

# CSE1030 – Introduction to Computer Science II

## Lecture #12


### Graphical User Interfaces I

## Goals for Today

- Introduction to:  
Graphical User Interface Programming
- Three Challenges:
  1. Java Swing is a HUGE API
  2. Have to be very comfortable with OOP
    - Extending classes, implementing Interfaces
  3. Event-Driven programming instead of Sequential

CSE1030 2

## CSE1030 – Lecture #12

- Introduction 
- Java GUI Programming
- Sequential versus Event-Driven
- We're Done!

CSE1030 3

## Let's look at these 2 programs...

`DemoLargestConsole.java`

`DemoLargestGUI.java`

CSE1030 4

## What Differences Did We See?

- What Interaction technologies were used for input and output?
  - Console → keyboard / command-line
  - GUI → keyboard & mouse / graphical display
- Who Drove the Interaction?
  - Console → the Program
  - GUI → the User

CSE1030 5

## Textual (Console) Interfaces

- Older Interaction Style
- Provides a means to express commands to a computer directly via typing and reading text
- May use function keys, single characters, abbreviations, or whole-word commands
- Primarily used today for older applications (e.g., ftp, telnet, Unix command-line)
- Can be difficult for Novices
- Often preferred by Expert users

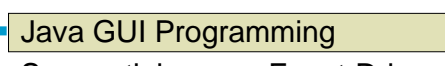
CSE1030 6

## Graphical User Interfaces

- Newer Style of Interaction
- Usually involves a Pointing Device and Graphical Display
- Richer Output (Graphics, Sound, Video)
- Several Variations
  - Point & Click (web pages)
  - Question & Answer (MS Windows "Wizards")
  - Forms (Data Entry, Spread Sheets)
  - WIMP (Windows, Icons, Menus, Pointers)
- Can be easier for Novices
- May not be preferred by Experts

CSE1030 7

## CSE1030 – Lecture #12

- Introduction
- **Java GUI Programming** 
- Sequential versus Event-Driven
- We're Done!

CSE1030 8

## GUI Programming with Java

- GUI Programming is accomplished with the javax.swing package
- Sun's Swing toolkit is Java's most advanced toolkit, and largest API
- Before Swing...
  - AWT (abstract windowing toolkit)
  - Most of AWT is now obsolete...
  - but AWT still used for a few things (drawing, images, etc.)
- Swing still uses many features of AWT

CSE1030 9

## GUI Program Organization

- How do you code a GUI program?

DemoHelloWorld.java  
DemoHelloWorld2.java  
DemoSwing.java

CSE1030 10

```
import ...  
  
public class NameOfProgram  
    extends JFrame  
    implements ActionListener  
{  
    public static void main(String[] args)  
    {  
        ...  
    }  
}
```

Identify packages containing classes used in the program

Java Swing (GUI) Library is HUGE. Extend and implement!

1. Construct the GUI frame
2. Give it a title
3. Show it
4. Done!

All the work is done here

CSE1030 11

```
import ...  
  
public class NameOfProgram  
    extends JFrame  
    implements ActionListener  
{  
    public static void main(String[] args)  
    {  
        ...  
    }  
}
```

Our GUI program is actually a JFrame extended and modified to suit our needs

CSE1030 12

## JFrame

- Java GUI Programs are instances of JFrame
  - JFrame is extended to make our own class
- Interaction is received through listeners
  - Listeners are implemented interfaces
  - There are listeners for many different kinds of input (keyboard, mouse, windows opening or closing, and many more)
- So we must be comfortable with extending classes and implementing interfaces

CSE1030 13

## JPanel

- JPanels contain GUI elements
  - This is composition
- The JPanel uses a Layout Manager to arrange the display (to layout the GUI components)
- GUI elements are most of the things you can see or interact with in a graphical program
  - JLabel, JButtons, JTextfield, images, etc.
  - Does not contain application-wide elements, like JMenu, JPopup, etc.

CSE1030 14

## JFrame Constructor

- Must...
  - Create and configure the GUI components
  - Install (“add”) listeners
    - Listeners are not just installed, they must be associated with particular GUI components
  - Arrange components in panel(s)
  - Get the JFrame’s content pane
  - Add panel(s) to content pane

CSE1030 15

## Also, Notice the “Inner Class”

- A Java Class can define a class within a class

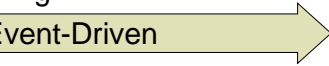
```
public class DemoSwing extends JFrame implements ActionListener
{
    ...

    public DemoSwing() { ... }

    // Note: WindowAdapter implements WindowListener
    private class WindowCloser extends WindowAdapter
    {
        public void windowClosing(WindowEvent event)
        {
            System.exit(0);
        }
    }
}
```

CSE1030 16

## CSE1030 – Lecture #12

- Introduction
- Java GUI Programming
- Sequential versus Event-Driven 
- We're Done!

CSE1030 17

## Some Example Code

- Sequential programs have linear execution
  - Step 1, Step 2, Step 3
  - `DemoLargestConsole.java`
- Event-Driven programs are chaotic
  - Whatever we are asked to do next, we have to be able to handle
  - `DemoLargestGUI.java`

CSE1030 18

## Sequential Programming

- In sequential programs, the program is under control
- The user is required to synchronize with the program:
  - Program tells user it's ready for more input
  - User enters more input and it is processed
- Examples:
  - Command-line prompts (DOS, UNIX)
  - Command-line programs (ftp, telnet)

CSE1030 19

## Sequential Programming (2)

- Flow of a typical sequential program:
  1. Prompt the user (Output)
  2. Read input from the keyboard (Input)
  3. Parse the input (Process...)
  4. Evaluate the result
  5. Generate output (Output)
  6. Repeat

CSE1030 20

## Sequential Programming (3)

- Advantages
  - Architecture is sequential (one step at a time)
  - Easy to model (flowcharts, state machines)
  - Easy to build
- Limitations
  - Hard to implement complex interactions
  - Interaction must proceed according to a pre-defined sequence
- To the rescue... Event-driven programming

CSE1030 21

## Event-driven Programming

- Instead of a user synchronizing with the program, the program synchronizes with, or reacts to, the user
- All communication from user to computer occurs via *events* and the code that handles the events
- An event is an action that happens in the system, such as:
  - A mouse button pressed or released
  - A key-press on the keyboard
  - A window is moved, resized, closed, etc.

CSE1030 22

## Classes of Events

- Typically two different classes of events:
  - User-initiated events
    - Events that result directly from a user action (e.g., mouse click, move mouse, key press)
  - System-initiated events
    - Events created by the system, as it responds to user action (e.g., scrolling text, re-drawing a window)
- Both classes of events need to be processed
- User-initiated events may generate system-generated events

CSE1030 23

## DemoLargestGUI

- The user requests that the program performs an action (finds the largest value) by clicking on the "Find Largest" button
- The button functionality is provided by a JButton object, which "fires an Action Event" when it is clicked
- By registering as an ActionListener for this button, we can intercept clicks (be informed when the user clicks the button)

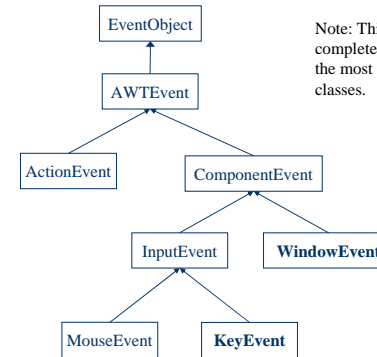
CSE1030 24

## Java Events

- When a user types characters or uses the mouse, Java's window manager sends a notification to the program that an event has occurred
  - E.g., when the user presses a key on the keyboard, a key pressed event occurs
- There are many, many kinds of events actionevents (button clicks), key events, mouse events, etc.
- Many are of no interest
- Some are of great interest

CSE1030 25

## Java's Event Class Hierarchy



Note: This diagram is not complete. It just shows only the most common event classes.

CSE1030 26

## Let's look at DemoKeyEvents

`DemoKeyEvents.java`

CSE1030 27

## Processing Events

- Signature for the keyPressed method is:  
`public void keyPressed(KeyEvent ke)`
- When our keyPressed method executes, it receives a KeyEvent object as an argument
- We use the KeyEvent object to
  - Determine which key was pressed, using
    - `getKeyChar`, `getKeyCode`, etc.
  - Determine the source of the event, using `getSource`
    - This is important if there is more than one component registered to receive key events (not the case in our example program).

CSE1030 28

## Java Events (2)

- To receive notification of events of interest, a program must install event listener objects
- It is not enough to simply know that an event has occurred; we need to know the event source
  - E.g., a key was pressed, but in which of several text fields in the GUI was the key pressed?
- So, an event listener must be installed for particular components that may generate the event
- Let's look at the code. First, the big picture...

CSE1030 29

## Event Sources

- Java's event classes are all subclasses of EventObject (see earlier slide)
- EventObject includes the getSource method:

```
public Object getSource()
```
- Didn't need this in our example program, because only one object (enterField) was registered to generate key events
- So, when the keyPressed method executes we know it is because a key was pressed in enterField
- But, let's say we have two JTextField components: (next slide)

CSE1030 30

## Let's look at DemoKeyEvents2

**DemoKeyEvents2.java**

CSE1030 31

## Listeners

- Java's listener classes are interfaces
- Reminder: Interfaces...
  - Contain only the design of a class
  - Do not have instance variables
  - Include only abstract methods
  - Include only public methods
  - Possibly: static final Data

CSE1030 32



## Listeners (2)

- The signature of our extended JFrame class includes the clause implements KeyListener
- This means our class must include definitions for the methods of the KeyListener listener

```
public void keyPressed(KeyEvent ke) {}
public void keyReleased(KeyEvent ke) {}
public void keyTyped(KeyEvent ke) {}
```

- Our implementation includes the code we want executed when a key is pressed, released, and/or

33

## Installing Listeners

- It is not enough simply to implement the methods of a listener
- The listener must also be installed (or “added”)
- Furthermore, it must be installed for the component to which the listener methods are to be associated
- Thus (from our example program)

```
enterField.addKeyListener(this);
```

Component to which  
the listener methods are  
to be associated

An object of a class  
that implements the  
listener methods

CSE1030 34

## Installing Listeners (2)

- Signature for the addKeyListener method:

```
public void addKeyListener(KeyListener)
```
- Description:
  - Adds the specified key listener to receive key events from this component.
- In our example, we used this as the “specified key listener”
  - Indeed, the current instance of our extended JFrame class (“this”) is a key listener because it implements the key listener methods
- Result: when a key-press event occurs on the enterField component, the keyPressed method in our extended JFrame class will execute!

CSE1030 35

## Let’s look at DemoMouseEvents

**DemoMouseEvents.java**

CSE1030 36

## Adapters and Inner Classes

- Let's Revisit:

**DemoSwing.java**

CSE1030 37

## Back to the Example Program...

```
...
public class NameOfProgramFrame extends JFrame
implements KeyListener
{
    ...

    private class WindowCloser extends WindowAdapter
    {
        public void windowClosing(WindowEvent we)
        {
            System.exit(0);
        }
    }
}
```

CSE1030 38

## Adapter Classes

- What is an adapter class?
  - A class provided as a convenience in the Java API
  - An adapter class includes an empty implementation of the methods in a listener
  - Programmers extend the adapter class and implement the methods of interest, while ignoring methods of no interest

CSE1030 39

## WindowAdapter

```
public abstract class WindowAdapter
implements WindowListener
{
    void windowActivated(WindowEvent we) {}
    void windowClosed(WindowEvent we) {}
    void windowClosing(WindowEvent we) {}
    void windowDeactivated(WindowEvent we) {}
    void windowDeiconified(WindowEvent we) {}
    void windowIconified(WindowEvent we) {}
    void windowOpened(WindowEvent we) {}
}
```

CSE1030 40

## Using the WindowAdapter Class

- Define an inner class that extends the WindowAdapter class
  - Implement the listener methods of interest
  - Ignore other listener methods
- In the frame constructor, use the appropriate “add” method to add an object of the extended WindowAdapter class

- In our example program...

```
this.addWindowListener(new WindowCloser());
```

CSE1030 41

## Examples of Listeners and Adapters

Listeners (# methods)	Adapters
KeyListener (3)	KeyAdapter
WindowListener (7)	WindowAdapter
MouseListener (5)	MouseAdapter
MouseMotionListener (2)	MouseMotionAdapter
MouseListener (7)	MouseListenerAdapter
ActionListener (1)	-
ItemListener (1)	-
FocusListener (2)	FocusAdapter

(Note: MouseInputListener combines MouseListener and MouseMotionListener)

CSE1030 42

## Extending Adapters vs. Implementing Listeners

- Largely a matter personal choice
- Our example program does both
  - The KeyListener methods were implemented
  - The WindowAdapter class was extended
- Could have done the opposite, i.e.,
  - Extend the KeyAdapter class
  - Implement the WindowListener methods
- Note: a Java class can implement many listeners, but it can extend only one class
  - Java does not include multiple inheritance

CSE1030 43

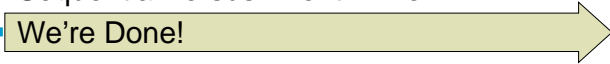
## Pros and Cons

- Using adapter classes
  - Advantage
    - Only the listener methods needed are defined
  - Disadvantage
    - A bit complicated to setup
    - Need to define an inner class, then instantiate an object of the inner class to pass to the appropriate “add” method
- Implementing listener methods
  - Advantage
    - A class can implement many different listener interfaces
  - Disadvantage
    - Must implement all the methods defined in the listener (even those not used)

CSE1030 44

## CSE1030 – Lecture #12

- Introduction
- Java GUI Programming
- Sequential versus Event-Driven
- We're Done!



Next topic...

Graphical User Interface II