

# CMSC 132: Object-Oriented Programming II

---



## Graphic User Interface (GUI)

Department of Computer Science  
University of Maryland, College Park

1

## 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)

2

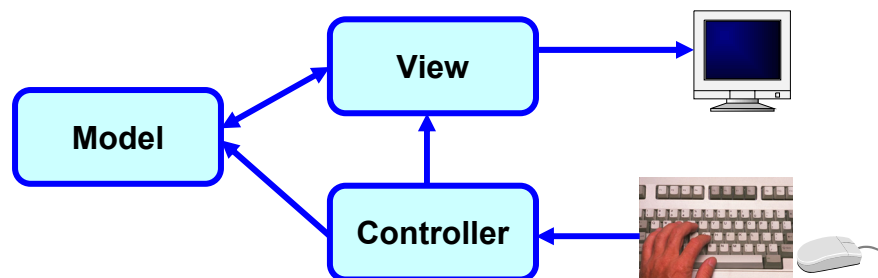
## GUI Topics

- Model-View-Controller model
- Java support for GUIs
  - Containers
  - Components
  - Events
- Event-driven programming

3

## 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



4

## 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)

5

## Java GUI Classes

### ■ AWT (Abstract Window Toolkit) (java.awt.\*)

- Old GUI framework for Java (Java 1.1)
- Some reliance on native code counterparts
- Platform independence problems

### ■ Swing (javax.swing.\*)

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

### ■ SWT (Standard Widget Toolkit; from Eclipse)

6

## Creating a GUI in Java

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

7

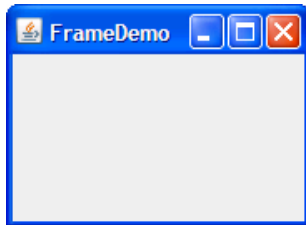
## GUI Elements 1 – Container

- **Definition**
  - Abstractions occupying space in GUI
- **Properties**
  - Usually contain one or more widgets
  - Can be nested in other containers
- **Examples**
  - JFrame, JDialog, JPanel, JScrollPane

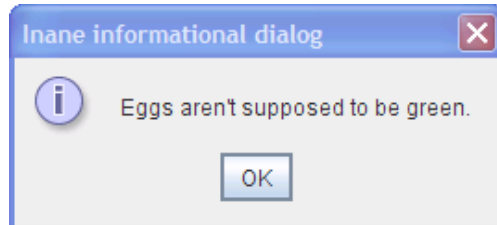
8

# Java Containers

## ■ JFrame

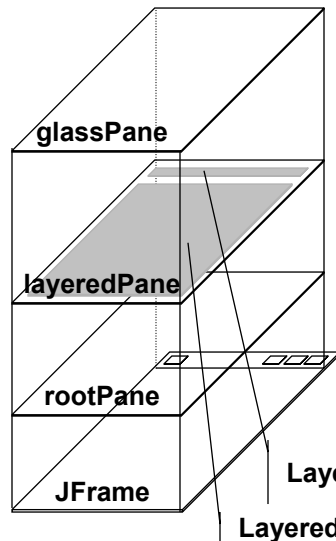


## ■ JDialog



9

# JFrame Structure



- Most things go into content pane
  - `getContentPane()`
- Use glassPane for pop up menus, some animations
- Methods
  - `getRootPane()`
  - `getLayeredPane()`
  - `getContentPane()`
  - `getGlassPane()`
- Can set...Pane explicitly

10

## GUI Elements 2 – Component

### ■ Definition

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

### ■ Examples

- Labels (fixed text)
- Text areas (for entering text)
- Buttons
- Checkboxes
- Tables
- Menus
- Toolbars
- Etc...

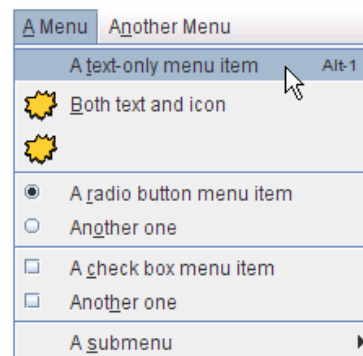
11

## Java Components

### ■ JButton



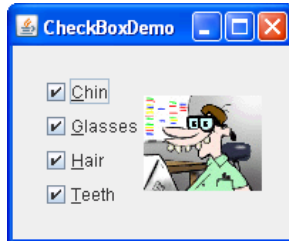
### ■ JMenu



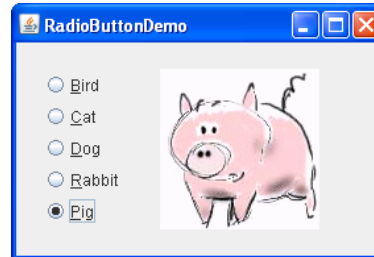
12

# Java Components

## ■ JCheckBox



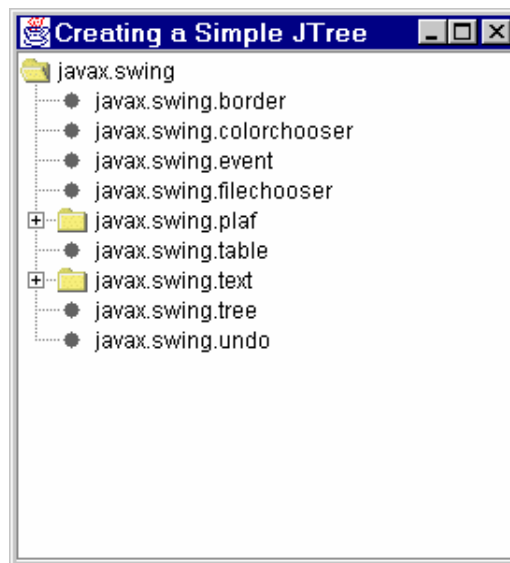
## ■ JRadioButton



13

# Java Components

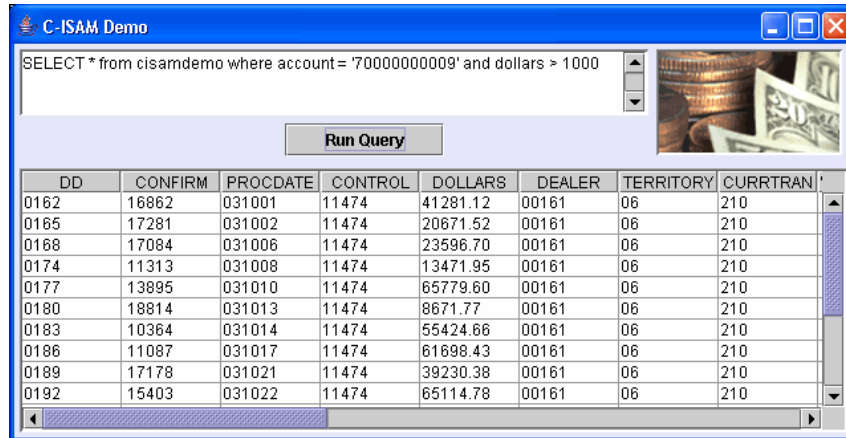
## ■ JTree



14

# Java Components

## ■ JTable



The screenshot shows a Java application window titled "C-ISAM Demo". At the top, there is a text area containing the SQL query: "SELECT \* from cisamdemo where account = '70000000009' and dollars > 1000". Below the text area is a "Run Query" button. To the right of the button is a small image of a stack of US dollar bills. Below the button is a JTable displaying the results of the query. The table has 8 columns: DD, CONFIRM, PROCDATE, CONTROL, DOLLARS, DEALER, TERRITORY, and CURRTRAN. The data is as follows:

DD	CONFIRM	PROCDATE	CONTROL	DOLLARS	DEALER	TERRITORY	CURRTRAN
0182	16862	031001	11474	41281.12	00161	06	210
0165	17281	031002	11474	20671.52	00161	06	210
0168	17084	031006	11474	23596.70	00161	06	210
0174	11313	031008	11474	13471.95	00161	06	210
0177	13895	031010	11474	65779.60	00161	06	210
0180	18814	031013	11474	8671.77	00161	06	210
0183	10364	031014	11474	55424.66	00161	06	210
0186	11087	031017	11474	61698.43	00161	06	210
0189	17178	031021	11474	39230.38	00161	06	210
0192	15403	031022	11474	65114.78	00161	06	210

15

# Java Components

## ■ JTable

- Each JTable object
  - Gets its data from an object implementing TableModel interface
  - Displays contents of TableModel object
- DefaultTableModel class implements TableModel
- Many different ways to use JTable to display data

16

## Layout

### ■ Definition

- Arrangement of GUI components in container

### ■ Layout specification

- Logical terms (2<sup>nd</sup> row, 1<sup>st</sup> column, left)
  - Preferred approach
- Actual coordinates (100 pixels, 5 inches)
  - Can be too rigid, limited to certain window sizes

17

## Java Layout Manager

### ■ Layout manager

- Entity translating layout specifications into actual coordinates at runtime, depending on conditions

### ■ Examples

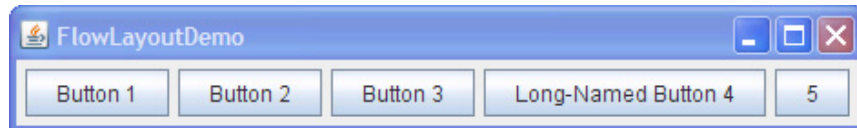
- FlowLayout
- BorderLayout
- GridLayout
- GridBagLayout

18

## Java Layout Manager

### ■ **FlowLayout**

- Lays out components from left to right

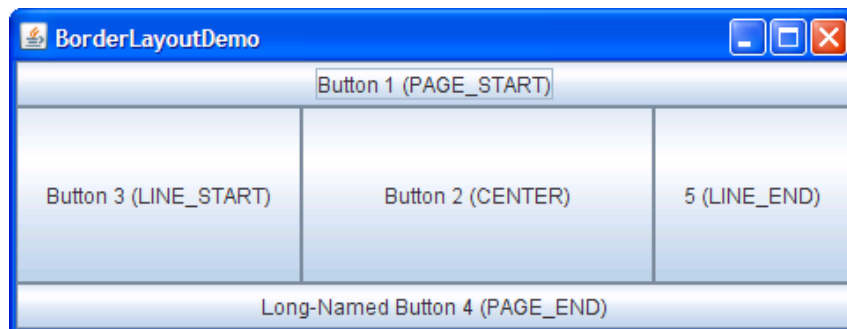


19

## Java Layout Manager

### ■ **BorderLayout**

- Designates portions of the container as North, South, East, West, and Center



20

## Java Layout Manager

### ■ GridLayout

- Lays out components in a grid (rows & columns)
- Makes components the same size

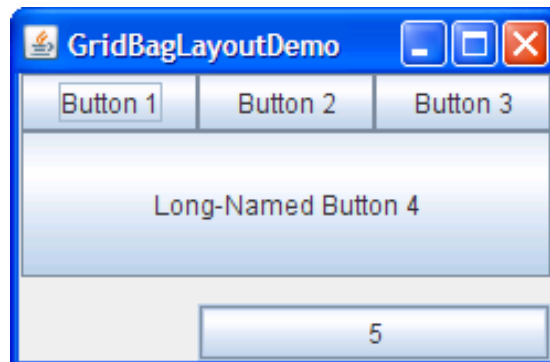


21

## Java Layout Manager

### ■ GridBagLayout

- Uses rows and columns of varying lengths
- Very flexible



22

## GUI Elements 3 – Events

### ■ Definition

- Action or condition occurring outside normal flow of control of program

### ■ Examples

- Mouse clicks
- Keyboard input
- Menu selections
- Window actions

23

## Event-driven Programming

### ■ Normal (control flow-based) programming

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

### ■ Event-driven programming

- Unable to predict time & occurrence of event
- Approach
  - Start with main()
  - Build GUI
  - Await events (& perform associated computation)

24

## Event-driven Programming in Java

### ■ During implementation

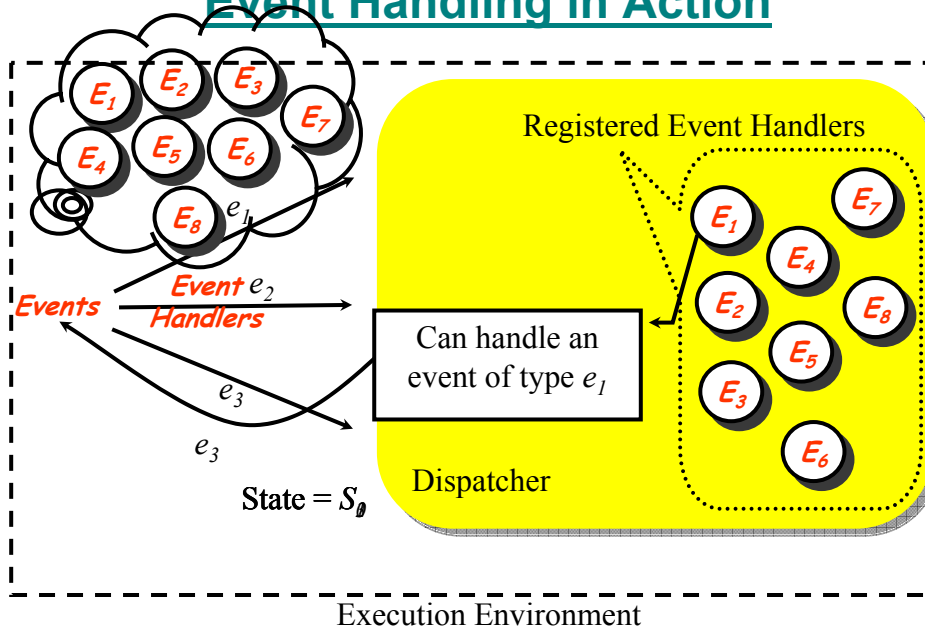
- Implement **event listeners** for each event
- Usually one event listener class per widget

### ■ At run time

- Register listener object with widget object
- Java generates **event object** when events occur
- Java then passes event object to event listener

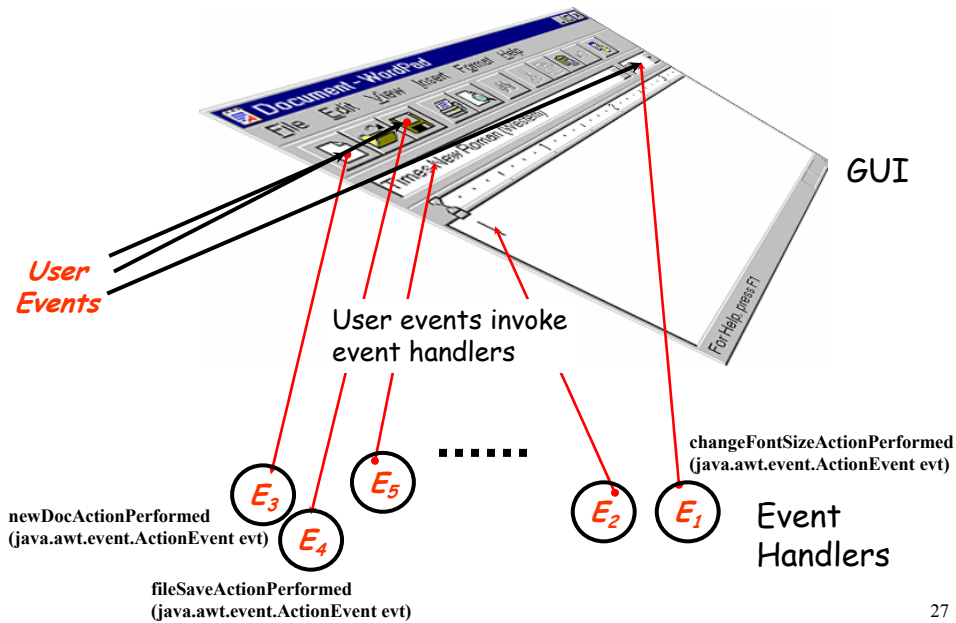
25

## Event Handling in Action



26

## GUIs are Event-Driven Software



## Event-driven Programming in Java

- **Example listeners & actions causing event**
  - **ActionEvent** ⇒ clicking button in GUI
  - **CaretEvent** ⇒ selecting portion of text in GUI
  - **FocusEvent** ⇒ component gains / loses focus
  - **KeyEvent** ⇒ pressing key
  - **ItemEvent** ⇒ selecting item from pull-down menu
  - **MouseEvent** ⇒ dragging mouse over widget
  - **TextEvent** ⇒ changing text within a field
  - **WindowEvent** ⇒ closing a window
- **In Java**
  - GUI events handled in **event dispatching thread**

## Event Dispatching Thread

- **Background thread to process events**
  - From AWT graphical interface event queue
- **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

29

## Event Dispatching Thread

- **Example code**
  - Allows current thread to execute GUI code in dispatching thread
  - `createAndDisplayGUI`
    - Method that actually defines the GUI

```
javax.swing.SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        createAndDisplayGUI();  
    }  
});
```

30

## Java Support For GUIs

- **Several GUI code examples**
  
- **Additional Resources**
  - **Appendix C of textbook**
  - **Javadoc for the JDK**
  - **Swing tutorial**
  - **Course slides and code handouts**
  - **Java Ranch**