# Easy Event-Handling: Language       v.3

## Dr. William C. Jones, Jr.             www.javabook.org

The individual discussions here are links from the main file. Each is numbered with the section that contains the link to it. Section 0. Java prerequisites You need to know a little about programming in Java before you study this material; the necessary knowledge is reviewed here. For one thing, you need to know that Java has a class of objects named String. A String constant is indicated by quote marks, as in String s = "Hello World!".

If you want to test whether two String values s and t are equal in terms of the characters they contain, you see whether the expression s.equals(t) is true or false. You do not test s==t; the == symbol is mainly for use with numeric values.

Some methods have parameters, that is, values that give the information the object needs to carry out the assigned task. You will not need to know how to define methods with parameters or return values, but you need to know how to use existing ones.

In a method call, parameters are written inside the parentheses that follow the name of the method. For instance, you will see in this material statements such as the following:

someObject.setText("you're wrong");

This displays the string of characters "you're wrong" (without quotes) on the object. You can retrieve the string currently displayed if you use the expression someObject.getText(). Parameters can also be numbers, as in someObject.width(8) or true/false values, as in setVisible(true) .

Say you have a class of objects named e.g. Turtle that you have written or someone has written for you (such as classes in the Sun library). Say that this class allows you to create a Turtle object using sam = new Turtle(). Then you can create a subclass of it named MyClass as follows:

**public class MyClass extends Turtle  
 {  
 public void doMyOwnMethod()**  
 { // your coding goes here  
 }  
 }

That is, you make sure that your class says extends Turtle (or whatever name is provided) and you define your own instance methods inside your class. Now you can create a MyClass object using sue = new MyClass(), and you can tell it to carry out the instructions in doMyOwnMethod using sue.doMyOwnMethod(). If you only need the MyClass object to execute just one command, you can skip storing the object in a variable; just use a statement such as new MyClass().doMyOwnMethod().

A class meant to be compiled in a separate file is generally declared as public, so it can be used by other classes. Methods belonging to a class are usually declared as public, so they can be used by outside classes. They are sometimes declared as private, when they are not for outside classes to use. Variables belonging to a class are usually declared as private.

You also need to be able to understand and use an if-statement, such as:

**if** (this.getText().equals ("900"))  
 { this.setText ("good!");  
 }  
 **else**  
 { this.setText ("wrong");  
 }

The left and right braces { } have to be nested properly (left matched to right). In more complex situations, you may have one or more additional alternatives. For instance, you could add the following phrase immediately before the else in the above statement:

**else if** (this.getText().equals ("899")) // part of the above  
 { this.setText ("close!");  
 }

In other cases, you may have several of these else-if phrases. In simpler cases, you may omit all else-phrases of the if-statement. The following is a complete statement:

**if** (input.length() > 0)  
 { modelAccount.makeChange (input);  
 itsOutput.say (modelAccount.currentBalance());  
 }

If you have only one statement inside the braces in one of these phrases, you are permitted to omit the braces around that statement.

### Section 1. Running GeogQuiz (and other EPanel subclasses) in applets and BlueJ

If you want to create a webpage that contains an applet for GeogQuiz, you first write the applet, as follows.

**public class GeogQuizApplet extends javax.swing.JApplet  
 {  
 public void start()**  
 { setContentPane (new GeogQuiz());  
 } //======================  
 }

This puts a GeogQuiz object, the panel that has the content of the applet, inside the applet area. In the webpage itself, you put the following HTML code (a very simple webpage that has nothing but this applet can consist of only these two lines):

<APPLET CODE="GeogQuizApplet.class"   
 WIDTH=700 HEIGHT=600> </APPLET>

Now if you compile the GeogQuizApplet class along with the GeogQuiz class and the various E-object classes, the webpage will display the same thing that the application frame displays. Note that you are required to set the size of the applet in the HTML coding. By contrast, an application program invokes the EFrame constructor, as in the following class. That constructor contains coding that set the size and calls setContentPane for the GeogQuiz object.

**class GeogQuizApp  
 {  
 public static void main (String[] args)**  
 { new EFrame (new GeogQuiz());  
 } //======================  
 }

You may of course choose a different size. The one shown makes the applet 700 pixels wide and 600 pixels tall. That is only somewhat smaller than a standard 14-inch monitor's screen.

To run the GeogQuiz program from BlueJ, you need to do the following, which is implicit in the single new EFrame (new GeogQuiz()) statement in the definition of GeogQuizApp:

1. Click on the GeogQuiz class box and select its constructor. This creates a GeogQuiz object, gives it the name GeogQuiz\_1, and puts a picture of it on the workbench.
2. Click on the EFrame class box and select its constructor. It will ask you for the parameter, which you enter as GeogQuiz\_1. This causes the EFrame to appear on your screen.

### Section 3. Using Color in Java programs

The Sun standard library contains the java.awt.Color class, which defines thirteen standard colors: black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, yellow, red, and white. Note that java.awt.Color.cyan is a greenish-blue and java.awt.Color.magenta is reddish-blue. You can also create your own color by supplying red/green/blue (RGB) values in the range from 0 to 255, inclusive, as in

new java.awt.Color(16, 255, 0) // green with a reddish tinge

The ELabel, EButton, and EField classes, as well as other component classes discussed later, are subclasses of Sun's JComponent, which has the following two methods. You can use the setBackground method to change the background color of the component from the standard white supplied for E-objects. And you can change the color of the characters (i.e., the foreground) displayed on a component with the setForeground method.

someComponent.setBackground (someColor);  
 someComponent.setForeground (someColor);

If you use the java.awt.Color class frequently in your coding, you can put at the top of your source code file a line that says import java.awt.\*; then you can just refer to the class as Color instead.

### Section 6. Discussion of the BankAccount class of objects

This part is not currently available.