CMSC 132: Object-Oriented Programming II

Graphical User Interface (GUI)

Department of Computer Science
University of Maryland, College Park
Graphical User Interface (GUI)

- **User interface**
  - Interface between user and computer
  - Both input and output
  - Affects *usability* of computer

- **Interface improving with better hardware**
  - Switches & light bulbs
  - Punch cards & teletype (typewriter)
  - Keyboard & black/white monitor (text)
  - Mouse & color monitor (graphics)
Model-View-Controller (MVC)

- Model for GUI programming (Xerox PARC ’78)
- Separates GUI into 3 components
  1. Model ⇒ application data
  2. View ⇒ visual interface
  3. Controller ⇒ user interaction
MVC Model of GUI Design

- **Model**
  - Should perform actual work
  - Should be independent of the GUI
    - But can provide access methods

- **Controller**
  - Lets user **control** what work the program is doing
  - Design of controller depends on model

- **View**
  - Lets user see what the program is doing
  - Should not display what controller **thinks** is happening (base display on model, not controller)
Programming Models

Normal (control flow-based) Programming

- **Approach**
  - Start at main()
  - Continue until end of program or exit()

Event-driven Programming

- **Event - Action** or condition occurring outside normal flow of control of program (e.g., mouse clicks, keyboard input, etc.)
- Unable to predict time & occurrence of event

- **Approach**
  - Start with main()
  - Define system elements and register event listeners
  - Await events (& perform associated computation)
Event Handling in Action

Events

Can handle an event of type $e_1$

Execution Environment

Registered Event Handlers

$E_1$, $E_4$, $E_5$, $E_6$, $E_7$, $E_8$
GUIs are Event-Driven Software

User events invoke event handlers

User Events

Event Handlers

changeFontSizeActionPerformed (java.awt.event.ActionEvent evt)
newDocActionPerformed (java.awt.event.ActionEvent evt)
fileSaveActionPerformed (java.awt.event.ActionEvent evt)
Desktop Java Graphics APIs: From “Filthy Rich Clients”
by Chet Haase and Romain Guy, Chap1, Page 12
ISBN-978-0-13-241393-0
Book Web Site: http://www.filthyrichclients.org/
GUIs in Java

- AWT (Abstract Window Toolkit) (java.awt.*)
  - First graphical user interface toolkit for Java
  - Old GUI framework for Java (Java 1.1)
  - Reliance on native system libraries
  - Platform independence problems
  - Responsible for input event mechanisms

- Java 2D
  - Graphics Library of Java
  - Introduced in JDK 1.2
  - Basics and advance drawing operation, image manipulation, and drawing
  - Handles Swing’s Rendering operations

- Swing (javax.swing.*)
  - GUI framework first introduced in JDK 1.2
  - Includes AWT features plus many enhancements
  - Pure Java components (no reliance on native code)
  - Pluggable look and feel architecture

Some of this material is from “Filthy Rich Clients” (see reference in previous slide).
Steps for Creating a GUI in Java

1. Define a **container** to hold components
   - Examples: JFrame, JApplet…

2. Add GUI **components** to the container
   - Examples: JButton, JTextField, JScrollBar…
   - Use layout manager to determine positions

3. Add actions to GUI
   - Add event listeners to GUI components

4. Schedule the GUI processing in the EDT (Event-Dispatching Thread)
Step 1 (Define Container)

Container Definition
- Abstractions occupying space in GUI

Properties
- Usually contain one or more widgets
  - widget - actual item user can see
- Can be nested in other containers

Container Examples
- JFrame, JDialog, JPanel, JScrollPane
Step 2 (Define Components)

**Component Definition**
- Actual items (widgets) user sees in GUI

**Examples**
- Labels (fixed text)
- Text areas (for entering text)
- Buttons
- Checkboxes
- Tables
- Menus
- Toolbars
- Etc…
Step 3 (Set Event Listeners)

Implementation
- Implement event listeners for each event
- Usually one event listener class per widget
- Inner class usually utilized to implement listener
- Register (add) listener object with widget object

At run time
- Java generates event object when events occur
- Java then passes event object to event listener

Example of Java listeners & Actions Causing Event
- ActionListener → clicking button in GUI
- CaretListener → selecting portion of text in GUI
- FocusListener → component gains / loses focus
- KeyListener → pressing key
- MouseListener → mouse clicked
- WindowListener → closing a window
Step 4 (Schedule GUI Processing in EDT)

- What is a thread?
- Event Dispatching Thread (EDT)
  - EDT is a background thread to process events
  - These events are mainly updates that
    - Cause components to redraw themselves
    - Represent input events
- Swing uses a single-threaded painting model
  - Event Dispatching thread is the only valid thread for updating GUI components
  - Avoid updating GUI components from other threads
    - A source of common bugs
Event Dispatching Thread

Code that allows current thread to execute GUI code in dispatching thread

```java
public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            createAndDisplayGUI(); // actually creates GUI
        }
    });
}
```
Additional Resources

- Javadoc from the JDK
  
- Swing tutorial -
  http://java.sun.com/docs/books/tutorials/uiswing/components/

- Filthy Rich Clients
  http://filthyrichclients.org/