**8. More useful features of E-objects**

�

When you use one window to open other windows, you do not want all windows to disappear just because you close some subordinate window (which you can do with **someFrame.dispose()**). Only the main window should be created as an EFrame that terminates the whole program. The EFramelet class is provided for this purpose. If you use **new EFramelet(someEPanel)** instead of EFrame, you get a smaller frame, only 500 pixels wide and 300 pixels tall, and closing that window will not terminate the program. EFramelet is otherwise the same as EFrame, so you can use setSize and setTitle.

The next sample program is named DiceThrow. When DiceThrow is to be run as a main frame, use new EFrame (new DiceThrow()). But if you wanted, for instance, to let a user of the BankView frame take a break from work and run the DiceThrow, you would add a button to BankView whose onClick method says new EFramelet (new DiceThrow()). So clicking that button would make the dice game frame appear. But closing the DiceThrow frame would not lose the BankView frame as well.

Listing 8 displays two dice and two buttons. When the user clicks the first button, the dice start rolling fairly fast. When the user clicks the second button, the dice stop rolling. This program requires that you have six pictures named "1die.jpg", "2die.jpg", ..., "6die.jpg", showing the six possible die faces, in the same folder with the program that is using them. It uses a random number generator: You get one of 0, 1, 2, 3, 4, 5 from randy.nextInt(6) (line 13). In general, randy.nextInt(n) gives one of the n nonnegative numbers less than n. The accompanying figure shows how the frame could look after several clicks. Initially, though, both dice show a 1-spot. If you would like to see this DiceThrow program in action, click this [execution](http://docs.google.com/EEdemos_CD.html#8) link.



What makes the dice roll fast? Lines 13-16 put a random die number on the labels and then set their timers on delays of 0.15 and 0.09 seconds, respectively. Thereafter, every 0.12 seconds, each die's timer beeps, which executes the coding in the onBeep method. Why those numbers? Because experimentation showed they work well.

The ELabel method getPicture() returns the name of the current picture. The onBeep coding (line 19) gets the first character charAt(0) of the picture's name (which will be '1', '2', '3', '4', '5', or '6') and subtracts the Unicode value for '0' from the first character's Unicode value. That will produce one of the numbers 1, 2, 3, 4, 5, or 6. (One difference is that a character can be part of a string but a number cannot; a number can be doubled but a character cannot). The coding calculates the next number (lines 21-26; the next after 6 is 1). It then displays that picture and sets the delay for the next beeping to 0.12 seconds. Note: **n++** changes n to be 1 more than it was, i.e., it **increments** n; and **n--** changes n to be 1 less than it was, i.e., it **decrements** n.

Listing 8 The DiceThrow program

**public class DiceThrow extends EPanel**

{

�� private java.util.Random randy = new java.util.Random();��� // 1

�� private ETimer one = new RollTimer();���������������������� // 2

�� private ETimer two = new RollTimer();���������������������� // 3

�� private final int NUM\_SIDES = 6;��������������������������� // 4

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

**�� public DiceThrow()**����������������������������������������� // 5

�� {� add (new GoButton().text ("Click to roll the 2 dice"));� // 6

����� one.setPicture ("1die.jpg");���������������������������� // 7

����� two.setPicture ("1die.jpg");���������������������������� // 8

����� add (one, two);��� // binds them together��������������� // 9

����� add (new StopButton().text ("Click to stop"));���������� //10

�� }� //======================

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

**�� private class GoButton extends EButton**��������������������� //11

�� {� public void onClick()����������������������������������� //12

����� {� int n = 1 + randy.nextInt (NUM\_SIDES);��������������� //13

�������� one.delay (150).setPicture (n + "die.jpg");���������� //14

�������� n = 1 + randy.nextInt (NUM\_SIDES);������������������� //15

�������� two.delay (90).setPicture (n + "die.jpg");����������� //16

����� }��������������������������������������������� ����������//17

�� }� //======================

**�� private class RollTimer extends ETimer�������������������** ��//18

�� {� public void onBeep()������������������������������������ //19

����� {� int n = this.getPicture().charAt (0) - '0';���������� //20

�������� **if** (n >= NUM\_SIDES)���������������������������������� //21

�������� {� n = 1;�������������������������������������������� //22

�������� }���������������������������������������������������� //23

�������� **else**������������������������������������������������� //24

�������� {� n++;���������������� ������������������������������//25

�������� }���������������������������������������������������� //26

�������� this.delay (120).setPicture (n + "die.jpg");��������� //27

����� }������������������������������������������������������� //28

�� }� //======================

**�� private class StopButton extends EButton�����������������** ��//28

�� {� public void onClick()����������������������������������� //29

����� {� one.stop();������������������������������������������ //30

�������� two.stop();������������������������������������������ //31

����� }������������������������������������������������������� //32

�� }� //======================

}

**Additional features of EFrames**

The EFrame you get with new EFrame(someEPanel) (a) has a blank title on it, (b) has a width of 700 pixels, (c) has a height of 600 pixels, and (d) will terminate the program when the window is closed (i.e., revert to the command window if that is where you started it from). You may want to override these values.

The EFrame method **setSize(widthInt, heightInt)** lets you set the width and height of the frame, measured in pixels. And the EFrame method **setTitle (someString)** puts that string in the title bar of the frame. For instance, you can have a wide (520 pixels) but short (130 pixels) frame for the DiceThrow program, with the title "Dice Throw" at the top of the window, using the following main method (calling the **setVisible** method aligns the components with the borders).

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

{

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

�� {� EFrame frame = new EFrame (new DiceThrow());

����� frame.setSize (520, 130);

����� frame.setTitle ("Dice Throw");

����� frame.setVisible (true);

�� }� //======================

}

**Other occasionally useful features of E-objects**

The following two methods in the EPanel class are sometimes useful. Each one pops up a box that displays a message and gets input from the user. The names come from JavaScript.

�▪▪▪▪▪▪ **prompt(messageString)** displays the message and waits for the user to enter a string of characters. That string is returned to the calling statement. Example: String input = prompt("What is your name?") could obtain the user's name for later use.

�▪▪▪▪▪▪ **confirm(messageString)** displays the message and waits for the user to indicate yes or no. It returns true if the user indicates yes. Example: The test of� if (confirm ("play again?"))... sees if the user wants to play a game again.

The EButton class has the same setPicture and getPicture methods that ELabel has, as well as a convenience method **picture("fileName")**, analogous to text("textName"), that returns the EButton itself, so it can be used in, e.g., add (new XButton().picture("y.gif")).

Each of the classes of components has a **setToolTipText(someString)** method that displays an explanatory message when the cursor lingers over the component.

If you know a little **HTML**, you can change the format of the text displayed on EButtons and EFields.� For instance, you could replace line 6 of the DiceThrow program by the following, which underlines the word "Click" and breaks for a new line after "roll".

�� add (new GoButton().text

����� ("<html><u>Click</u> to roll<br> the 2 dice</html>");

[**Exercise**](http://docs.google.com/EEexercises.html#8) **8.1** Change DiceThrow to make the initial arrangement of dice two random choices, rather than always starting with "1die.jpg".

**Exercise 8.2** Change DiceThrow to have it stop rolling the dice after each has changed 15 times, even if the user has not clicked the button yet. Hint: Count to 30, then cancel delays.

**Exercise 8.3** How would you change DiceThrow to have the same effect if the pictures were named "die1.jpg" etc.?

**Exercise 8.4\*** Revise DiceThrow to roll three dice at a time instead of two.

**Exercise 8.5\*** A paragraph in this section describes how you could have the DiceThrow played from the BankView frame. Do what it says and try it out.

**Exercise 8.6\*** The DiceThrow has both dice increase the number by 1 each time. How would you have the second die decrease by 1 while the first die increases by 1?

**Exercise 8.7\*\*** After working Exercise 8.5, revise it to make the user stop playing the dice game after 40 seconds. Have a variable ETimer itsStop = new Stopper(), have a click of the button execute itsStop.delay(40000), and have the onBeep() method of the Stopper class execute the dispose method for the DiceThrow frame.

**9. Summary of E-object classes**

This section lists all E-object classes and methods described in this material. [R] means the method returns a value that is useful.

**Group 1**:� These are classes you normally do not make subclasses of.

**new EFrame (somePanel)** [subclass of Sun's JFrame] terminates the program when closed

**new EFramelet (somePanel)** [subclass of Sun's JFrame] does not terminate the program

����� x.dispose()������������������������� // close the frame

����� x.setTitle (titleString)����������� // set the title at the top of the frame

����� x.setSize (width, height)������ // make it width pixels wide and height pixels tall

����� x.setVisible (true)���������������� // make the frame appear on the screen;

����������������������������������������������� // �� only needed if modified after creating it

**new ELabel (textString)** [subclass of Sun's JLabel]

**new ELabel (textString, fileString)** put the text as a caption under the picture in the file

����� x.getText()�� [R]������������������ // return the String written on the component

����� x.setText (textString)����������� // change what the component says on it

����� x.getPicture()�� [R]�������������� // return the file name of the picture on it ("" if none)

����� x.setPicture (fileString)�������� // place the picture of that file name on the label

**new ETextArea (numberOfRows, numberOfColumns)** [subclass of Sun's JScrollPane]

����� x.say (stringAppended)�������� // add this string as the next line in the textarea

����� x.getText()�� [R]������������������ // return the String written on the component

����� x.setText (titleString)����������� // change what the component says on it; so "" clears it

**Group 2**:� You subclass an EPanel to add components and instance variables to it.� You subclass the other three listed here to react to events (the subclass defines onClick, onEnter, or onBeep).

**new EPanel()��** [subclass of Sun's JPanel]

����� x.add (someComponent) // insert the component in the panel, book-reading order

����� x.add (c1, c2)��������������������� // insert a panel containing components c1 and c2

����� x.add (c1, c2, c3)���������������� // insert a panel containing components c1, c2, and c3

����� x.lineBreak()����� [R]������������� // return a component that makes the start of a new row

����� x.popUp (messageString)���� // pop-up a separate window with a message

����� x.prompt (someString) [R]���� // pop-up a separate window with a message;

����������������������������������������������� // �� return a String input by the user

����� x.confirm (someString) [R]��� // pop-up a separate window with a request for

����������������������������������������������� //��� confirmation; return a boolean

**new EButton()��** [subclass of Sun's JButton]

����� x.getText()�� [R]������������������ // return the String written on the component

����� x.setText (textString)����������� // change what the component says on it

����� x.text (textString)�� [R]��������� // place the text on the component; return the component

����� x.getPicture()�� [R]�������������� // return the file name of the picture on it ("" if none)

����� x.setPicture (fileString)�������� // place the picture of that file name on the button

����� x.picture (fileString)������������� // place the picture of that name on the button; return the EButton

����� x.onClick()�������������������������� // respond to click if a subclass redefines it

����� x.setEnabled (boolean)�������� // enable or disable response to a click

**new EField()��** [subclass of Sun's JField]

����� x.getText()�� [R]������������������ // return the String written on the component

����� x.setText (textString)����������� // change what the component says on it

����� x.text (textString)�� [R]��������� // place the text on the component; return the component

����� x.onEnter()������������������������� // respond to ENTER key if a subclass redefines it

����� x.setEnabled (boolean)�������� // enable or disable response to ENTER key

����� x.width (numCharacters)� ���� // make it many characters wide; return the EField itself

**subclass of ETimer()��** [subclass of ELabel]

����� x.onBeep()������������������������� // action to be taken; it must be defined in each subclass

����� x.delay (milliseconds)� [R]��� // execute onBeep() after a delay; return the ETimer itself

����� x.stop()������������������������������ // terminate the timer, so onBeep() will not be executed

����� and all ELabel methods, by inheritance

**Group 3**:� These are classes that are not essential to basic GUI programming.� You normally do not make subclasses of these.

**new EOutputFile (fileString)** [subclass of Sun's PrintWriter]

����� x.println (messageString)����� // prepare the information for printing to the file

����� x.close()����������������������������� // print all information to the file and terminate the connection