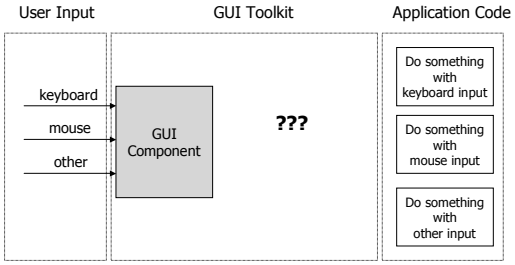
CS202 Java Object Oriented Programming
GUI Programming – Event Handling

Chengyu Sun
California State University, Los Angeles

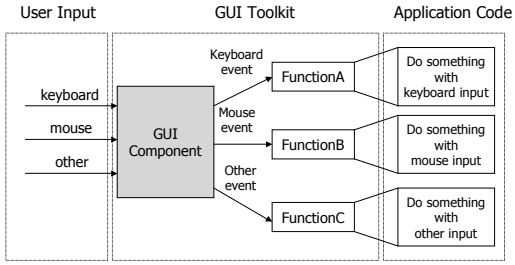
Events

- Mouse clicks
- Mouse movement
- Keyboard input
- Window state changes (minimized, maximized, closed)
- Table or list selection changes
- …

GUI Toolkit



Callback Functions and Events



Callback Functions and Event

- Callback functions
  - Function names are pre-defined – so the GUI toolkit knows which function to invoke when it receive some user input
  - No function implementation – exactly what to do with the input is left to the application
- Events
  - Parameters to the callback functions
  - Information about the user input
    - Which key is pressed, which button is clicked, position of the cursor …

Java Event Handling

- Application
  - Event handling code must implement some event listener interface
  - Event handler must be registered to some GUI component(s)
    - component.addEventListener(EventHandler)
- System
  - When certain event happens, system creates an event object
  - Invokes event handling code with the event object as parameter

Swing Events and Listeners

- http://java.sun.com/docs/books/tutorial/uiswing/events/eventsandcomponents.html
- Three important types of events
  - Action events
  - Mouse events
  - Mouse motion events

Action Events

- Clicks a button, presses Return when typing in a text field, or chooses a menu item
- `ActionEvent` class
  - getSource()
  - getActionCommand()

Handle Action Events

- Register to listen to Action Events
  - component.addActionListener(*EventHandler*)
- *EventHandler* must be an object of a class which implements the ActionListener interface

```
public interface ActionListener {

   public void actionPerformed( ActionEvent e );

}
```

Mouse Events

- Presses a mouse button while the cursor is over a component
- `MouseEvent` class
  - getButton()
  - getClickCount()
  - getX()
  - getY()

Handle Mouse Events

- Register to listen to Mouse Events
  - component.addMouseListener(EventHandler)
- *EventHandler* must be an object of a class which implements the MouseListener interface

```
public interface MouseListener {

   public void mouseClicked( MouseEvent e );
   public void mouseEntered( MouseEvent e );
   public void mouseExited( MouseEvent e );
   public void mousePressed( MouseEvent e );
   public void mouseReleased( MouseEvent e );
}
```

Mouse Motion Events

- Moves the mouse over a component
- Mouse motion events are also represented by the `MouseEvent` class

Handle Mouse Motion Events

- Register to listen to Mouse Motion Events
  - component.addMouseMotionListener(EventHandler)
- *EventHandler* must be an instance of a class which implements the MouseMotionListener interface

```
public interface MouseMotionListener {

    public void mouseMoved( MouseEvent e );
    public void mouseDragged( MouseEvent e );
}
```

Mouse Event Example

- DrawLines.java

Better Event Handling Code – Inner Class

- Do we have to use the main class as the event handler?

```
public class EventDemo extends JFrame {

    SomeEventHandler h = new SomeEventHandler();
    Jbutton b = new Jbutton( "Button" );
    b.addActionListener( h );
      … …

  class SomeEventHandler implements ActionListener {
    … …
  }
}
```

Better Event Handling Code – Adapter Class

- Do we have to leave all those empty methods around?

```
public class EventDemo extends JFrame {

    SomeEventHandler h = new SomeEventHandler();
    JButton b = new JButton( "Button" );
    b.addMouseListener( h );
      … …

  class SomeEventHandler extends MouseAdapter {
    … …
  }
}
```

Better(?) Event Handling Code – Anonymous Inner Class

- Do we have to use the same class to handle the events for all components?

```
public class EventDemo extends JFrame {

    JButton b = new JButton( "Button" );
    b.addMouseListener(
      new MouseAdapter() {
        public void mouseClicked(MouseEvent e) {
          …
        }
      });
    … …
}
```