Chapter 9: Graphical User Interfaces

- Users have become accustomed to using a graphical user interface (GUI) through which they interact with a program
- Java provides strong support for building GUIs through the `java.awt` package
- Chapter 9 focuses on:
  - GUI components
  - event-driven programming
  - containers and component hierarchies
  - layout managers

GUI Elements

- The key elements of a Java graphical user interface:
  - GUI components
  - layout managers
  - event processing
- *GUI components*, such as text fields and buttons, are the screen elements that a user manipulates with the mouse and keyboard
- *Layout managers* govern how the components appear on the screen
- *Events* signal important user actions, like a mouse click

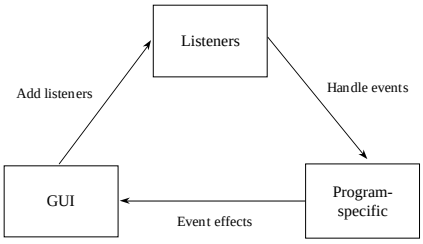
Event-Driven Programming

- Programs with GUIs must respond to events, generated by GUI components, that indicate that specific actions have occurred
- A special category of classes, called *listeners*, wait for events to occur
- Therefore, a GUI program is composed of:
  - the code that presents the GUI to the user
  - the listeners that wait for events to occur
  - the specific code that is executed when events occur

Event-Driven Programming

- There is a *listener interface* defined for each event type
- Each listener interface contains the abstract methods required to respond to specific events
- A listener class implements a particular listener interface
- Listeners are "added" to a particular GUI component
- When a component generates an event, the method corresponding to that event is executed in the listener

The GUI Program Model



Event Interfaces

- Multiple listeners can be added to a component
- Multiple components can be processed by the same listener
- Furthermore, one listener class can implement multiple listener interfaces
- Therefore one class can listen for many types of events
- See `Events.java`

Containers

- A *container* is a special category of GUI components that group other components
- All containers are components, but not all components are containers
- An applet is a container
- Therefore, buttons, text fields, and other components can be added to an applet to be displayed
- Each container has an associated layout manager to control the way components in it are displayed

Containers

- Some containers must be attached to another graphical surface:
  - panel
  - applet
- An applet is attached to a browser or appletviewer window
- Other containers can be moved independently:
  - window
  - frame
  - dialog

Component Hierarchies

- A GUI is created when containers and other components are put together
- The relationships between these components form a *component hierarchy*
- For example, an applet can contain panels which contain other panels which contain buttons, etc.
- See `Rings_Display.java`
- Careful design of the component hierarchy is important for visually pleasing and consistent GUIs

Layout Managers

- There are five predefined layout managers in the `java.awt` package:
  - flow layout
  - border layout
  - card layout
  - grid layout
  - grid bag layout
- Each container has a particular layout manager associated with it by default
- A programmer can also create custom layout managers

Flow Layout

- Components are placed in a row from left to right in the order in which they are added
- A new row is started when no more components can fit in the current row
- The components are centered in each row by default
- The programmer can specify the size of both the vertical and horizontal gaps between the components
- Flow layout is the default layout for panels and applets
- See `Flow.java`

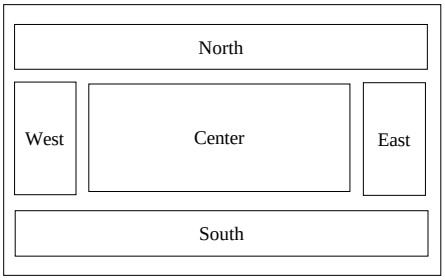
Grid Layout

- Components are placed in a grid with a user-specified number of columns and rows
- Each component occupies exactly one grid cell
- Grid cells are filled left to right and top to bottom
- All cells in the grid are the same size
- See `Grid.java`

Border Layout

- Defines five locations each of which a component or components can be added
  - North, South, East, West, and Center
- The programmer specifies the area in which a component should appear
- The relative dimensions of the areas are governed by the size of the components added to them
- See `Border.java`

Border Layout



Card Layout

- Components governed by a card layout are "stacked" such that only one component is displayed on the screen at any one time
- Components are ordered according to the order in which they were added to the container
- Methods control which component is currently visible in the container
- See `Card.java`

Grid Bag Layout

- Designed as a two-dimensional grid of columns and rows
- However, not all cells in the grid are the same size
- Components may span multiple columns and rows
- Each component in a grid bag layout is associated with a set of constraints, defined by the `GridBagConstraints` class
- A grid bag layout is the most versatile, and most complex, of the predefined layout managers
- See `Grid_Bag.java`

GUI Design

- Careful design of a graphical user interface is key to a viable software system
- To the user, the user interface is the system
- For each situation, consider which components are best suited and how they should best be arranged
- See `Quotes.java`