Window Listeners

When the user clicks the close-window button (or either of the two accompanying buttons), the JFrame fires an event known as a window event. A JFrame can use the method **setWindowListener** to register a window listener to respond to such window events. A window listener can be programmed to respond to a window event in any way you wish. Window events are objects of the class **WindowEvent**. A window listener is any class that satisfies the **WindowListener interface.**

If a class implements the WindowListener interface, it must have definitions for all seven of these method headings. If you do not need all of these methods, then you can define the ones you do not need to have empty bodies, like this:

Window Listener

- public void windowDeiconified(WindowEvent e)
  {}

The **WindowListener interface** and the **WindowEvent class** are in the package **java.awt.event**.

- **public void windowOpened(WindowEvent e)**

  Invoked when a window is opened.

- **public void windowClosing(WindowEvent e)**

  Invoked when a window is in the process of being closed. Clicking the close-window button causes an invocation of this method.

- **public void windowClosed(WindowEvent e)**

  Invoked when a window has been closed.

- **public void windowIconified(WindowEvent e)**

  Invoked when a window is iconified. When you click the minimize button in a JFrame, it is iconified.

- **public void windowDeiconified(WindowEvent e)**

  Invoked when a window is deiconified. When you activate a minimized window, it is deiconified.

- **public void windowActivated(WindowEvent e)**

  Invoked when a window is activated. When you click in a window, it becomes the activated window. Other actions can also activate a window.

- **public void windowDeactivated(WindowEvent e)**

  Invoked when a window is deactivated. When a window is activated, all other windows are deactivated. Other actions can also deactivate a window.

The WindowListener Interface

When the user clicks any of the three standard JFrame buttons (for closing the window, minimizing the window, and resizing the window), it generates a window event. Window events are sent to window listeners. In order to be a window listener, a class must implement the WindowListener interface.

The dispose Method

The class JFrame has a method named dispose that will eliminate the invoking JFrame without ending the program. When dispose is invoked, the resources consumed by the JFrame and its components are returned for reuse, so the JFrame is gone, but the program does not end (unless dispose eliminates all elements in the program, as in a one-window program). The method dispose is often used in a program with multiple windows to eliminate one window without ending the program.

Syntax

JFrame_Object.dispose();

//Sample Code - Demonstrating Window Listener

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class WindowListenerDemo1 extends JFrame
{
   public static final int WIDTH=600;
   public static final int HEIGHT=500;

   public static final int SMALLWIDTH=300;
```

```
public static final int SMALLHEIGHT=250;

//Inner Class
private class CheckonExit implements WindowListener
{
	public void windowOpened(WindowEvent e)
	{
		System.out.println("Window is opened");
	}

	public void windowClosing(WindowEvent e)
	{
		ConfirmWindow cwindow=new ConfirmWindow();
		cwindow.setVisible(true);
	}

	public void windowClosed(WindowEvent e)
	{
		System.out.println("Window is closed");
	}

	public void windowIconified(WindowEvent e)
	{
		System.out.println("Window is Iconified");
	}

	public void windowDeiconified(WindowEvent e)
	{
		System.out.println("Window is Deiconified");
	}

	public void windowActivated(WindowEvent e)
	{
		System.out.println("Window is Activated");
	}

	public void windowDeactivated(WindowEvent e)
	{
		System.out.println("Window is Deactivated");
	}

}
```

```
//Inner Class
private class ConfirmWindow extends JFrame implements ActionListener
{
	public ConfirmWindow()
	{
		setSize(SMALLWIDTH,SMALLHEIGHT);
		getContentPane().setBackground(Color.CYAN);
		setLayout(new BorderLayout());

		JLabel confirmLabel=new JLabel("Do you really want to exit");
		add(confirmLabel,BorderLayout.CENTER);

		JPanel ButtonPanel=new JPanel();
		ButtonPanel.setLayout(new FlowLayout());

		JButton yesButton=new JButton("Yes");
		yesButton.addActionListener(this);
		ButtonPanel.add(yesButton);

		JButton noButton=new JButton("No");
		noButton.addActionListener(this);
		ButtonPanel.add(noButton);

		add(ButtonPanel,BorderLayout.SOUTH);
	}

	public void actionPerformed(ActionEvent e)
	{
		String buttonString=e.getActionCommand();

		if(buttonString.equals("Yes"))
			System.exit(0);
		else if(buttonString.equals("No"))
			dispose(); //Destroys only the confirm window
		else
			System.out.println("Error");
	}

}


public static void main(String[] args)
{
```

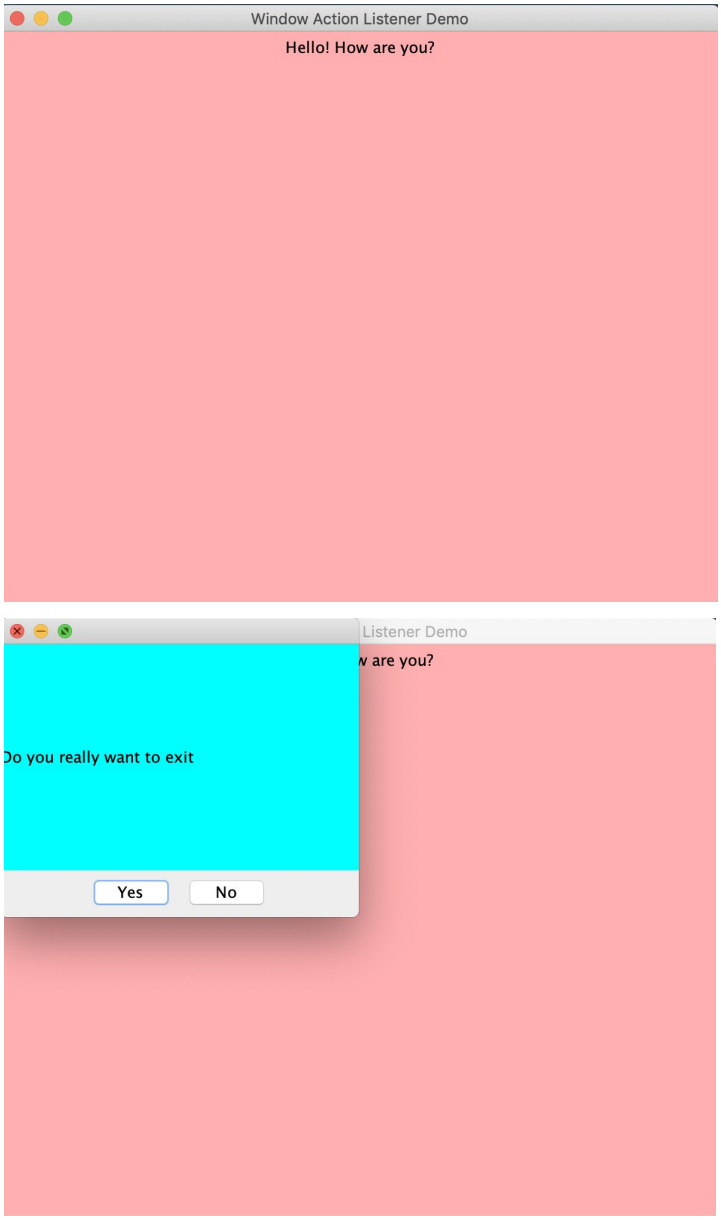
```
        WindowListenerDemo1 win=new WindowListenerDemo1();
        win.setVisible(true);
    }

    public WindowListenerDemo1()
    {
        super("Window Action Listener Demo");
        setSize(WIDTH,HEIGHT);
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        setLayout(new FlowLayout());
        addWindowListener(new CheckonExit());
        getContentPane().setBackground(Color.PINK);
        JLabel promptLabel=new JLabel("Hello! How are you?");
        add(promptLabel);

    }

}


//Sample Output
```




```
Window is Activated
Window is opened
Window is Deactivated
Window is Activated
Window is Deactivated
```

The WindowAdapter Class

If you make a window listener a derived class of the class **WindowAdapter**, then you have only to define the method headings in the **WindowListener** interface that you need. The other method headings inherit trivial implementations from WindowAdapter.
The class **WindowAdapter** is in the **java.awt.event** package and so requires an import statement.

//Sample Code - Window Adapter

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class WindowListenerDemo2 extends JFrame
{
	public static final int WIDTH=600;
	public static final int HEIGHT=500;

	public static final int SMALLWIDTH=300;
	public static final int SMALLHEIGHT=250;

	//Inner Class
	private class CheckonExit extends WindowAdapter
	{

		public void windowClosing(WindowEvent e)
		{
			ConfirmWindow cwindow=new ConfirmWindow();
			cwindow.setVisible(true);
		}

	}

	//Inner Class
	private class ConfirmWindow extends JFrame implements ActionListener
	{
		public ConfirmWindow()
		{
			setSize(SMALLWIDTH,SMALLHEIGHT);
			getContentPane().setBackground(Color.CYAN);
			setLayout(new BorderLayout());

			JLabel confirmLabel=new JLabel("Do you really want to exit");
			add(confirmLabel,BorderLayout.CENTER);
```

```
                JPanel ButtonPanel=new JPanel();
                ButtonPanel.setLayout(new FlowLayout());

                JButton yesButton=new JButton("Yes");
                yesButton.addActionListener(this);
                ButtonPanel.add(yesButton);

                JButton noButton=new JButton("No");
                noButton.addActionListener(this);
                ButtonPanel.add(noButton);

                add(ButtonPanel,BorderLayout.SOUTH);
        }

        public void actionPerformed(ActionEvent e)
        {
                String buttonString=e.getActionCommand();

                if(buttonString.equals("Yes"))
                        System.exit(0);
                else if(buttonString.equals("No"))
                        dispose(); //Destroys only the confirm window
                else
                        System.out.println("Error");
        }

}


public static void main(String[] args)
{
        WindowListenerDemo2 win=new WindowListenerDemo2();
        win.setVisible(true);
}

public WindowListenerDemo2()
{
        super("Window Action Listener Demo");
        setSize(WIDTH,HEIGHT);
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        setLayout(new FlowLayout());
        addWindowListener(new CheckonExit());
```

```
            getContentPane().setBackground(Color.PINK);
            JLabel promptLabel=new JLabel("Hello! How are you?");
            add(promptLabel);

        }

}
```