  - Names of families of font (mapped to fonts by local installation)
    - Serif (has little projections)
    - SanSerif (no little projections)
    - Monospaced (every character takes same space—can be UGLY)
    - Dialog (default)
    - DialogInput
  - Styles
    - PLAIN, BOLD, ITALIC
  - Size (in pixels)

ImageIcon

- Small image of fixed size
  - ImageIcon icon = new ImageIcon(filename);
  - Doesn't throw FileNotFoundException! Why?
- Formats
  - GIF
  - JPEG
  - PNG
- Can be placed on JLabel or JIcon by constructor
- Icons can be shared with multiple components

GUI Design

- Everyone thinks they can design GUIs perfectly
- Good GUI design requires
  - Artistic skill
    - Taste
    - Color sense
  - Usability considerations
    - Number of visible controls
    - Modes
    - Organization
    - Visibility
- CS 3053 Human Computer Interaction