**3. Textfields**

Our next example is a program that asks two simple questions. The user is to answer each question by typing the answer into a box provided for that purpose. The program's response will appear in the button or space where the user clicks or types. The frame is to look as follows when it first comes up for the user to see. If you would like to see this **MathQuiz** program in action, click this [execution](http://docs.google.com/EEdemos_AB.html#3) link.



When you solve a programming problem from scratch, you start with a drawing of how things look, as in the figure. 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. As stated earlier, you can do this using java MathQuizApp with a main method that contains the single statement new EFrame (new MathQuiz()); or you can use an applet whose start method contains setContentPane (new MathQuiz()); either works.

The accompanying block contains a reasonable design for the program, once the frame is created. See how easily it follows from the picture. And once you have a design for a program, the Java coding follows easily almost line-for-line. Note: A **textfield** is a rectangular area where the user can type input.

**STRUCTURED NATURAL LANGUAGE DESIGN for MathQuiz creation**

1. Add to the frame the question "What is the square of 30?"

2. Add to the frame a textfield where the user can type the answer to the question.

3. Add to the frame the question "How many bytes in a KB?"

4. Add to the frame a textfield where the user can type the answer to the question.

5. Suspend action until the user clicks a button or types in the textfield.

Listing 3 contains the MathQuiz class. The logic of the constructor follows the design:

�€€€€€€€€ Line 2 adds a label that contains the question "What is the square of 30?".

�€€€€€€€€ Line 3 adds a textfield where the user will type the answer; this SquareField object is wide enough to hold at least 8 characters.

�€€€€€€€€ Line 4 pushes the next component down to the front of the next row of components.

�€€€€€€€€ Line 5 adds another label that displays "How many bytes in a KB?".

�€€€€€€€€ Line 6 adds a textfield where the user will type the answer; this BytesField object is wide enough to hold at least 8 characters.

An EField has the same **setText(someString)**, **getText()**, and **text(someString)** methods that ELabels and EButtons have. The first two are inherited from Sun's JTextField class. Remember that text(someString) returns the EField itself, so you can use that value within thisEFrame.add(someComponent). If you want to use color in your programs, go to this [language](http://docs.google.com/EElanguage.html#3) link.

A subclass of **EField** is generally defined inside the EPanel subclass. You usually have just one method in it: **public void onEnter()** (see lines 8 and 17) will execute when the user presses the ENTER key within the textfield. A SquareField object is defined here to react to the ENTER key by first looking at the text the user has typed; line 9 does the looking. If the user typed "900", this textfield displays the answer "good!" (line 10), otherwise it displays the answer "you're wrong." (line 13). You can set the width of an EField, measured in characters, with the instance method **width(numCharacters)**; it returns the EField object itself. *Note that onEnter() is not called by coding anywhere in the program; it is called by the user's action of pressing the ENTER key.*

Listing 3 The MathQuiz program

**public class MathQuiz extends EPanel**

{

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

**�� public MathQuiz()�������������������������������** �����������// 1

�� {� add (new ELabel ("What is the square of 30?"));������� ��// 2

����� add (new SquareField().width (8));���� //in characters ��// 3

����� add (lineBreak());������������������������������������ ��// 4

����� add (new ELabel ("How many bytes in a KB?"));��������� ��// 5

����� add (new BytesField().width (8));���� //in characters� ��// 6

�� }� //======================

**�/\*\* Controllers: React to ENTER key. \*/**

**�� private class SquareField extends EField**����������������� ��// 7

�� {� public void onEnter()��������������������������������� ��// 8

����� {� **if** (this.getText().equals ("900"))����������������� ��// 9

�������� {� this.setText ("good!");������������������������� ��//10

�������� }�������������������������������������������������� ��//11

�������� **else**�������������� �����������������������������������//12

�������� {� this.setText ("you're wrong.");����������������� ��//13

�������� }�������������������������������������������������� ��//14

����� }����������������������������������������������������� ��//15

�� }� //======================

**�� private class BytesField extends EField**������������������ ��//16

�� {� public void onEnter()��������������������������������� ��//17

����� {� **if** (this.getText().equals ("1024"))���������������� ��//18

�������� {� this.setText ("good!");������������������������� ��//19

�������� }����������� �����������������������������������������//20

�������� **else**����������������������������������������������� ��//21

�������� {� this.setText ("you're wrong.");����������������� ��//22

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

����� }����������������������������� ��������������������������//24

�� }� //======================

}

[**Exercise**](http://docs.google.com/EEexercises.html#3) **3.1** If you changed the question in line 2 of MathQuiz to "What is the cube root of 64?", how would you have to change the rest of the program to make the textfield give the right responses?

**Exercise 3.2** Change MathQuiz so that, instead of displaying the answer in the textfield, a little box pops up with the answer. Hint: Review what the popUp method does.

**Exercise 3.3** Add a third question to MathQuiz that asks, "What is the square root of 144?". Allow both right answers to the question to have the "good!" response.

**Exercise 3.4\*** Change GeogQuiz so that, each time the user clicks the picture, it changes, from Rome.jpg to Ireland.jpg to grand\_canyon.jpg and then back to Rome.jpg. Hint: Put the picture on an EButton instead of an ELabel, and use the getPicture method.

**4. Instance variables**

The OrderForm class is designed to let the user make a purchase. It has the user fill out his/her name, address, and credit card number. The user types this information in three different textfields. Moreover, the user can click a button to say whether the company can email ads to the user. The frame looks as follows when the program begins. If you would like to see this OrderForm program in action, click this [execution](http://docs.google.com/EEdemos_AB.html#4) link.



The three textfield components itsName, itsAddress, itsCard, and the button component itsAds, have to be available for use in two or more methods. So they are declared outside of all methods. This makes them **instance variables** of OrderForm. A declaration of an instance variable is exactly as for a local variable except that we put the word **private** at the beginning of each such declaration. This means that the compiler will not allow any class outside of the OrderForm class to mention an instance variable. This is **encapsulation** -- it guarantees the integrity of the data you keep in those variables. You can, however, mention an instance variable anywhere inside the OrderForm class.

Each OrderForm object you create has its own set of four instance variables (though we usually make only one OrderForm object in our programs). The button in the OrderForm class is declared in line 4 as follows:

�� private EButton itsAds = new AdsButton().text ("yes");

This creates the button and initializes what it says on it to the word "yes". It also has the variable itsAds refer to the button. Any use of itsAds elsewhere in the program is a reference to this button.

The three textfields are defined similarly. They are added to the overall EPanel by a new kind of add method that has two parameters, such as the following in line 6:

�� add (new ELabel ("What is your name?"), itsName);

This **add(someComponent, someOtherComponent)** method in the EPanel class creates a small EPanel that contains the two components and then adds that small EPanel to the OrderForm EPanel. The effect is to "bind together" the two components by the small EPanel, so they cannot appear on different rows of the display. Typically, the first component is a label and the second is a textfield or button. The EPanel class also has a three-component add method **add(comp1, comp2, comp3)** for "binding together" three related components:.

The OrderForm class uses textfields that are just plain EFields; they do not belong to a subclass of EField and thus they do not have an onEnter method that contains coding. This means that, when the user click the ENTER key in the textfield, nothing happens. But when the user clicks the SubmitButton, all of the information in the three textfields and on the button is written to a file. You may ignore lines 23-28 on **EOutputFile** if you wish (unless your instructor says otherwise). It is not used elsewhere in this material. It is only here to show that data can be written to a disk file by an application (not by an applet). The close method makes sure all data goes to the physical file.

Listing 4 The OrderForm program

**public class OrderForm extends EPanel**

{

�� private EField itsName = new EField().width (20);���������� // 1

�� private EField itsAddress = new EField().width (20);������� // 2

�� private EField itsCard = new EField().width (20);���������� // 3

�� private EButton itsAds = new AdsButton().text ("yes");����� // 4

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

**�� public OrderForm()������������������������������** �����������// 5

�� {� add (new ELabel ("What is your name?"), itsName);������� // 6

����� add (new ELabel ("What is your address?"), itsAddress);� // 7

����� add (new ELabel ("What is your credit card #?"),�������� // 8

����������� itsCard);����������������������������������������� // 9

����� add (new ELabel ("Can we send you email ads?"), itsAds); //10

����� add (new SubmitButton().text ("Click here when done"));� //11

�� }� //======================

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

**�� private class AdsButton extends EButton**�������������������� //12

�� {� public void onClick()����������������������������������� //13

����� {� **if** (this.getText().equals ("yes"))������������������� //14

�������� {� this.setText ("no");������������������������������ //15

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

�������� **else**������������������������������������������������� //17

�������� {� this.setText ("yes");����������������������������� //18

�������� }��������������������������� �������������������������//19

����� }������������������������������������������������������� //20

�� }� //======================

**�� private class SubmitButton extends EButton**����������������� //21

�� {� public void onClick()����������������������������������� //22

�� �� {� EOutputFile file = new EOutputFile ("data.txt");����� //23

�������� file.println (itsName.getText());�������������������� //24

�������� file.println (itsAddress.getText());����������������� //25

�������� file.println (itsCard.getText());�������������������� //26

�������� file.println (itsAds.getText());��������������������� //27

�������� file.close();���������������������������������������� //28

����� }������������������������������������������������������� //29

�� }� //======================

}

[**Exercise**](http://docs.google.com/EEexercises.html#4) **4.1** Change OrderForm by adding a request for the user's phone number.

**Exercise 4.2** Change MathQuiz in Listing 3 to have one instance variable, a label named myAnswer. When a person enters a value in either textfield, display the response of "good!" or "you're wrong" in myAnswer instead of in the textfield itself.

**Exercise 4.3\*** Change GeogQuiz in Listing 1 to have the displayed picture switch from Rome.jpg to Ireland.jpg or vice versa, every time a RightButton is clicked. Hint: Have an instance variable that stores a reference to the ELabel with the picture.

**Exercise 4.4\*** Change OrderForm by adding two buttons for extra-cost shipping, one for express mail and one for airmail. Make them yes/no buttons that are initially "no". Set them so that making one "yes" changes the other to "no" (you can't have it both ways).

**5. Basic String operations**

If you already know about the String methods length, substring, and charAt, Unicodes, and using a plus sign for string concatenation, you can skip directly to the [next section](http://docs.google.com/EasyEvents_C.html) if you wish. This section does not describe new GUI features, but it provides further practice with what you have learned so far.

The **ArtQuiz** class has eight true/false questions (though the coding presented here only shows two of them, to save space). Each time the user clicks a button to give the answer, a smiley-face appears on a label named toReact if the answer is right and a frowny-face appears if the answer is wrong. Also, the total number right so far and the total number wrong so far are displayed on two labels named numRight and numWrong.

The frame looks as follows when the user has already clicked "true" for each question. If you would like to see this ArtQuiz program in action, click this [execution](http://docs.google.com/EEdemos_AB.html#5) link.



The variables numRight, numWrong, and toReact have to be available for use in two or more methods. So they are declared in lines 1-3 as instance variables of ArtQuiz objects, outside of all methods.

The numRight and numWrong labels are initialized with some text. The toReact label does not have an initial text, because it shows a picture (smiley-face or frowney-face) each time the user gives an answer. These labels are added to the EPanel as a group by the line 12 (adding all at once binds them together). The pictures are in a subfolder named "pic" (lines 15 and 21).

If you have a string of characters s, then **s.length()** is the number of characters in the string. For instance, "cat".length() is 3, "stew".length() is 4, and "".length() is zero. Also, **s.charAt(0)** is the character in the first position of s, s.charAt(1) is the character in the second positiion of s, etc. (numbering in Java normally starts from 0). For instance, "cat".charAt(0) is 'c' and "cat".charAt(2) is 't'. "cat".charAt(3) is not allowed.

Each character has a numeric value called its **Unicode value**. The Unicode for digit '0' is 48, for digit '1' is 49, up through 57 for digit '9'. Therefore, '3' minus '0' is the number 3, '7' minus '0' is the number 7, etc. In general, if c is a digit character, then c - '0' is the number that we represent in writing by the character c. We use this in lines 13-14 to add 1 to the count of right answers so far: Similarly, lines 16-17 compute the new count of wrong answers so far.

Putting a plus sign between two string values creates a new string consisting of the first string followed by the second string. This is called **string** **concatenation**. For instance, if s has the value "Bill", then s + " rules" has the value "Bill rules". Putting a plus sign between a numeric value and a string of characters, as in count+"right" where count is an int variable, converts the number to the character(s) that represent it in writing.

If s is any String value, the expression **s.substring(1)** is the string consisting of all characters except the first one. In general, s.substring(k) is all but the first k characters. You will see this substring String method used from time to time later in this material.

Listing 5 The ArtQuiz program

**public class ArtQuiz extends EPanel**

{

�� private ELabel numRight = new ELabel ("0 right"); ����������// 1

�� private ELabel numWrong = new ELabel ("0 wrong");� ���������// 2

�� private ELabel toReact = new ELabel("");� // no text �������// 3

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

**�� public ArtQuiz()������������������������������������ ��**�����// 4

�� {� add (new ELabel ("Mozart was a dancer"));��������� ������// 5

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

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

����� add (new ELabel ("Rembrandt was a painter"));�������� ���// 8

����� add (new RightButton().text ("true"));���������������� ��// 9

����� add (new WrongButton().text ("false"));���������������� �//10

����������� // several more questions would be here

����� add (lineBreak());������������������������� �������������//11

����� add (numRight, toReact, numWrong);���������� ������������//12

�� }� //======================

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

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

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

����� {� toReact.setPicture ("pic/smileyface.gif");�� ���������//15

�������� int count = 1 + numRight.getText().charAt (0) - '0'; �//16

�������� numRight.setText (count + " right so far"); ����������//17

����� }��������������������������������������������� ����������//18

�� }� //======================

**�� private class WrongButton extends EButton��������� ���������**//19

�� {� public void onClick()��������������������������� ��������//20

����� {� toReact.setPicture ("pic/frowneyface.gif");��� �������//21

�������� int count = 1 + numWrong.getText().charAt (0) - '0'; �//22

�������� numWrong.setText (count + " wrong so far");������ ����//23

����� }���������������������������������������������������� ���//24

�� }� //======================

}

[**Exercise**](http://docs.google.com/EEexercises.html#5) **5.1** Change ArtQuiz to add the true/false question for "da Vinci was a sculptor".

**Exercise 5.2** What difference would it make if you left out the "1+" part of lines 16 and 22? What if you also added 1 to count in lines 17 and 23?

**Exercise 5.3** The ArtQuiz class starts giving bad information if the user answers more than 9 questions right. Change ArtQuiz so that it does not try to add any more to numRight when the count of right answers reaches 9; do the same for numWrong.

**Exercise 5.4\*** There are e.g. 8 true/false questions altogether in the finished ArtQuiz class, a test-taker could click the correct answer to one of them 8 times and then show off the score of 8 right and 0 wrong. Change ArtQuiz to prevent this as follows: Each time a button is clicked, put a period at the beginning of its text, e.g., change "true" to ".true". But do not count it as an answer if its text already has a period (because it was already counted as an answer). Hint: Check whether the first character on the button == '.'.

**Exercise 5.5\*** Add two instance variables to ArtQuiz, both of int type, both initialized to zero. One will keep count of the number of wrong answers and the other will keep track of the number of right answers. Replace lines 16 and 22 to use these two variables appropriately. Also include the protective change described in Exercise 5.3.

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