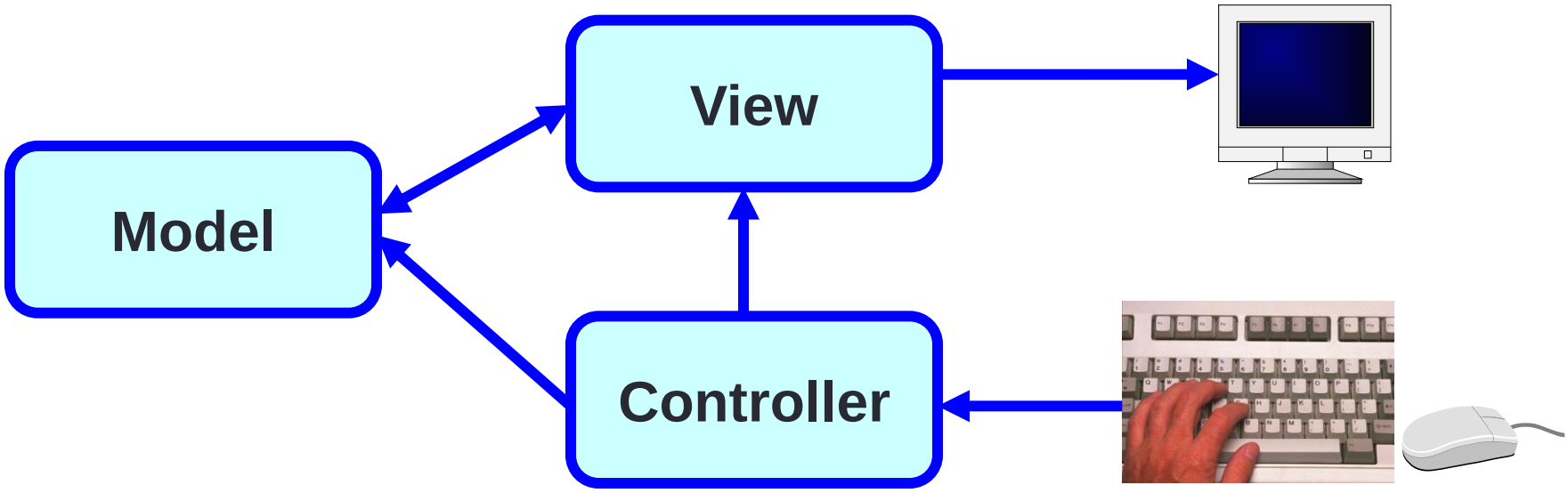
# CMSC 132: OBJECT-ORIENTED PROGRAMMING II


## Graphical User Interfaces (GUIs)

Department of Computer Science
University of Maryland, College Park

# Model-View-Controller (MVC)

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



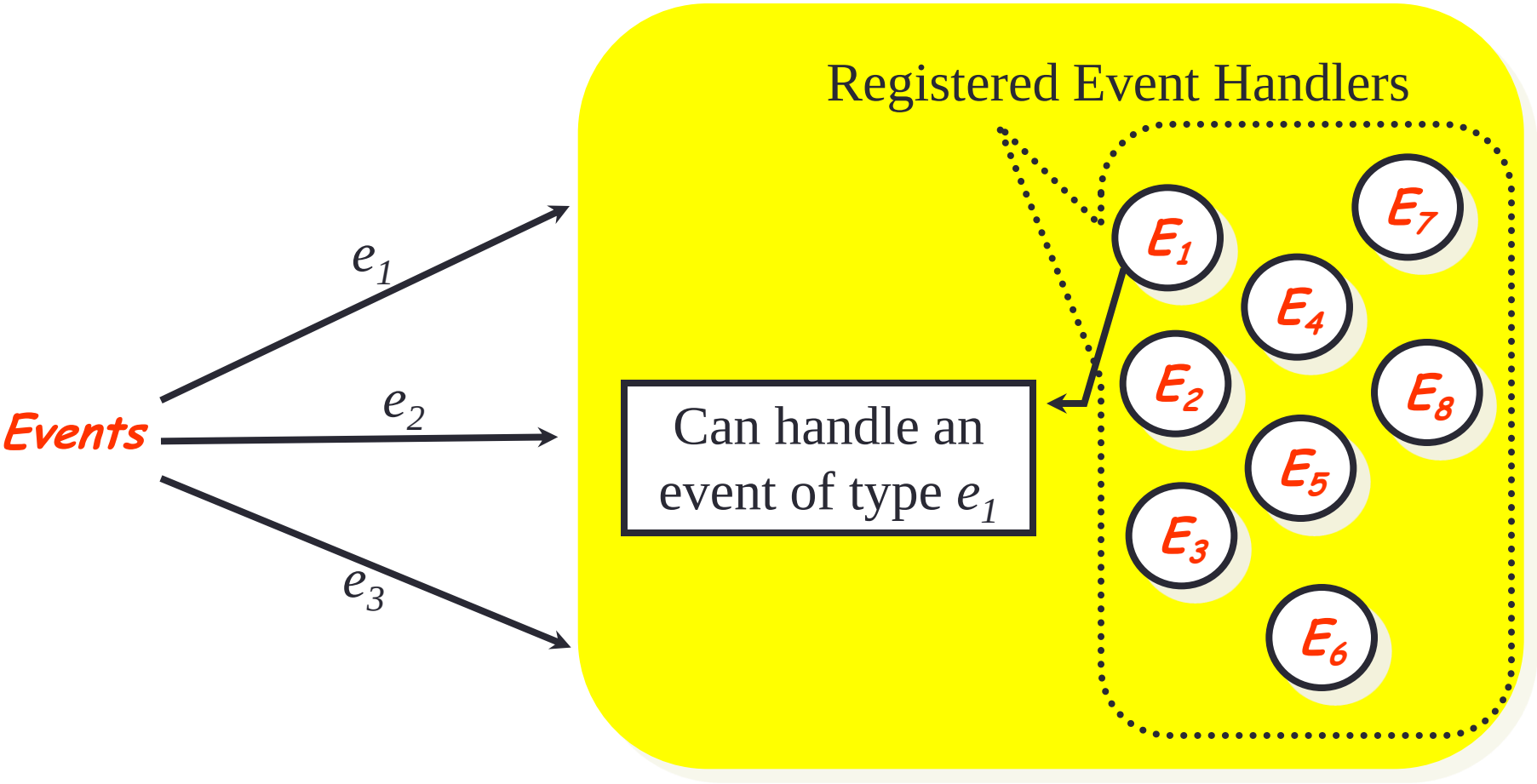
# MVC Model of GUI Design

- 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)

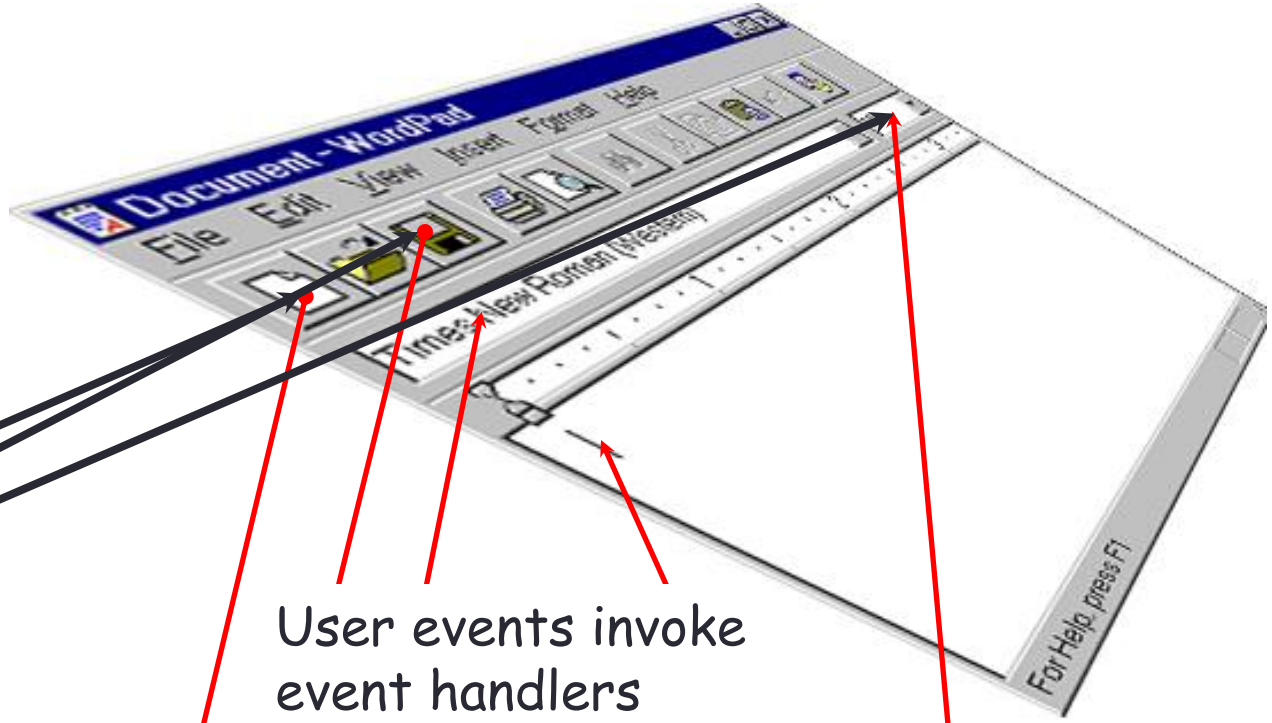
# Programming Models

- 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



# GUIs are Event-Driven Software




GUI

*User Events*

User events invoke event handlers

$E_3$ $E_4$ $E_5$ ..... $E_2$ $E_1$

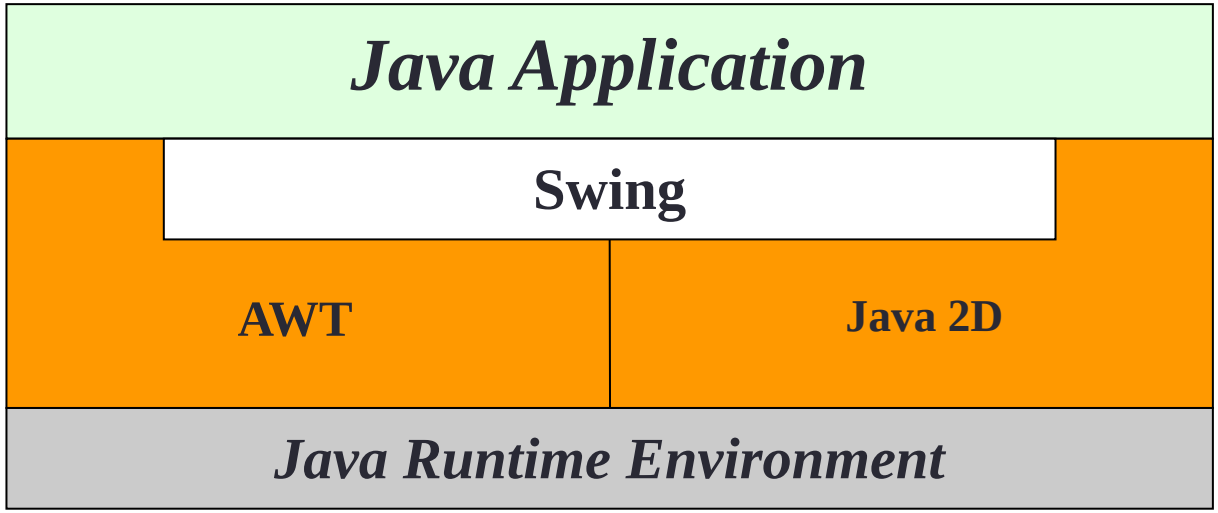
Event Handlers

**changeFontSizeActionPerformed (java.awt.event.ActionEvent evt)**

**newDocActionPerformed (java.awt.event.ActionEvent evt)**

**fileSaveActionPerformed (java.awt.event.ActionEvent evt)**

# GUIs in Java




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

- (Step 1) Define a container to hold components (e.g., JFrame, Japplet)
- (Step 2) Add GUI components to the container (JButton, JTextField, …)
  - Use layout manager to determine positions
- (Step 3) Add actions to GUI
  - Add (register) event listeners to GUI components
  - Usually one event listener class per widget (item user sees)
  - Inner class usually utilized to implement listener
  - Example of Java listeners & Actions Causing Event
    - ActionListener → clicking button in GUI
  - At run time
    - Java generates event object when events occur
    - Java then passes event object to event listener
- (Step 4) Schedule the GUI processing in the EDT (Event-Dispatching Thread)

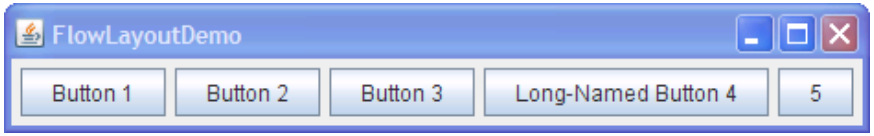
# Schedule GUI Processing in EDT

- What is a thread?
- Event Dispatching Thread (EDT) is a background thread to process GUI 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 bugs)
- Java Code that allows current thread to execute GUI code in dispatching thread

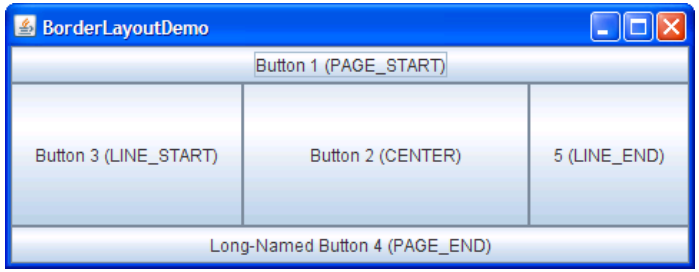
```
public static void main(String[] args) {
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndDisplayGUI(); // actually creates GUI
            }
        });
 }
```

# Java Layout Manager

- Definition
  - Arrangement of GUI components in container
- Layout manager
  - Entity translating layout specifications into actual coordinates at runtime, depending on conditions
- Examples
  - FlowLayout (lays out component from left to right)




  - BorderLayout (designates portions of the container as North, South, East, West, and Center)



# Examples

- Main Examples
  - eventHandlingIntro
  - singleClassBorderLayout
  - timer
  - textFileReaderSingleClass (illustrates MVC)
- Additional Examples
  - textFileReaderFont
  - textFilerReaderFontSlider
  - textFileReaderFont
  - tables

# Beware of Long Computations in Swing

- Swing uses a single-threaded model
- Long computations in the EDT freezes the GUI
- Usually you place the computation is a separate thread
- We will talk about this matter once we have covered threads

# Additional Resources

- Javadoc from the JDK
- Swing tutorial
  http://java.sun.com/docs/books/tutorial/uiswing/components/
- Filthy Rich Clients
  http://filthyrichclients.org/