Graphic 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)
GUI Topics

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

- JFrame
- JDialog
**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

LayeredPane manages (optional) JMenuBar
LayeredPane contains contentPane
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…
Java Components

- JButton
- JMenu
Java Components

- JCheckBox
- JRadioButton
Java Components

- JTree

![Diagram of a Simple JTree](image-url)
Java Components

**JTable**

![JTable Example](image)

```sql
SELECT * from cisamdemo where account = '70000000009' and dollars > 1000
```

<table>
<thead>
<tr>
<th>DD</th>
<th>CONFIRM</th>
<th>PROCDATE</th>
<th>CONTROL</th>
<th>DOLLARS</th>
<th>DEALER</th>
<th>TERRITORY</th>
<th>CURRTRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0162</td>
<td>16862</td>
<td>031001</td>
<td>11474</td>
<td>41281.12</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0165</td>
<td>17281</td>
<td>031002</td>
<td>11474</td>
<td>20671.52</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0168</td>
<td>17084</td>
<td>031006</td>
<td>11474</td>
<td>23596.70</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0174</td>
<td>11313</td>
<td>031008</td>
<td>11474</td>
<td>13471.95</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0177</td>
<td>13895</td>
<td>031010</td>
<td>11474</td>
<td>65779.60</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0180</td>
<td>18814</td>
<td>031013</td>
<td>11474</td>
<td>8671.77</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0183</td>
<td>10364</td>
<td>031014</td>
<td>11474</td>
<td>55424.66</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0186</td>
<td>11087</td>
<td>031017</td>
<td>11474</td>
<td>61698.43</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0189</td>
<td>17178</td>
<td>031021</td>
<td>11474</td>
<td>39230.38</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
<tr>
<td>0192</td>
<td>15403</td>
<td>031022</td>
<td>11474</td>
<td>65114.78</td>
<td>00161</td>
<td>06</td>
<td>210</td>
</tr>
</tbody>
</table>
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
**Layout**

- **Definition**
  - Arrangement of GUI components in container

- **Layout specification**
  - Logical terms (2\textsuperscript{nd} row, 1\textsuperscript{st} column, left)
    - Preferred approach
  - Actual coordinates (100 pixels, 5 inches)
    - Can be too rigid, limited to certain window sizes
Java Layout Manager

Layout manager

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

Examples

- FlowLayout
- BorderLayout
- GridLayout
- GridBagLayout
Java Layout Manager

FlowLayout

- 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
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
Programming Models

- 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)
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
Event Handling in Action

Events

Registered Event Handlers

Can handle an event of type $e_1$

Dispatcher

Execution Environment

State = $S_0$
GUIs are Event-Driven Software

User events invoke event handlers

User events:
- \( E_1 \): changeFontSizeActionPerformed (java.awt.event.ActionEvent evt)
- \( E_2 \): newDocActionPerformed (java.awt.event.ActionEvent evt)
- \( E_3 \): fileSaveActionPerformed (java.awt.event.ActionEvent evt)
- \( E_4 \)
- \( E_5 \)
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
Event Dispatching Thread

Example code

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

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