Richard G Baldwin (512) 223-4758, baldwin@austin.cc.tx.us, http://www2.austin.cc.tx.us/baldwin/

Event Handling in JDK 1.1, Component Events

Java Programming, Lecture Notes # 97, Revised 02/24/99.

- Preface
- Introduction
- Overview
- Sample Program
  - Interesting Code Fragments
  - Program Listing

Preface

Students in Prof. Baldwin's **Intermediate Java Programming** classes at ACC will be responsible for knowing and understanding all of the material in this lesson beginning with the Spring semester of 1999.

Sample program was successfully tested using JDK 1.2 under Win95 on 2/24/99.

Introduction

This lesson was originally written on September 22, 1998, using the JDK 1.1.6 download package. The purpose of this lesson is to illustrate the use of **component** events.

Overview

If you instantiate an object of type **ComponentListener** and register that object on an object that is a subclass of the **Component** class, methods of the listener object will be invoked whenever the object is *hidden*, *moved*, *resized*, or *shown*.

Hiding and showing are accomplished by invoking the **setVisible()** method on the object with a parameter of *false* or *true*.

Resizing and moving are accomplished by physically performing these actions on the visual representation of the object on the screen.

Information regarding the event is passed into the **ComponentListener** methods in the form of an object of type **ComponentEvent**.

Sample Program

This program illustrates component events, and invokes the **componentShown()** and **componentHidden()** methods of the **ComponentListener** interface under program control.

The user can invoke the **componentMoved()** and **componentResized()** methods by moving and resizing the **Frame** object on the screen.

The program also illustrates the use of an anonymous inner class to terminate the program when the user clicks the *close* button on the **Frame** object..

It also illustrates the use of the controlling class as a listener class. In this situation, the listener methods are defined as methods of the controlling class rather than being defined as methods of separately compiled listener classes.

Typical output from the program is shown in the program listing that follows later.

This program was tested using JDK 1.1.6 under Win95.

Interesting Code Fragments

We will begin with the first line of the definition of the controlling class. Note that the controlling class implements the **ComponentListener** interface. Having done this, the listener methods declared in that interface must be defined in the controlling class, and an object of the controlling class (**this**) is the listener object which gets registered on itself (**this**).

```
class Event34 extends Frame implements ComponentListener{
```

I will skip the **main** method (that does nothing but instantiate an instance of the controlling class) and move on to the constructor. I will highlight only those statements in the constructor that are interesting with respect to component events or the structure of the program.

The next fragment shows the code to register the object as a listener on itself. This is a somewhat cryptic programming style that I have not used widely in these tutorial lessons up to this point but you do see it being used by other authors.

```
    this.addComponentListener(this);
```

The next fragment shows statements that cause the **componentShown()** and **componentHidden()** methods of the **ComponentListener** interface to be invoked several times in succession.

```
    this.setVisible(true);//invoke shown event
    this.setVisible(false);//invoke hidden event
    this.setVisible(true);//invoke shown event
```

To invoke the other two methods of the interface, you will need to manually resize and move the **Frame** object on the screen.

The last fragment shows the definition of one of the methods of the **ComponentListener** interface. The definitions of all four methods are essentially the same, so only one of them is shown here.

```
  public void componentResized(ComponentEvent e){
    System.out.println("Resized\n" + e.getSource());
  }//end componentResized()
```

The code in the program that was not highlighted in the fragments above can be viewed in the complete listing of the program that follows in the next section.

Program Listing

```
/*File Event34.java
Copyright 1997, R.G.Baldwin

This program illustrates component events, and invokes the
componentShown() and componentHidden() methods of the
ComponentListener interface under program control.

The user can invoke the componentMoved() and
componentResized() methods by moving and resizing the
Frame object.

The program also illustrates the use of an anonymous
inner class.

It also illustrates the use of the controlling class as a
listener class.  In this situation, the listener methods
are defined as methods of the controlling class rather than
being defined as methods of separately compiled listener
classes.

Typical output from the program with line breaks manually
inserted is shown below:
```

```
Shown
Event34[frame0,0,0,350x100,layout=java.awt.BorderLayout,
resizable,title=Copyright 1998 R.G.Baldwin]
Hidden
Event34[frame0,0,0,350x100,layout=java.awt.BorderLayout,
resizable,title=Copyright 1998 R.G.Baldwin]
Shown
Event34[frame0,0,0,350x100,layout=java.awt.BorderLayout,
resizable,title=Copyright 1998 R.G.Baldwin]
Moved
Event34[frame0,31,43,350x100,layout=java.awt.BorderLayout,
resizable,title=Copyright 1998 R.G.Baldwin]
Resized
Event34[frame0,31,43,282x139,layout=java.awt.BorderLayout,
resizable,title=Copyright 1998 R.G.Baldwin]


This program was tested using JDK 1.1.6 under Win95.
**********************************************************/
import java.awt.*;
import java.awt.event.*;

//Note that the controlling class implements the
// ComponentListener interface.
class Event34 extends Frame implements ComponentListener{

  public static void main(String[] args){
    new Event34();//instantiate this object
   }//end main
  //=====================================================//

  public Event34(){//constructor
    //Add a component listener
    this.addComponentListener(this);
    this.setSize(350,100);
    this.setTitle("Copyright 1998 R.G.Baldwin");
    this.setVisible(true);//invoke shown event
    this.setVisible(false);//invoke hidden event
    this.setVisible(true);//invoke shown event

    //Anonymous inner-class listener to terminate program
    this.addWindowListener(
      new WindowAdapter(){//anonymous class definition
        public void windowClosing(WindowEvent e){
          System.exit(0);//terminate the program
        }//end windowClosing()
      }//end WindowAdapter
    );//end addWindowListener
  }//end constructor
  //-----------------------------------------------------//

  //Define the methods of the ComponentListener interface
  public void componentResized(ComponentEvent e){
    System.out.println("Resized\n" + e.getSource());
  }//end componentResized()
  public void componentMoved(ComponentEvent e){
    System.out.println("Moved\n" + e.getSource());
  }//end componentMoved()

  public void componentShown(ComponentEvent e){
    System.out.println("Shown\n" + e.getSource());
  }//end componentShown()

  public void componentHidden(ComponentEvent e){
    System.out.println("Hidden\n" + e.getSource());
  }//end componentHidden()

}//end class Event34
//=========================================================//
```

-end-