CS 402

LECTURE ON EVENT HANDLING FOR JAVA

By George Koutsogiannakis

1. Different ways of creating an event
2. Events are handled by a number of different ways. No matter how we structure the event handler the following items need to be completed:
   1. Declare the object that it is the source of the event
   2. Identify the Listener Interface for that type of event source object
   3. Create an object of the class that acts as the event handler class. This means that this class has to implement the Listener Interface.
   4. Register the source object with the event listener.
3. Here is a simple example of event handling. In this example the source is a button, the Listener Interface is the ActionListener interface and the event is to exit the program

import javax.swing.\*;

import java.awt.event.\*;

public class GenerateEvent

{

public static void main (String [] args)

{

…………………

// various lines of code

…………………….

JButton mybutton=new JButton(“Exit”);

//registration and implementation using anonymous inner class

mybutton.addActionListener( new ActionListener() {

public void actionPerformed (ActionEvent e)

{

System.exit(0);

}

} //end anonymous inner class

); // end call to ActionListener

// other lines of code

}

}

An alternative way is to create an independent inner class for the event handler

import javax.swing.\*;

import java.awt.event.\*;

public class GenerateEvent

{

public static void main (String [] args)

{

…………………

// various lines of code

…………………….

JButton mybutton=new JButton(“Exit”);

EventHandler eh= new EventHandler();

//registration

Mybuton.addActionListener(eh);

// other lines of code

} //end of main

class Event handler implements ActionListener

{

Public void ActionPerformed(ActionEvent e)

{

System.exit(0);

}

} //end of inner class

}// end of outer class

Another way is to do the following

import javax.swing.\*;

import java.awt.event.\*;

public class GenerateEvent implements ActionListener

{

public static void main (String [] args)

{

…………………

// various lines of code

…………………….

JButton mybutton=new JButton(“Exit”);

//registration

Mybutton.addActionListener(this);

//other lines of code

} // end of main method

public void actionPerformed(ActionEvent e)

{

System.exit(0);

}

} // end of class

A fourth way is to create two outer classes. One is our program that needs the event handler and the other is the outer class for the event handler

import javax.swing.\*;

import java.awt.event.\*;

import packageName.MyEventHandler;

public class GenerateEvent implements ActionListener

{

public static void main (String [] args)

{

…………………

// various lines of code

…………………….

JButton mybutton=new JButton(“Exit”);

//registration

MyEventHandler meh=new MyEventHandler();

Mybutton.addActionListener(meh);

//other lines of code

}

}

Now we create the outer class for the event handler and compile it in a different file

package packageName;

import java.awt.event.\*;

public class MyEventHandler implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

System.exit(0);

}

} // end of class

1. Updating the GUI

Sometimes we need to update the GUI after we have presented the GUI to the user. That means that we need to repaint. Since every container has a paint method that gets called automatically whenever the container object is created or when the mouse is clicked or whenever we want to generate a paint event, we can always override the paint method to accomplish this. Remember that when you create a JFrame for instance, the paint method will be called automatically. If you want the paint method to be called again you can call repaint() which automatically calls the update() method and which then calls the paint method. A container like JPanel has a paint method which is called paintComponent

Therefore:

public void paint (Graphics g)

{

}

has to be overridden (or public void paintComponent in the case of JPanel)

Here is an example where the paint method is overridden

//Author: George Koutsogiannakis

//Date: 8/31/02

import javax.swing.JOptionPane;

import javax.swing.JApplet;

import javax.swing.\*;

import java.text.DecimalFormat;

import java.awt.Graphics;

import java.awt.\*;

import java.awt.event.\*;

public class Mortgagecalculation extends JApplet implements Runnable

{

String interest;

double interestnumber;

String time;

String loanamount;

String result;

String string;

double timenumber;

double decimalinterestnumber;

double loannumber;

double payment;

double z;

double x;

double a;

double y;

double totalpayments;

double totalinterest;

DecimalFormat twodigits = new DecimalFormat("0.00");

public void init()

{

}

public void start ()

{

Mortgagecalculation m=new Mortgagecalculation();

//2=information message

interest = JOptionPane.showInputDialog (m, "Enter the annual interest rate with up to three decimal places", " REQUEST FROM USER", 2);

interestnumber = Double.parseDouble(interest);

time= JOptionPane.showInputDialog (this, "Enter the time in months");

timenumber = Double.parseDouble(time);

loanamount=JOptionPane.showInputDialog (m, "Enter the amount of the loan");

loannumber=Double.parseDouble(loanamount);

decimalinterestnumber=interestnumber\*0.01;

System.out.println(decimalinterestnumber);

z= decimalinterestnumber/12;

a= (1+z);

System.out.println(a);

x= Math.pow( a, timenumber);

System.out.println(timenumber);

System.out.println(x);

y= 1- (1/x);

payment= (loannumber\*z) / y;

System.out.println(payment);

//Format the payment number two a two digit format to the right of the decimal point.

//Class decimalFormat belongs to package java.text

result="Your montly payment is:"+" "+"$"+" "+twodigits.format(payment);

JOptionPane.showMessageDialog(this, result , "MONTLY PAYMENT" , JOptionPane.INFORMATION\_MESSAGE);

Container c= getContentPane();

// Create a Canvas

Canvas canvas =new Canvas();

//canvas.setBackground(Color.red);

//canvas.setSize(200,200);

c.setLayout(new BorderLayout(5,5));

c.add (canvas, BorderLayout.NORTH);

//Create a text Area to output more information in string

string="If you would like to exit this program click on the exit button,\n"+

"if you would like to enter new numbers clickon the continue button";

JTextArea textarea = new JTextArea(string, 25, 15);

//USE "BOX LAYOUT MANAGER" TO ARRANGE COMPONENTS LEFT TO RIGHT WITHIN THE CONTAINER.

//THE CONTAINER IS THE TEXT AREA IN THIS CASE.

Box box =Box.createHorizontalBox();

box.add(new JScrollPane (textarea));

JButton exit=new JButton ("Exit");

box.add(exit);

JButton contin =new JButton ("Continue");

box.add(contin);

c.add(box, BorderLayout.SOUTH);

exit.addActionListener( new ActionListener() {

public void actionPerformed (ActionEvent e)

{

destroy();

}

} //end anonymous inner class

); // end call to ActionListener

contin.addActionListener( new ActionListener() {

public void actionPerformed (ActionEvent e)

{

start();

}

} //end anonymous inner class

); // end call to ActionListener

repaint();

}

public void destroy ()

{

System.exit(0);

}

public void run()

{

result="This is your montly mortgage payment:"+" "+payment;

JOptionPane.showMessageDialog(this, result , "MONTLY PAYMENT" , JOptionPane.INFORMATION\_MESSAGE);

}

public void paint (Graphics g)

{

super.paint(g);

totalpayments=timenumber\*payment;

totalinterest = totalpayments-loannumber;

g.drawString("The total payments made are:"+" " + twodigits.format(totalpayments) , 25, 25);

g.drawString("The total interest paid is:"+" " + twodigits.format(totalinterest) , 25, 50);

}

}

Notice that the exit button is not going to work with the Browser because of the security sandbox that a Browser presents with respect to system calls. It would had worked with a JFrame.

Multiple Eventhandlers can exist for one source and vice versa (multiple sources can use one Eenthandler).

Here is another example:

//created by George Koutsogiannakis on 2/25/2004

import javax.swing.JPanel;

import javax.swing.JFrame;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.JButton;

public class MyCanvas extends JFrame

{

int index=1;

Canvasexample canvas=new Canvasexample();

public MyCanvas() {

super("A CANVAS EXAMPLE");

Container c=getContentPane();

JPanel panel=new JPanel();

canvas.setBackground(Color.red);

canvas.setSize(200,200);

c.add(canvas, BorderLayout.SOUTH);

panel.setSize(100,100);

JButton button=new JButton("Activate Graphics");

JButton exitbutton=new JButton("Exit");

panel.add(button);

panel.add(exitbutton);

c.add (panel, BorderLayout.NORTH);

c.setSize(500,600);

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

Handler handle=new Handler();

button.addActionListener(handle);

ExitHandler exit=new ExitHandler();

exitbutton.addActionListener(exit);

}

public static void main(String[] args)

{

MyCanvas canvasex= new MyCanvas();

canvasex.pack();

canvasex.setVisible(true);

}

class Canvasexample extends Canvas

{

public void paint(Graphics g)

{

super.paint(g);

if (index==1)

{

g.setColor(new Color(255,255,255));

g.drawRect(10, 10, 100, 100);

}

else {

g.setColor(new Color(0,255,0));

g.drawLine(10,10, 100, 100);}

}

}

class Handler implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

index=2;

canvas.repaint();

}

}

class ExitHandler implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

System.exit(0);

}

}

}

1. Animation

To create animation we need to do the following:

* 1. Create an MediaTracker object
  2. Load the image (s)
  3. Register the images with the MediaTracker object
  4. Create a Thread class to handle the animation

Within the run method of the thread sleep the thread for some time and when the thread wakes up call the paint method by using repaint() call.

* 1. Override the paint method to accomplish the drawing of the image

Example:

mport java.applet.Applet;

import java.awt.\*;

//

//

// MediaTrackerDemo

//

//

public class MediaTrackerDemo extends Applet implements Runnable

{

Image[] imgArray = null;

MediaTracker tracker = null;

int current = 0;

Thread animThread=null;

// Check for a mouse click, to start the images downloading

public boolean mouseDown(Event evt, int x, int y)

{

if (tracker == null)

{

// Create a new media tracker, to track loading images

tracker = new MediaTracker(this);

// Create an array of three images

imgArray = new Image[3];

// Start downloading the images

for (int index=0; index < 3; index++)

{

// Load the image

imgArray[index] = getImage( getDocumentBase(),

"anim" + (index+1) + ".gif");

// Register it with media tracker

tracker.addImage(imgArray[index], index);

}

// Start animation thread

animThread = new Thread(this);

animThread.start();

}

return true;

}

public void paint (Graphics g)

{

g.setColor(Color.white);

g.fillRect(0,0, 200, 200);

g.setColor(Color.black);

// Check to see if images have started loading

if (tracker == null)

{

g.drawString ("Click to start loading",20,20);

}

else

// Check to see if images have loaded

if (tracker.checkAll())

{

g.drawImage(imgArray[current++], 0, 0, this);

if (current >= imgArray.length) current=0;

}

else

// Still loading

{

g.drawString ("Images are loading...", 20,20);

}

}

public void run()

{

try

{

tracker.waitForAll();

while(true)

{

// Repaint the images

repaint();

Thread.sleep(2000);

}

}

catch (InterruptedException ie) {};

}

}