CMSC 132:
Object-Oriented Programming II

Graphical User Interface (GUI)

Department of Computer Science

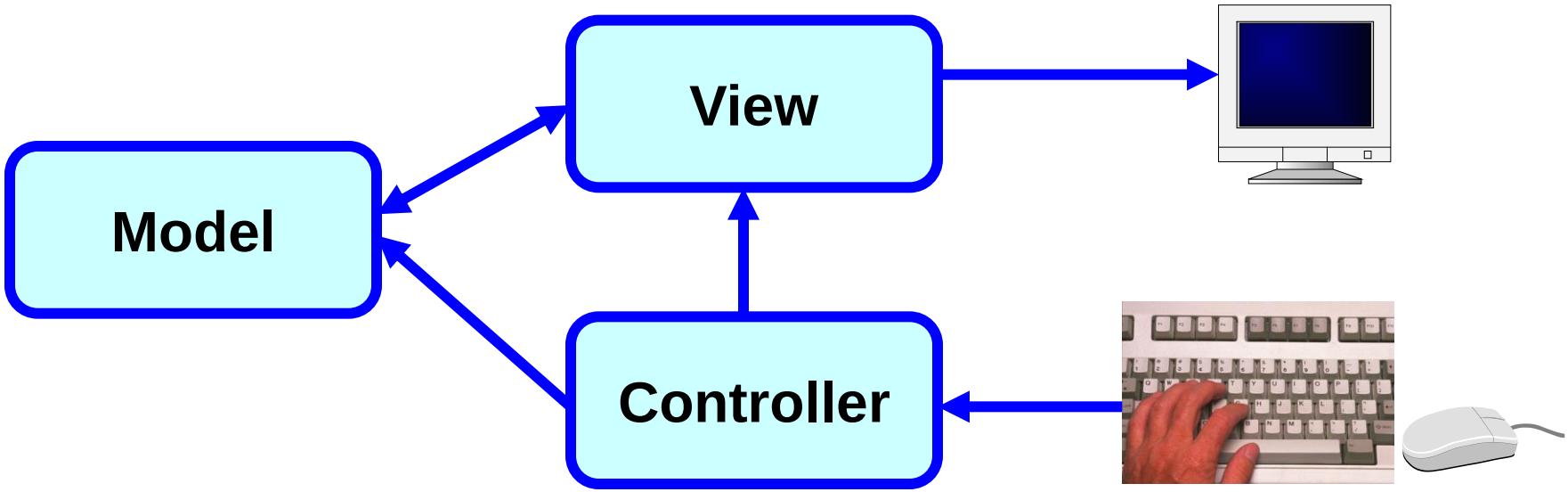
University of Maryland, College Park

Graphical User Interface (GUI)

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

Model-View-Controller (MVC)

- **Model for GUI programming (Xerox PARC ’78)**
- **Separates GUI-Oriented Program into components:**
  1. **Model ⇒ application data**
  2. **View ⇒ visual interface**
  3. **Controller ⇒ user interaction**



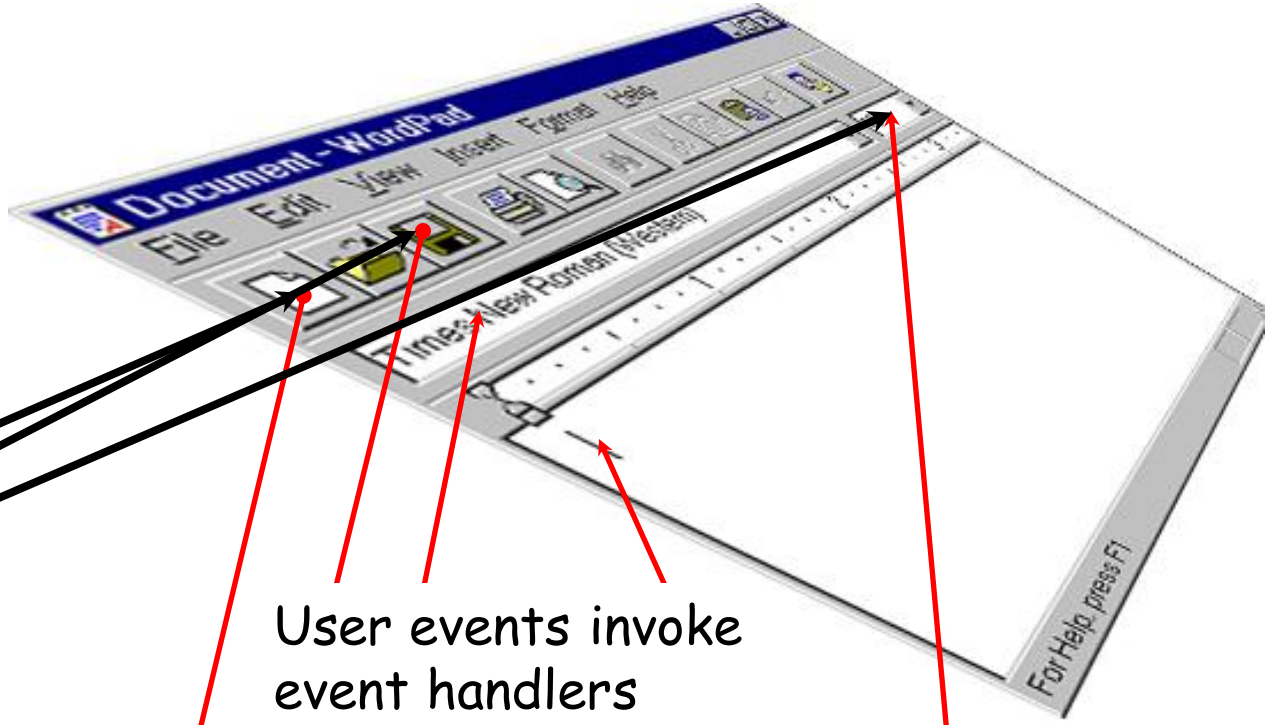
MVC Model of GUI Design

- **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)**

GUIs are Event-Driven Software



GUI

User Events

User events invoke event handlers

E_3 E_4 E_5 ..... E_2 E_1

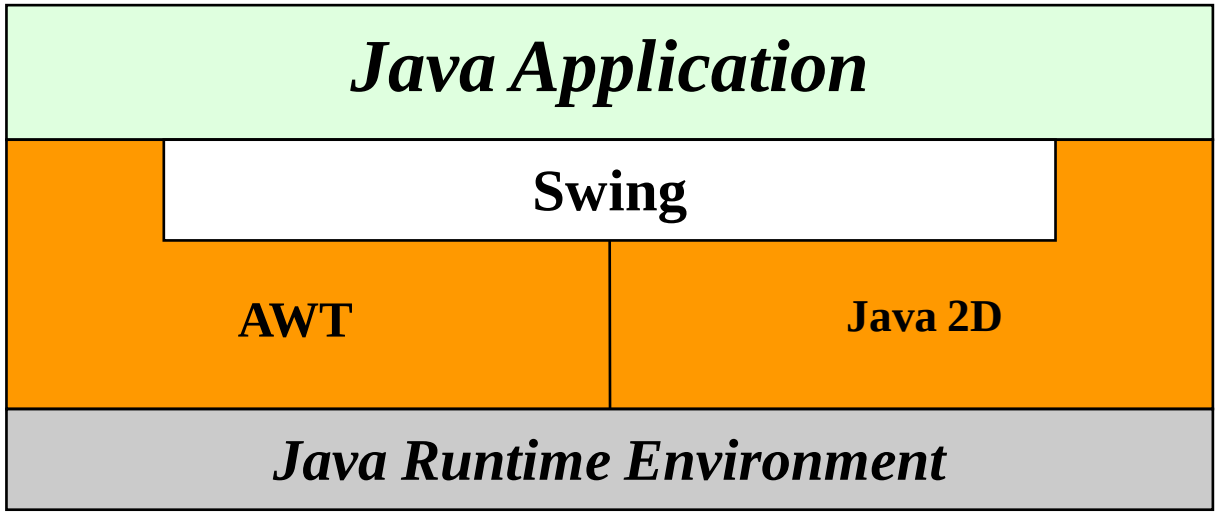
Event Handlers

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 advanced 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

Steps for Creating a GUI in Java

1. Define a container to hold components
   - Examples: JFrame, JApplet…
2. Optional: Choose a Layout Manager
3. Add GUI components to the container
   - Examples: JButton, JTextField, JScrollBar…
   - layout manager will determine positions
4. Add actions to GUI
   - Add event listeners to GUI components
5. Schedule the GUI processing in the EDT (Event-Dispatching Thread)

Step 1 (Define Container)

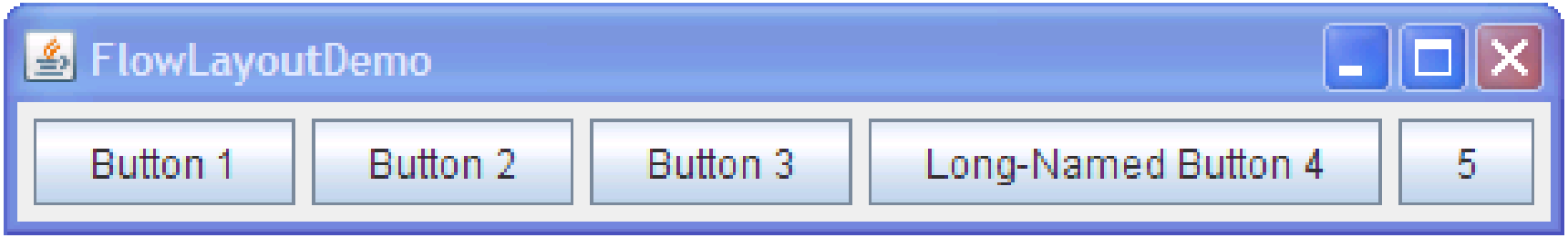
- **Container Definition**
  - **Abstractions occupying space in GUI**
- **Properties**
  - **Usually contain one or more widgets**
    - **widget - actual item user can see**
  - **Can be nested in other containers**
- **Container Examples**
  - **JFrame, JDialog, JPanel, JScrollPane**

Step 2: Choose Layout Manager

- **Layout Manager defines how widgets are arranged within the container**

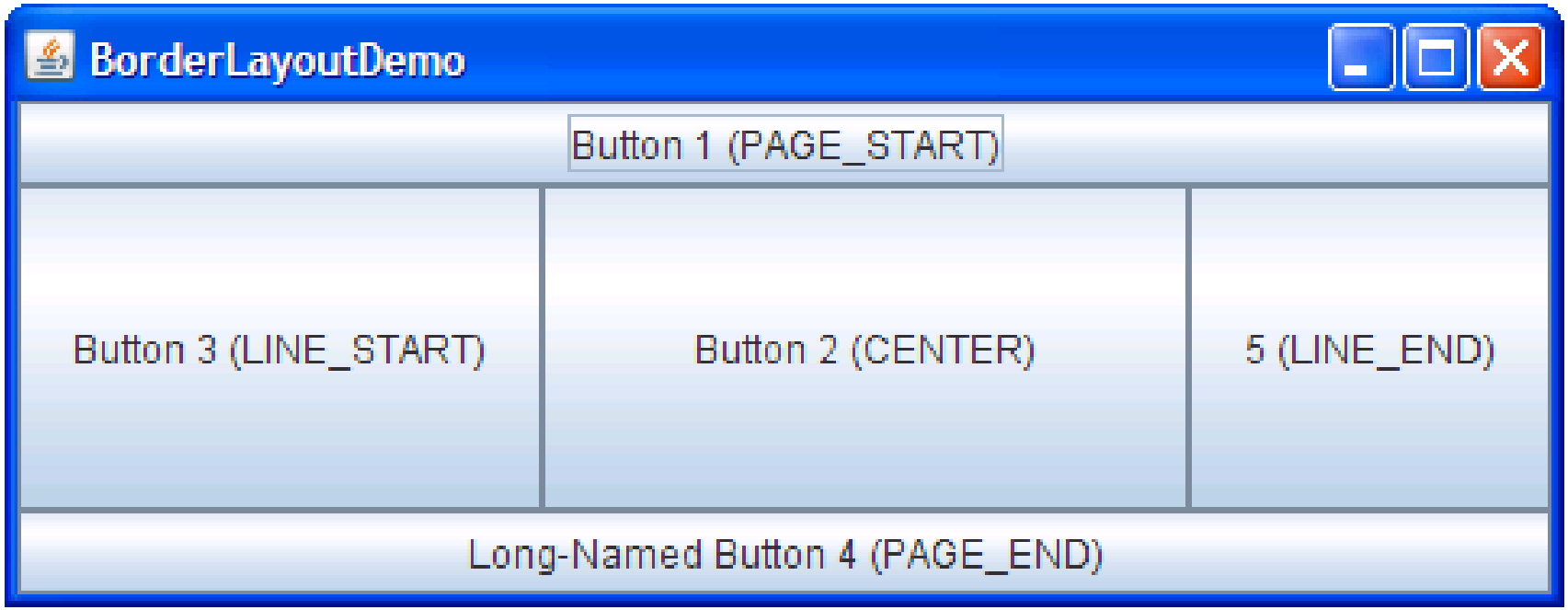
Java Layout Manager

- **FlowLayout**
  - **Default Layout**
  - **Lays out components from left to right**



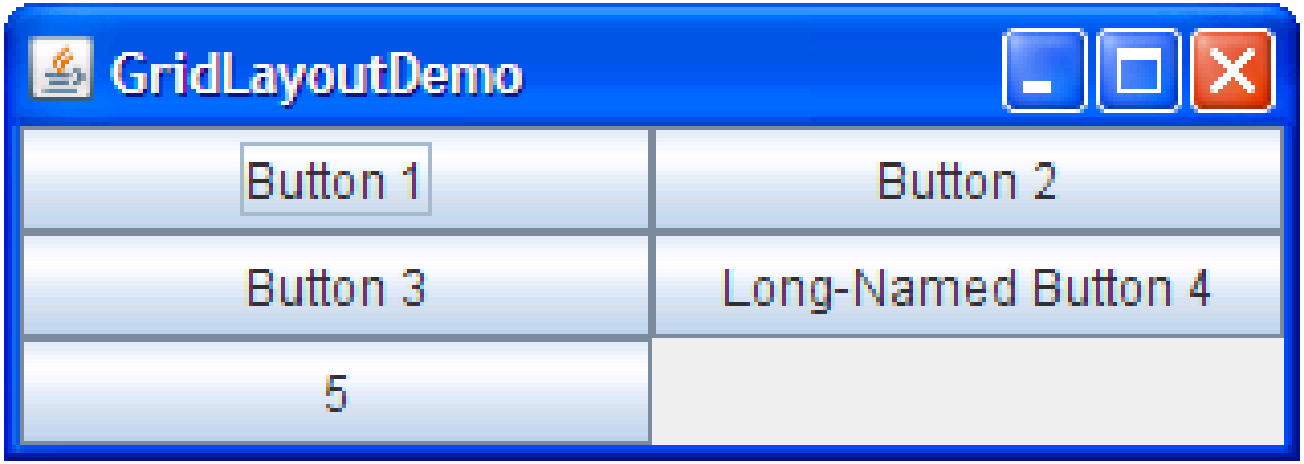
Java Layout Manager

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



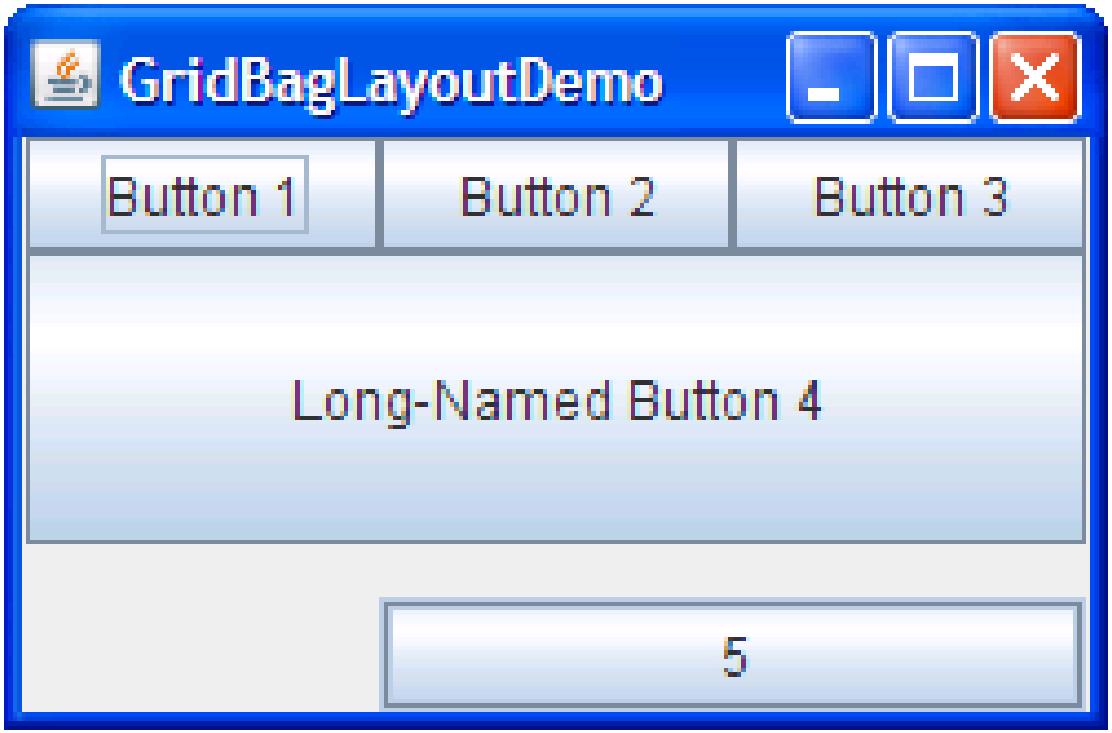
Java Layout Manager

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



Java Layout Manager

- **GridBagLayout**
  - **Uses rows and columns of varying lengths**
  - **Very flexible**



Other Layout Mangers

- **Overview is here:**

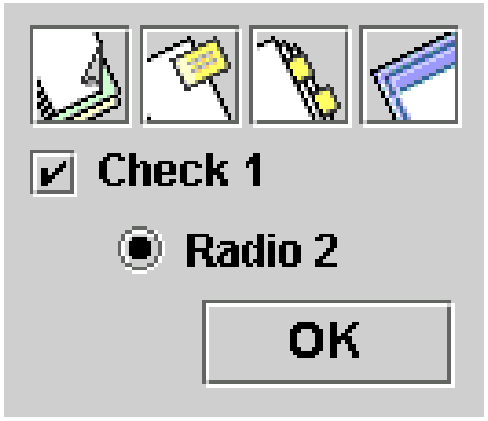
http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html

Step 3 (Define Components)

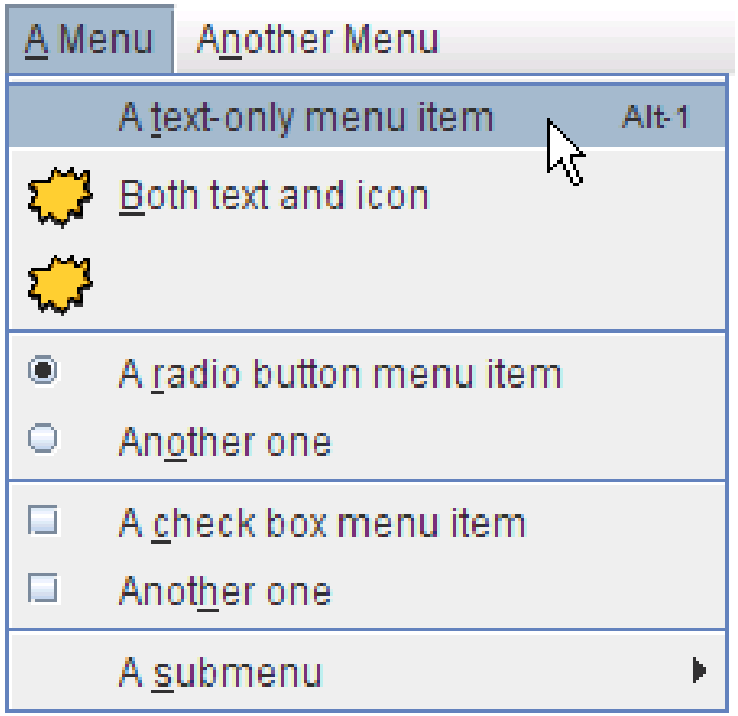
- **Component Definition**
  - **Actual items (widgets) user sees in GUI**
- **Examples**
  - **Labels (fixed text)**
  - **Text areas (for entering text)**
  - **Buttons**
  - **Checkboxes**
  - **Tables**
  - **Menus**
  - **Toolbars**
  - **Etc…**

Java Components

- JButton

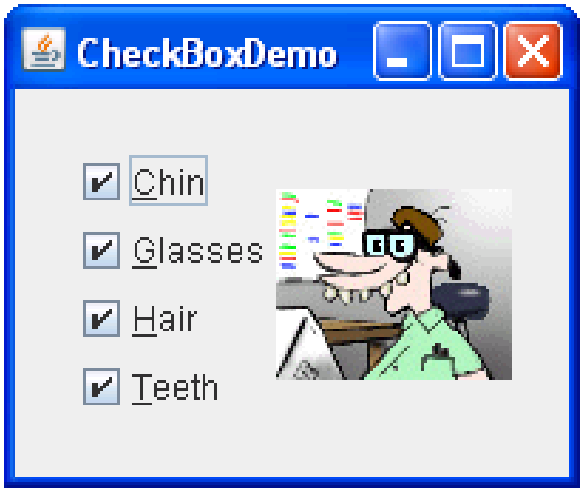


- JMenu

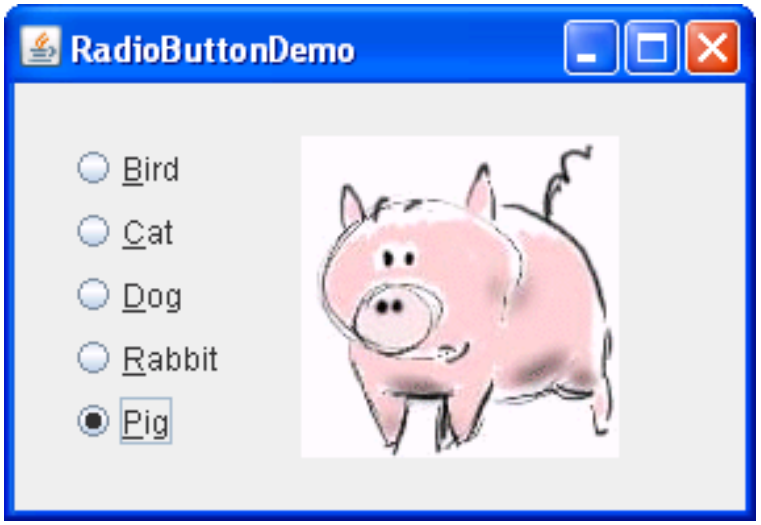


Java Components

- JCheckBox

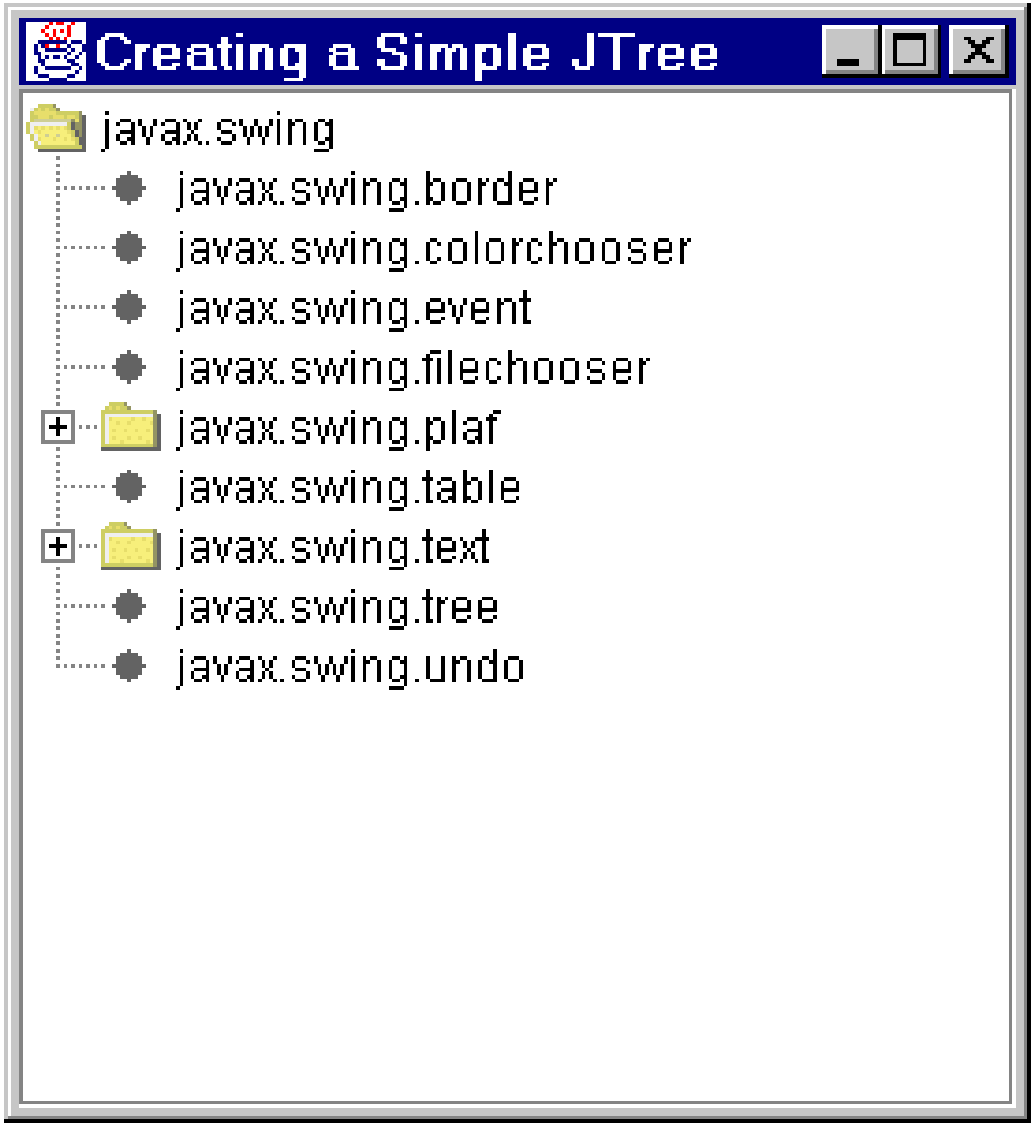


- JRadioButton



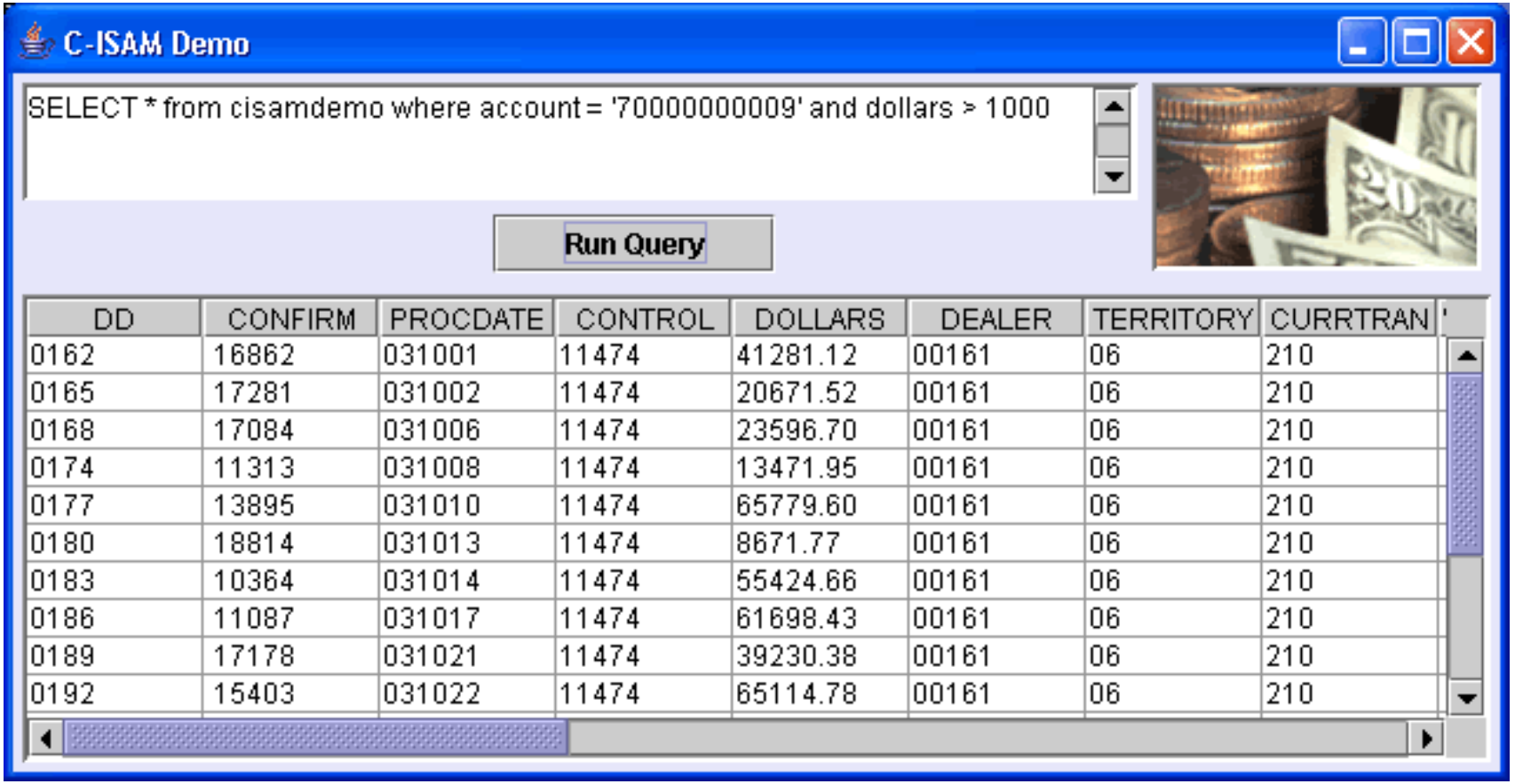
Java Components

- **JTree**



Java Components

- **JTable**



DD	CONFIRM	PROCDATE	CONTROL	DOLLARS	DEALER	TERRITORY	CURRTRAN
0162	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

Step 4 (Set Event Listeners)

- **Implementation**
  - Implement event listeners
  - Register (add) listener object with widget
  - Inner class usually utilized to implement 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 5 (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

- **Passing code to EDT.**

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

Simple Example

- **EXAMPLES: SimpleGUI1, SimpleGUI2**

Additional Resources

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