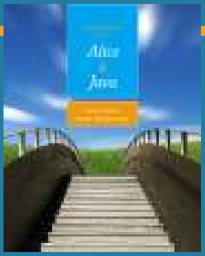
Chapter 7: Events

Programming with Alice and Java
First Edition

by
John Lewis
and
Peter DePasquale



Objectives

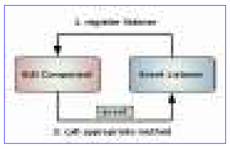
- Explore the core elements of event processing in Java.
- Learn about the different types of events in Java.
- Explore various components used in a graphical user interface.
- Use listener objects to process Java events.
- See how inner classes can be used effectively to create listeners.

Event Processing

- Event processing is a technique used to create a *graphical user interface* (GUI).
- A GUI *component* is an object that represents a screen element (button, text field, slider, menu).
- *Containers* are components that can hold other components.
- GUI components generate events that represent user actions related to that component.
- A program that is oriented around the GUI, responding to events from the user, is called *event-driven*.

Event Processing (continued)

- A listener is an object that "waits" for an event to occur and responds when it does occur.
- A relationship between the listener and the component that generates an event should be established.

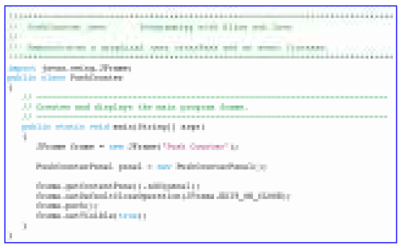


Event Processing (continued)

- Many components and events are predefined by classes in the Java class library.
- A programmer can write her own listener classes to perform any action she desires when events occur.
- To create a Java program that uses GUI, a programmer must:
  - create and set up the necessary components;
  - implement listener classes that define what happens when a particular event occurs;
  - establish the relationship between the listeners and the components that generate the events of interest.

Buttons

The PushCounter class



```
//********************************************************************
//  PushCounter.java       Programming with Alice and Java
//
//  Demonstrates a graphical user interface and an event listener.
//********************************************************************
import javax.swing.JFrame;
public class PushCounter
{
   //-----------------------------------------------------------------
   //  Creates and displays the main program frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Push Counter");

      PushCounterPanel panel = new PushCounterPanel();

      frame.getContentPane().add(panel);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Buttons (continued)

- A *label* can be used to display a line of text or an image.
- A *push* button allows the user to initiate an action with the press of a button.
- A JButton is a class that generates this action event.
- Several events are defined in the Java standard library.

Action Events

- To respond to the event a listener object is needed.
- The Listener class represents the action listener.

The actionPerformed method is called when the event occurred and ActionEvent object is passed to that method

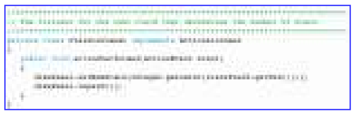
Event Types

Event	What it Indicates	Listener Interface
ActionEvent	A button was pushed, enter was pressed in a text field, or a menu item was selected.	ActionListener
ChangeEvent	An object's state changed in some way.	ChangeListener
ComponentEvent	A component was hidden, moved, resized, or shown.	ComponentListener
ContainerEvent	A component was added or removed from a container.	ContainerListener
FocusEvent	A component gained or lost the keyboard focus.	FocusListener
ItemEvent	The state of selectable item (check box, menu) changed.	ItemListener
KeyEvent	A Keyboard key was pressed.	KeyListener
MouseEvent	The mouse interacted with a component (mouse button push, rollover, drag).	MouseListener MouseMotionListener
WindowEvent	The application window was opened, closed, iconified, maximized, minimized, etc,	WindowFocusListener WindowListener WindowStateListener

Another Example

- Controls include a *text box*, *check boxes* and a set of *radio buttons*.
- Instead of using a separate public class for each listener, the public class contains private classes that define listeners.
- These classes are called *inner classes*.
- Only outer classes can make use of the inner class.

Listener Examples



A text field listener

A check box listener

Mouse Events

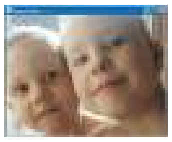
- A mouse event is generated when the mouse interacts with a GUI component.
- Components can generate mouse events that indicate that:
  - a mouse button was pressed;
  - a mouse button was released;
  - a mouse button was clicked;
  - the mouse cursor moved over a component;
  - the mouse cursor moved off a component;
  - the mouse was moved;
  - the mouse was dragged.

MouseListener interface (first five)

MouseMotionListener interface (last two)

Keyboard Events

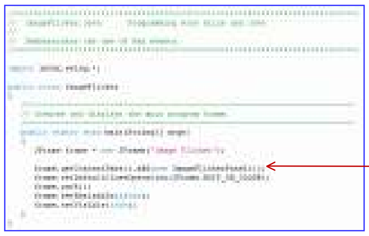
- A *keyboard event* (or *key event*) occurs when a keyboard key is pressed.
- Key events allow a program to respond immediately as the user presses keys.
- A listener in Java responds when *any* key is pressed, then decides what to do based on the specific key pressed.



In the ImageFlicker program three different images appear in the image frame when user presses 1, 2, or 3. When space bar is pressed one of the images is picked at random.

Example

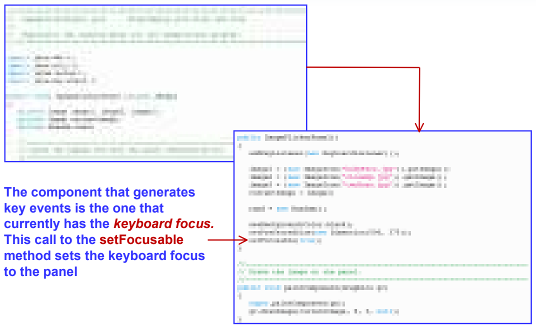
The main method



[illegible]

Display panel for the program

Example (continued)

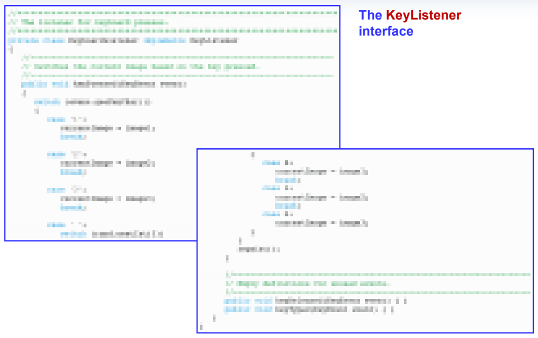


[illegible]

The component that generates key events is the one that currently has the *keyboard focus*. This call to the setFocusable method sets the keyboard focus to the panel

Example (continued)

The KeyListener interface



[illegible]

Summary

- A GUI is made up of components, events that represent user actions, and listeners that respond to those events.
- A listener can be created by implementing an appropriate listener interface.
- Listeners are often defined as inner classes because of the intimate relationship between the listener and the GUI components.
- Radio buttons operate as a group, providing a set of mutually exclusive options.
- Java mouse events are separated into two categories with two listener interfaces.
- A listener may have to provide empty method definitions for unused events to satisfy the interface.
- To generate a keyboard event, a component must have the keyboard focus.