IJARSCT
ISSN (Online) 2581-9429
ISSN (Print) 2581-XXXX
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 1, Issue 1, January 2021
Impact Factor: 4.819

The Delegation Event Model in Java

Meenakshi Khamkar
Lecturer, Department of Computer Engineering
Vidyalankar Polytechnic, Mumbai, India

Abstract: *The delegation event model provides mechanisms to generate and handle events in Java. When user performs any action for e.g. Clicking on a button, selecting option from list, that action is handled and processed using event handling concepts i.e. event, source of event, Event Listener. How this complete process is handled that we will be exploring in this paper.*

Keywords: Event handling in Java, Event Delegation Model in Java, Event Listener, Source of Event

I. INTRODUCTION

When we design any user interactive GUI in java, we need to handle different user actions i.e. Events. When User click on button, which process is followed in background that we will be exploring. In Event Delegation Model a source creates an event and sends it to one or more listeners. The listener accepts that event. Once the event is accepted, the listener handles that event using a separate piece of code. The advantage of this Model is that the logic which handles events is completely separated from the user interface logic that generates those events. The following sections define different terms used in Delegation Event Model.

1. **Event:** An event is an action which is generated by source. For e.g. when we click on button it should proceed further, so clicking is an event.
2. **Event Source:** A source is an object which generates an event. In above example button is source of event.

Following table gives different sources of events.

Button	Generates action events when the button is pressed.
Check box	Generates item events when the check box is selected or deselected.
Choice	Generates item events when the choice is changed.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Menu Item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scroll bar	Generates adjustment events when the scroll bar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.
Keys on keyboard	Generates key Event when key is pressed
Mouse	Generates mouse Event when mouse is pressed

3. **Event Listener:** A listener is an interface which is notified when an event is generated. It must be registered with one or more sources to receive notification about events. It also implements methods to process events.
4. **Event Classes:** There are different Event classes available for e.g. ActionEvent class for handling button, list, Menubar related events. Following table provides different Event Classes.

Event Class	Description
Action Event	Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
AdjustmentEvent	Generated when a scroll bar is manipulated.
ComponentEvent	Generated when a component is hidden, moved, resized, or becomes visible.

IJARSCT
ISSN (Online) 2581-9429
ISSN (Print) 2581-XXXX
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 1, Issue 1, January 2021
Impact Factor: 4.819

ContainerEvent	Generated when a component is added to or removed from a container.
FocusEvent	Generated when a component gains or loses keyboard focus.
InputEvent	Abstract super class for all component input event classes.
ItemEvent	Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.
KeyEvent	Generated when input is received from the keyboard.
MouseEvent	Generated when the mouse is dragged, moved, clicked, pressed, or released;
MouseWheelEvent	also generated when the mouse enters or exits a component.
TextEvent	Generated when the mouse wheel is moved. (Added by Java 2, version 1.4)
WindowEvent	Generated when the value of a text area or text field is changed.

II. STEPS FOR EVENT HANDLING

1. Import event handling package.
2. Implement respective Listener Interface.
3. Register Listener using registration methods for particular source.
4. Implement methods of specific Listener Interface for processing events.

For studying all these steps we will take one example.

Example: Design one application for displaying 2 buttons yes, no using Java. After clicking each button, it should display different messages. In this problem statement button is a source of event, clicking is an event, for handling button events ActionListener Interface and actionPerformed method from same Listener is used.

Button program with Event Handling

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
        /*
        <applet code="ButtonDemo" width=250 height=150>
        </applet>
        */
public class ButtonDemo extends Applet implements ActionListener{
        String msg;
        Button b1,b2;
        public void init(){
                b1 = new Button("Yes");
                b2 = new Button("No");
                add(b1);
                add(b2);
                b1.addActionListener(this);
                b2.addActionListener(this);
        }
        public void actionPerformed(ActionEvent ae){
                String str = ae.getActionCommand();
                if(str.equals("Yes")){
                        msg = "You pressed Yes.";
                }else {
                        msg = "You pressed No.";
```

1) Event handling package

2) Event Listener Interface

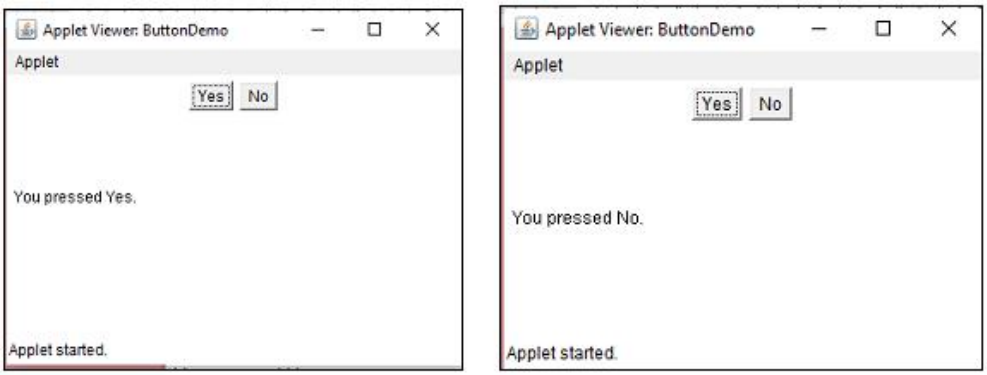
3) Registration method of ActionListener for button b1, b2(to inform we are going to click on b1, b2)

4) Method for button event handling from ActionListener Interface

IJARSCT
ISSN (Online) 2581-9429
ISSN (Print) 2581-XXXX
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 1, Issue 1, January 2021
Impact Factor: 4.819

```
            }
            repaint();
        }
        public void paint(Graphics g){
            g.drawString(msg, 6, 100);
        }
}
```

Output:



In this program, we have created 2 buttons using AWT Button class. For creating this output we need to import 2 packages i.e. java.awt, java.applet. For using event handling java.awt.event this package is used. We are creating applet window, so we should extend Applet class and implement Action Listener for handling Button related events. We have created 2 buttons using label "Yes" and "No". Then add both the buttons on Applet window using add command. Then registration method addActionListener() is called for both the buttons for b1, b2 to inform source i.e. Button, that user is going to click on these buttons. Then actionPerformed method is defined for handling button related events i.e. Clicking on button. If we click yes button it is displaying message "You clicked yes", otherwise "You clicked No". These messages are displayed using pain and drawString() method. In this example we used ActionListener Interface.

The ActionListener Interface

This interface defines the actionPerformed() method that is invoked when an action event occurs. Its general form is shown here:

void actionPerformed(ActionEvent ae)

In this paper we have seen example of button control only. There are many AWT controls. We can use event handling for all these controls. Following is the table for all AWT controls with their respective source, EventListener, Event handling method.

Source	Event(action)	EventListener	Event handling method	Event Class
Button	Clicking on button	ActionListener	actionPerformed( )	ActionEvent
Checkbox	Selecting checkbox	ItemListener	itemStateChanged()	ItemEvent
Choice	Selecting options from choice	ItemListener	itemStateChanged()	ItemEvent
CheckboxGroup	Select checkbox from given option	ItemListener	itemStateChanged()	ItemEvent
List	Selecting option from list	ActionListener	actionPerformed( )	ActionEvent
TextField	Enter value in textfield	ActionListener	actionPerformed( )	ActionEvent
Scrollbar	Scrolling scrollbar	AdjustmentListener	adjustmentValueChanged()	AdjustmentEvent

IJARSCT
ISSN (Online) 2581-9429
ISSN (Print) 2581-XXXX
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 1, Issue 1, January 2021
Impact Factor: 4.819

III. CONCLUSION

Using Event delegation model we can process events effectively and design more interactive applications in java. Only we should follow proper steps and use correct interface, methods for each control to execute all the events.

REFERENCES

[1]. Complete Reference-Schildt, Herbert-Mcgraw Hill Education,New Delhi, ISBN:9789339212094