EVENT HANDLING

Naming Patterns for the Event Model

A *listener* is an object that is interested in being notified when a particular *event* takes place. The origin of this event is usually an object called the *source*, which notifies interested listeners when the event occurs. In this setup, a listener can be added to or removed from the list of listeners notified by a source about the occurrence of a particular event. This setup is the basis of the *event model* which is depicted in Figure 3.1. The JavaBean specification stipulates naming patterns for the event model to facilitate its use by builder tools to assemble event-based applications. Figure 3.1 shows where the naming patterns for handling events of type X are applied:

• An event class with the name XEvent, that extends the java.util.EventObject class.

```
public class XEvent extends java.util.EventObject {
        public XEvent(Object source) {
                super(source);
        }
}
```

• A listener interface with the name XListener, that specifies the specific method to be called in a listener when an event of the type XEvent occurs. The listener interface extends the java.util.EventListener interface.

```
public interface XListener extends java.util.EventListener {
        public void methodAInXListener(XEvent ev);
}
```

A listener interested in XEvents must implement the XListener interface, and must be registered with the source in order to be informed about XEvents.

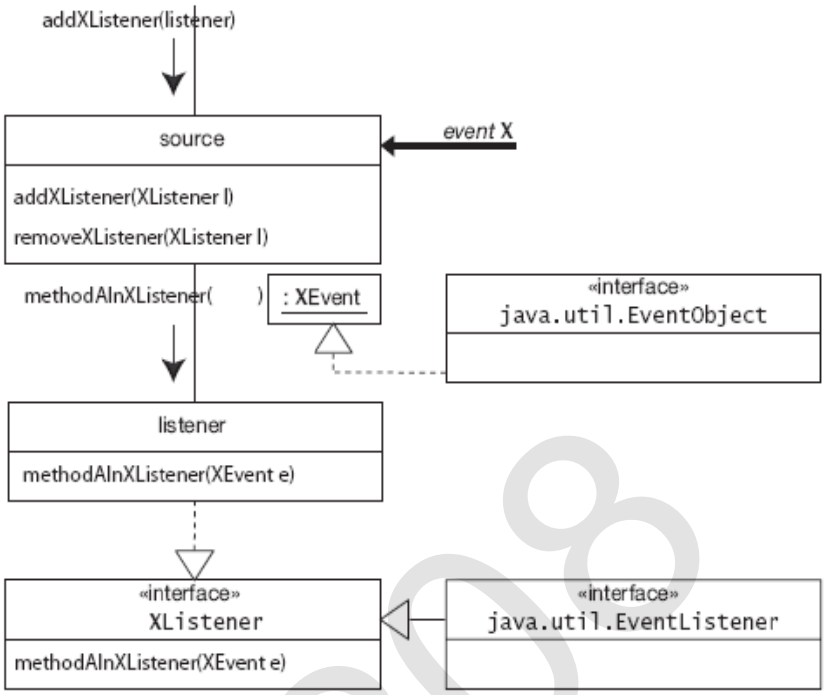
```
public class ListenerObject implements XListener {
        public void methodAInXListener(XEvent e) { /* ... */ }
}
```

• A source for XEvent, that implements the methods addXListener() and remove XListener(). These methods are used to add or remove a listener interested in XEvents, respectively. The parameter of these methods is of the type XListener.

```
public class SourceObject {
        public synchronized void addXListener(XListener listener) { /* ... */ }
        public synchronized void removeXListener(XListener listener) { /* ... */ }
}
```

Note that there are no naming patterns defined for the names of the source and the listener classes. Neither is there any standard convention for naming the methods specified in the listener interface.

Figure 3.1 *The Event Model*



A listener interested in XEvent events is registered with the source using the addXListener() method.
The listener must implement the XListener interface in order to recieve events of type XEvent.
The listener is informed about events of type XEvent via the methodAInXListener()
in the XListener interface.

METHOD1:

```
package vijay.java.eventlistener;
import java.awt.*;
import java.awt.event.*;

class MyEvent extends Frame implements ActionListener{
	TextField tf;
	Button b;
	MyEvent(){

		tf=new TextField();
		tf.setBounds(60,50,170,20);

		b=new Button("click me");
		b.setBounds(100,120,80,30);


		b.addActionListener(this);

		add(b);
		add(tf);
		setSize(300,300);
		setLayout(null);
		setVisible(true);

	}

	public void actionPerformed(ActionEvent e){
		tf.setText("Welcome");
	}

	public static void main(String args[]){
		new MyEvent();
	}
}
```

METHOD 2:

```
package vijay.java.eventlistener;

import java.awt.*;
import java.awt.event.*;

public class MyEvent2 extends Frame {
	TextField tf;
	Button b;
	public MyEvent2(){

		tf=new TextField();
		tf.setBounds(60,50,170,20);

		b=new Button("click me");
		b.setBounds(100,120,80,30);
		Outer o = new Outer(this);
		b.addActionListener(o);

		add(b);add(tf);

		setSize(300,300);
		setLayout(null);
		setVisible(true);

	}
	public static void main(String args[]){
		new MyEvent2();
	}
}
class Outer implements ActionListener{
	MyEvent2 obj;
	Outer(MyEvent2 obj){
		this.obj=obj;
	}

	public void actionPerformed(ActionEvent e){
		obj.tf.setText("welcome");
	}

}
```

METHOD:1

```
package vijay.java.eventlistener;

import java.awt.Button;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;


public class TwoButtonMyFrame extends Frame implements ActionListener {

	TextField tf;
	Button b1,b2;
	Label l;
	TwoButtonMyFrame(){
		l= new Label("Please Enter the Number:");
		l.setBounds(40, 40, 150, 20);
		add(l);
		tf = new TextField();
		tf.setBounds(230,40,200,20);
		add(tf);
		b1= new Button("OK");
		b1.setBounds(230,75,30,30);
		b1.addActionListener(this);
		add(b1);
		b2= new Button("CANCEL");
		b2.setBounds(280,75,50,30);
		b2.addActionListener(this);
		add(b2);
		setTitle("my frame ji");
		setLayout(null);
		setSize(400, 400);
		setVisible(true);
	}

	public static void main(String args[]){
		new TwoButtonMyFrame();
	}
	@Override
	public void actionPerformed(ActionEvent e) {
		// TODO Auto-generated method stub
		if(e.getSource()==b1)
			tf.setText("You have pressed OK");
		if(e.getSource()==b2)
			tf.setText("You have pressed CANCEL");


	}

}
```

METHOD2:

```
package vijay.java.eventlistener;

import java.awt.Button;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TwoButtonMyEvent extends Frame {

		TextField tf;
		Button b1,b2;
		Label l;
		TwoButtonMyEvent(){

			l= new Label("Please Enter the Number:");
			l.setBounds(40, 40, 150, 20);
			add(l);
			tf = new TextField();
			tf.setBounds(230,40,200,20);
			add(tf);
			b1= new Button("OK");
			b1.setBounds(230,75,30,30);
			add(b1);
			b2= new Button("CANCEL");
			b2.setBounds(280,75,50,30);
			add(b2);
			MyListener ml= new MyListener(this);
			b1.addActionListener(ml);
			b2.addActionListener(ml);
			setTitle("my frame ji");
			setLayout(null);
			setSize(400, 400);
			setVisible(true);
		}

		public static void main(String args[]){
			new TwoButtonMyEvent();
		}
}

class MyListener implements ActionListener{
	TwoButtonMyEvent frame;
	MyListener(TwoButtonMyEvent frame){
		this.frame= frame;
	}
		@Override
		public void actionPerformed(ActionEvent e) {
			// TODO Auto-generated method stub
			if(e.getSource()==frame.b1)
				frame.tf.setText("You have pressed OK");
			if(e.getSource()==frame.b2)
				frame.tf.setText("You have pressed CANCEL");

		}

	}
```