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

Event Handling in JDK 1.1, Text Events

Java Programming, Lecture Notes # 105, Revised 02/24/99.

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.

On 2/24/99, it was confirmed that the sample program in this lesson operates properly under JDK 1.2.

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 **text** events.

Overview

If you instantiate an object of type **TextListener** and register that object on an object that has an **addTextListener()** method, the **textValueChanged()** method of the listener object will be invoked whenever the text contents of the source object changes.

A text event is a semantic event, and several different sources can multicast the event. I will explain the use of the **TextListener** interface and **TextEvent** class using a simple **TextField** object..

Information regarding the event is passed into the **textValueChanged()** method in the form of an object of type **TextEvent**.

Sample Program

This program places a **TextField** object in a **Frame** object and registers a **TextListener** object on the **TextField**.

Whenever the text contents of the **TextField** change, a **TextEvent** is generated causing the **textValueChanged()** method of the listener object to be invoked.

Code in the **textValueChanged()** method extracts the source of the event from the incoming **TextEvent** object, and uses that source information to get and display the current text contents of the **TextField** object.

You can exercise the program by typing into the **TextField**.

Tested using JDK1.1.6 under Win95.

Interesting Code Fragments

We will begin with the first line of the definition of the controlling class, which simply shows that this class extends **Frame** and implements **TextListener**. Because it implements **TextListener**, an object of the controlling class is a listener object.

```
public class Event36 extends Frame implements TextListener{
```

I will skip the **main()** method that simply instantiates an object of the controlling class (which is a **TextListener** object).

The next fragment shows that part of the constructor that instantiates the **TextField** object, registers the listener (**this**) on it, and places it in the **Frame** object.

```
  Event36(){//constructor
    TextField myTextField =
                        new TextField("Initial String",30);
    myTextField.addTextListener(this);
    this.add(myTextField);
```

After this, I perform the necessary chores associated with the **Frame** parameters. That code can be viewed in the program listing later, and is not shown here.

That brings me to the **textValueChanged()** method of the **TextListener** interface that responds to text events, and displays the contents of the **TextField** every time its value changes.

```
  public void textValueChanged(TextEvent e){
    System.out.println(
                    ((TextField)e.getSource()).getText());
  }//end TextValueChanged()

}//end class Event36 definition
```

That is the end of this simple program.

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

Program Listing

This section contains a listing of the program.

```
/*File Event36.java, Copyright 1998, R.G.Baldwin
Illustrates Text events.

This program puts a TextField object in a Frame object and
registers a TextListener object on the TextField.  Whenever
the text contents of the TextField change, a TextEvent is
generated causing the textValueChanged() method of the
listener object to be invoked.

Code in the method extracts the source of the event from
the incoming TextEvent object, and uses that information
to get and display the current text contents of the
TextField object.

Tested using JDK1.1.6 under Win95.
**********************************************************/

import java.awt.*;
import java.awt.event.*;
import java.util.*;
//========================================================//
public class Event36 extends Frame implements TextListener{
  public static void main(String[] args){
    new Event36();
  }//end main
```

```
  //-------------------------------------------------//

  Event36(){//constructor
    TextField myTextField =
                         new TextField("Initial String",30);
    myTextField.addTextListener(this);
    this.add(myTextField);

    //Adjust Frame parameters and make it visible
    this.setLayout(new FlowLayout());
    this.setSize(350,100);
    this.setTitle("Copyright 1998, R.G.Baldwin");
    this.setVisible(true);

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

  //Define the method of the TextListener interface
  public void textValueChanged(TextEvent e){
    System.out.println(
                     ((TextField)e.getSource()).getText());
  }//end TextValueChanged()

}//end class Event36 definition
//=================================================//
```

-end-