**Easy Events�������������������������������������� v.3**

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

**Table of Contents** (click the link to jump to the part specified)

*Students should start with the first three parts in sequence:*

Part A�� 1. A first GUI program���� 2. Basic E-object components

[Part B](http://docs.google.com/EasyEvents_B.html)�� 3. Textfields������������������ 4. Instance variables������ 5. Basic String operations

[Part C](http://docs.google.com/EasyEvents_C.html)�� 6. Textareas����������������� 7. Timers

*The following parts can be done in any order:*

[Part D](http://docs.google.com/EasyEvents_D.html)�� 8. More useful features of E-objects�������� 9. Summary of E-object classes

Part E, Part F�� slots for future expansion in Nov 2003:� arrays, extended examples

**Introduction**

The Sun standard library gives Java programmers all the basic building blocks for making GUIs (Graphical User Interfaces). You use the Sun library classes to define more complex objects that work well for a particular purpose. This material describes a set of such definitions, suitable for the most common kinds of GUI. They are named EFrame, EPanel, ELabel, EButton, EField, ETextArea, and ETimer (the "E" stands for "Easy").

If you are not familiar with the building blocks used here, all the better! These E-objects are easier to understand and use than the building blocks they are made of. But studying these will take you most of the way towards understanding the building blocks themselves (JButton, ActionListener, etc.). A supplementary part of this material will give you a major start on the transition from E-objects to the more flexible Sun objects.

**LINKS**: This material is hyperlinked. It is intended to be studied on a computer. People who have previously studied a substantial amount of Java can read and understand this one main file without following any links. Others should follow the links as needed for finding the answers to exercises, seeing the sample programs actually execute, and reviewing Java language details.

You need to know a little about programming in Java before you study this material; the necessary knowledge is reviewed at this [language](http://docs.google.com/EElanguage.html#0) link. But you do not need to already know how to write your own constructors (we use only the simplest kind here). And you do not need to know how to write methods other than the kind that have the heading public void name() (in technical terms, no parameters and no return values).

If you have studied Java for less than two months, you should review the language links that appear from time to time. If you have had a fair amount of programming in some other language, even one as different from Java as Pascal or FORTRAN, the discussion at these language links is probably sufficient for you to understand everything presented here.

**1. A first GUI program**

Our first example is a program that asks two simple questions. The user is to answer each question by clicking on one of two buttons. The program's response will appear on the button where the user clicks. The frame is to look as follows when it first comes up for the user to see, except the user would see the rest of the photo. If you would like to see this **GeogQuiz** program in action, click this [execution](http://docs.google.com/EEdemos_AB.html) link.



When you solve a programming problem from scratch, you start by drawing pictures of how things will look. In this case, you would create the above drawing in Paint or sketch it on paper. Next you make a design for the solution. The first step of the solution is obviously to create a window or frame that will appear on the screen. If you run it from the command window, use java GeogQuizApp with the following application class.

**class GeogQuizApp // to create a frame from the command window**

{

**�� public static void main (String[ ] args)**

�� {� new EFrame (new GeogQuiz());

�� }� //======================

}

Now this is a bit complex, but it is not something you have to understand at this point. For other simple programs using EFrames, you always write the same application class as above except you change the name "GeogQuiz" in the two places. That is all there is to it. Just follow this formula.

Still, you probably want to know why it has to be this complex. The short answer is that the application program puts the GeogQuiz **panel** inside a **frame** that it creates. You can also run GeogQuiz in a webpage **applet**: Put setContentPage (new GeogQuiz()) in an applet's start method to cause the applet to contain the GeogQuiz panel. The complexity comes from separating the GeogQuiz object from the frame or applet it runs in. But this gives you more flexibility. We will refer to an EPanel used in this way as a GUI **program**. For detailed instructions on running the programs in this material from an applet or from BlueJ, click this [language](http://docs.google.com/EElanguage.html#1) link.

The accompanying block contains a reasonable design for the program, once the frame is created. See how easily the design follows from the picture. And once you have a design for a program, the Java coding follows easily almost line-for-line. The listing for the GeogQuiz class is below; we discuss the details in the following section. If you are the teacher of a class, you may want to read what is said at the [teachers](http://docs.google.com/EEteachers.html#1) link.

**STRUCTURED NATURAL LANGUAGE DESIGN for GeogQuiz creation**

1. Add to the frame the question "Where is Italy?"

2. Add to the frame a button that says "Europe" on it and says "right" when clicked.

3. Add to the frame a button that says "Asia" on it and says "wrong" when clicked.

4. Add to the frame the question "Where is India?"

5. Add to the frame a button that says "Europe" on it and says "wrong" when clicked.

6. Add to the frame a button that says "Asia" on it and says "right" when clicked.

7. Add to the frame a picture the user can contemplate while thinking out the answers.

8. Suspend action until the user clicks a button.

Listing 1 The GeogQuiz program

**public class GeogQuiz extends EPanel**

{

**�/\*\* The View:� Lay out the GUI components. \*/**

**�� public GeogQuiz()�������������������������������** �����������// 1

�� {� add (new ELabel ("Where is Italy?"));���������������� ���// 2

����� add (new RightButton().text ("Europe"));������������� ���// 3

����� add (new WrongButton().text ("Asia"));��������������� ���// 4

����� add (new ELabel ("Where is India?"));���������������� ���// 5

����� add (new WrongButton().text ("Europe"));������������� ���// 6

����� add (new RightButton().text ("Asia"));��������������� ���// 7

����� add (new ELabel("St. Peter's", "Rome.jpg")); //a photo ��// 8

�� }� //======================

**�/\*\* Controllers: React to click of button. \*/**

**�� private class RightButton extends EButton**��������� ���������// 9

�� {� public void onClick()��������������������������� ��������//10

����� {� this.setText ("right!");���������������������� �������//11

����� }������������������������������������������������� ������//12

�� }� //======================

**�� private class WrongButton extends EButton**������������� �����//13

�� {� public void onClick()������������������������������� ����//14

����� {� this.setText ("you're wrong.");������������������� ���//15

����� }�������������������������� �����������������������������//16

�� }� //======================

}

The part of the GeogQuiz class that begins public GeogView() and defines what components are added to the panel is called a **constructor**. It is different from a normal simple method in that it does not have the word void in its heading. It tells the computer how one constructs a GeogQuiz object (mostly, the components to add). After the GeogQuiz object is constructed, the runtime system pauses until some event takes place, such as a click of a button.

The "**this.**" phrase in this coding is optional, i.e., it can be omitted with any effect. It serves to remind you in lines 11 and 15 that the particular EButton object that the user clicked is the object to which the setText message is being sent. Likewise, we could replace "add" by "this.add" in lines 2-8, where "this" refers to the particular GeogQuiz object. In general, the use of "this" in a method refers to the object of the class containing the method.

**2. Basic E-object components**

An **EFrame** is for holding an EPanel. The expression **new EFrame(somePanel)** creates a new EFrame object with the given panel object for displaying its content. It is used for the main frame of an application, not an applet. The panel can also be a JPanel from the Sun standard library.

An **EPanel** is a rectangular area that can appear in a "window frame". A subclass of EPanel is to add components (buttons, pictures, etc.) as desired. If you study the complete GeogQuiz class, you can see that the lines numbered 2-8 in the constructor match up precisely with the first seven steps of the design. The key instance method for EPanels is **add(someComponent)** to place a component in the next available spot within the panel. It is used in the constructor:

�€€€€€€€€ Line 2 adds a **label** (i.e., a place on the screen) that contains the question "Where is Italy?".

�€€€€€€€€ Lines 3 and 4 add two buttons that contain the text "Europe" and "Asia". The first one will confirm the answer and the second will deny it.

�€€€€€€€€ Line 5 adds a label that contains the question "Where is Italy?".

�€€€€€€€€ Lines 6 and 7 add to the frame two buttons that contain the text "Europe" and "Asia". The first one will deny the answer and the second will confirm it.

�€€€€€€€€ Line 8 adds a label showing a picture of St. Peter's Cathedral in Rome.

The runtime system places all GUI (Graphical User Interface) components, such as labels and buttons, inside the frame in the order they are added. They are added left to right until one row is full, then added left to right in the next row, etc. Each row is centered. If you want more control over the layout, you have to use the Sun library.

One additional handy method for use with EPanels is, when adding components to a frame, you can force the next one to appear at the beginning of the following line of components if you add a **lineBreak()**. For instance, when the user does click-and-drag to resize the panel displaying GeogQuiz, the questions do not always stay on the same lines with their two buttons that have the answers. This is irritating. You can improve this by inserting add (lineBreak()) between lines 4 and 5 of the GeogQuiz constructor.

An **ELabel** denotes a place on the screen. The ELabel class has the following six instance methods and constructors. The first two are inherited from Sun's JLabel class.

�€€€€€€€€ **setText(someString)** changes the phrase written on the object to the string of characters in the parentheses.

�€€€€€€€€ **getText()** returns the phrase currently written on the object.

�€€€€€€€€ **setPicture(someString)** puts the picture from the specified file on the component.

�€€€€€€€€ **getPicture()** returns the string that names the file containing the current picture.

�€€€€€€€€ **new ELabel (someString)** creates the ELabel with someString written on it (its text).

�€€€€€€€€ **new ELabel (someString, fileString)** creates the ELabel with the given String written on it and displaying the image stored in the file named fileString.� Use the empty String "" for the text when you do not want any words to appear with the picture.

A subclass of **EButton** is generally defined inside the EPanel subclass. In the GeogQuiz class, RightButton and WrongButton are subclasses of EButton. EButton has the same setText and getText methods that ELabel has. EButton also has the text(someString) method, which does the same as setText but it returns the EButton itself, for convenience of use in add(someButton). You usually have just one method in a subclass of EButton: **public void onClick()** will execute when the user clicks the button.

The RightButton class is a subclass of EButton. It is defined here to display on itself the answer "right!" (line 11) when the user clicks it. The GeogQuiz class creates two objects from the RightButton class. One says "Europe" and displays "right!" when the user clicks it to answer the question "Where is France?". A different instance of the RightButton class says "Asia" and displays"right!" when the user clicks it to answer the question "Where is India?".

A WrongButton (another subclass of EButton) is defined to display on itself the answer "you're wrong." (line 15) when the user clicks it. The GeogQuiz class creates two instances of the WrongButton class. *Note that onClick() is not called by coding anywhere in the program; it is called by the user's action of clicking.*

You may be puzzled by the fact that the event-handling classes for buttons are placed inside the overall frame class. For this GeogQuiz class, they could just as well be placed outside the class (if you change "private class" to "class"). But in most situations, these event-handling classes need to be able to access private data fields of the EPanel; you will see this in programs discussed shortly. In such a case, the event-handling classes must be placed inside the EPanel class, to allow that access. That makes them **inner classes**. And since they are not of concern to outside classes, they should be declared as private members of the overall EPanel subclass. Note that "this" in the onClick and onEnter methods refers to "this" EButton or EField.

You probably thought it was weird that the answers for GeogQuiz appeared on the button instead of on some ELabel named e.g. sam. But if clicking a button showed "right" or "wrong" on sam, you would have to have sam.setText(..) inside your onClick methods. So sam would be declared as a private data field, which is a bit complicated. Still, if you really do not like the answers to appear on the button, you could instead have a little box pop up "out of nowhere" to display the answer using **popUp(someString)**. For instance, you could put popUp("right!") in place of the statement in line 11 to give the response.� Listing 2 illustrates this (click [execution](http://docs.google.com/EEdemos_AB.html#2) to see it).

Listing 2 The Popper program

**public class Popper extends EPanel**

{

**�� public Popper()�������������������������������** ������������// 1

�� {� add (new AnswerButton().text ("Where is this?"));������ // 2

����� add (new ELabel ("", "Rome.jpg")); //photo without text // 3

�� }� //======================

**�� private class AnswerButton extends EButton**���������������� // 4

�� {� public void onClick()���������������������������������� // 5

�� �� {� popUp ("Italy");������������������������������������ // 6

����� }������������������������������������������������������ // 7

�� }� //======================

}

You can view and work with the source code for the complete GeogQuiz application program by downloading the zipped folder that comes with this material. It contains both source code and compiled forms. To run it, enter java GeogQuizApp in the command window. Answers to exercises are at the exercise link, except for starred exercises.

[**Exercise**](http://docs.google.com/EEexercises.html) **2.1** If you changed Italy to China in line 2 of GeogQuiz and changed India to Peru in line 5, how would you change the rest of the program to make the buttons give right responses?

**Exercise 2.2** What would happen if you had the parameters for line 8 of GeogQuiz in the opposite order, writing new ELabel ("Rome.jpg", "St. Peter's") instead?

**Exercise 2.3** Change GeogQuiz so that it asks a third question: "Where is Egypt?" Have the two choices of Asia and Africa.

**Exercise 2.4\*** After working the previous exercise, change GeogQuiz so that each question has three possible answers, namely, Europe, Asia, and Africa.

**Exercise 2.5\*** Write a subclass of EPanel that has five true/false questions on one topic of your choosing, with either two or three of the questions having the answer true. Start a new row of components for each question.

**Exercise 2.6\*** Write a GUI program that has one button for each of the eight sample programs in parts A, B, C, D. �Clicking one button executes new EFrame (new GeogQuiz()), and similarly for the other seven examples.

Link to [next section](http://docs.google.com/EasyEvents_B.html) (Part B)