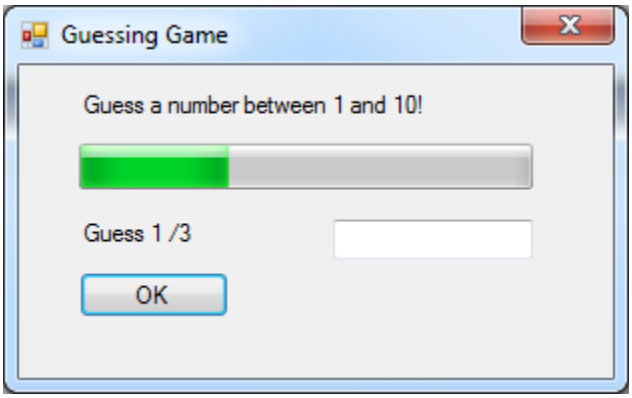
CIS 345 – Business Information Systems Development II – Fall 2014

In-class Exercise 9: Simple Form Application

Due on Blackboard: Tomorrow, Thursday, October 30 by 10:00 PM



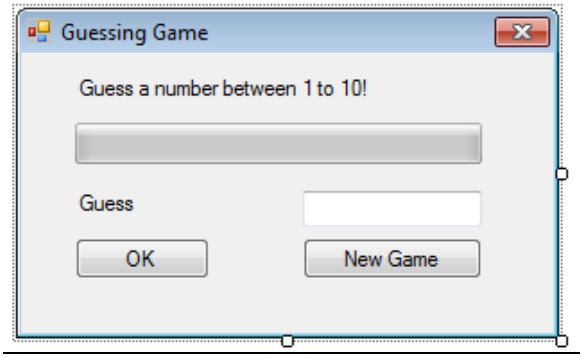
Create a number guessing game using a ***Windows Form Application*** within Visual Studio. This Guessing Game should prompt the user to guess a number between 1 and 10. It should generate a random number as the secret answer and compare the user guess against that number. It should give the user 3 tries to guess the correct answer. When the tries run out, it should deactivate the controls and allow the user to start a new game.

Project Details and Requirements

A sample executable file is uploaded and available on Blackboard for you to test program behavior.

Project Name: GuessingGame
Form Name: GuessingGame

Form GuessingGame [Design]



Properties: Since GuessingGame inherits from Form, a host of inherited properties are already available. The following properties will need to be set at ***design time*** using the IDE. Modify your form to look like that in the sample.

- Text (this property sets the form title): "Guessing Game"
- FormBorderStyle: FixedSingle
- MaximizeBox: False
- MinimizeBox: False
- StartPosition: CenterScreen

1) Set up the GuessingGame Form as show in the figure above. Use prefixes within the control names so that when you work with in code, you can easily distinguish a TextBox from a Label. e.g. "guessTextBox" should easily be interpreted as the TextBox which contains a guess, while "guessLabel" should easily suggest that is the label accompanying a guess.

Using the IDE, create the following controls. Set properties as specified below.

- Label, to give status feedback to the user. This will be the "Status Label."
  - Name: statusLabel
- Progress Bar, to give visual feedback
  - Name:progressBar
- Label, to prompt the user to guess. This will be the "Guess Label."
  - Name: guessLabel
- TextBox, to obtain input for the guess. This will be the "Guess Textbox."
  - Name: guessTextBox
  - TabIndex: 0
- Button, to allow user to press a button after entering a guess.
  - Name: okButton
  - Text: OK
- Button, to allow user to start a new game.
  - Name: startNewGameButton
  - Text: New Game

Form GuessingGame [Partial Class]

1) Within the GuessingGame partial class, add private instance variables, of type integer: secretAnswer, numberOfTries, and userGuess. ***Do not add any code in the Designer.cs or Program.cs classes.***

Any property setting that is listed below as being part of methods will NOT be set using the GUI Designer. It needs to be written in code!

METHODS:

Methods should go inside the class as with all objects. The precise location (top, bottom, etc.) does not matter so long as methods are located within the class block.

Create event handlers using the Properties window. If you write the event handler code manually, you will need to select the method within the properties/event window manually.

2) Load event handler.

Create a Load event handler for the Form.

Purpose: The purpose of this event handler is to set up some of the properties so that the Form is ready to be presented by the user.

General Method Logic:

- Set the minimum property of the progress bar to 0 and maximum to 3.
- Call the StartNewGame method *(Do this line after your method has been written – you will write this in part 4).*

3) DeactivateGame method

Parameters: none, *returns*: none

Purpose: This method should make the game unplayable by manipulating the *enabled* and *visible* properties of Controls. It will force the user to start a new game should they wish to continue.

General Method Logic:

- *Set the Enabled property of the OK button to false*
- *Set the Enabled property of the Guess Textbox to false*
- *Set the Visible property of the Start New Game button to true.*

4) StartNewGame method

Parameters: none, *returns*: none
Purpose: This method should reset everything to the initial state of a game, i.e., a new random number should be generated. Control properties should be reset to the starting state of a game. Variables should be reset to initial values.

General Method Logic:

- Declare an object of type Random and generate a new random number. This is the done in the same manner as before in the die-rolling application. Use the *Next* method. Use 1 and 11 as the min and max. *Go to the Object Browser to look up the Random class if you need to refresh your memory regarding using Random objects!*
- Set the OK button to be enabled.
- Set the New Game button to be invisible.
- Set the number of tries to 1
- Set the value property of the progress bar to the number of tries
- Set the Guess Label to show the number of tries. You will need to use string concatenation here i.e. string1 + integer1 + string2. You cannot use placeholders with the Text property.

  If you want to use placeholders, you will need to use the String class's Format method to create a formatted string first and then use it with a TextBox or a Label.
  e.g. string tempString = String.Format("{0} {1}", string1, string2);
  controlName.Text = tempString;

- Set the status label to show the initial message to the user, i.e. Guess a number between 1 and 10.
- Set the Enabled property of the Guess textbox to true
- Set the Text property of the Guess textbox to an empty string
- Call the Focus() method of the guess textbox.

5) ProcessMove method

Parameters: none, output: none
Purpose: This should be the main method which tests if the user had guessed the right number, if so, it should tell the user that they have won. If not, it tells them if their guess was too high or too low. If they run out of tries, it deactivates the buttons that allow the user to make a guess.

General Method Logic:

- If the guess is equal to the secret answer
    - set the status label to tell the user that they have won
    - Call the method to deactivate the game
- Else
    - If the user guess is higher than the secret answer
        - Set the status label to state that. Use string concatenation to combine the number and the text portions in one string that can be used to set the Text property.
            e.g. userGuess + " was too high".
    - Else
        - Set the status label to state that the guess was too low

    - Reset the Guess textbox text property to an empty string
    - Call the Focus() method of the Guess textbox to move the cursor back to the textbox.
    - Set the Value property of the progress bar to numberOfTries
    - Set the Guess label to show the guess number. Use string concatenation again here (you will need to use numberOfTries).
    - Increment the number of tries.
    - If the number of tries are equal to 4
        - Set the status label to indicate the correct answer and tell the user that they have lost
        - Call the DeactivateGame method

6) OK button Click event handler

Purpose: This method will store the user guess and call ProcessMove if there are still tries left.

General Method Logic:

- Store the value of the Text property from the textbox into userGuess. Convert into a number first. *There is no Console anymore! So don't try to use Console.ReadLine(). You are getting input from Text properties now!*
- If the number of tries are less than or equal to 3, call ProcessMove()

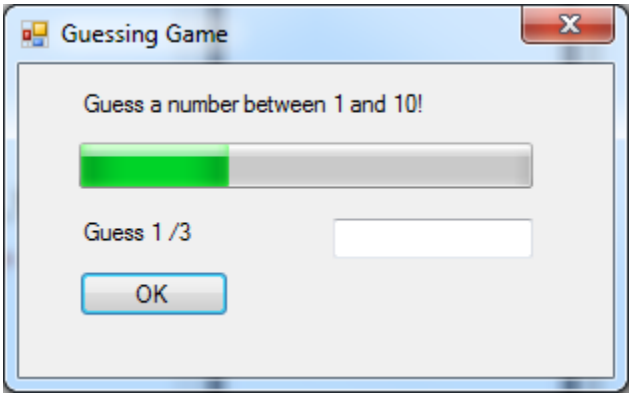
7) Start New Game button Click event handler

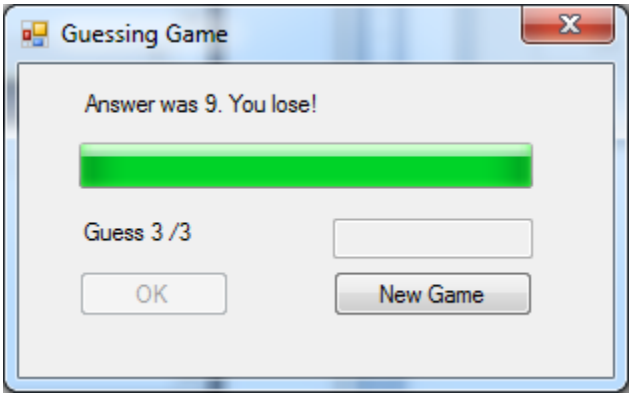
*Purpose:*This method will start a new game by calling the appropriate method.

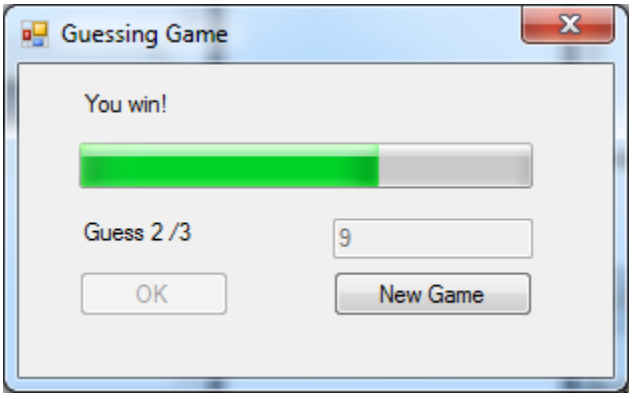
General Method Logic:

- Call the StartNewGame() method

Sample Output:



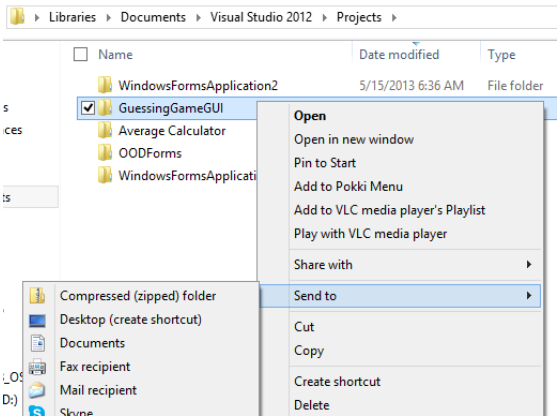




Submission Instructions

Submission should be made using a zip file that contains all of the Visual Studio C# project files. You will need to ***zip the entire project folder*** along with the .sln and .suo files. The folder will automatically contain the class source files as well as the executable file that is generated in \ProjectName\bin\Debug folder. Upload file to the Blackboard assignment drop box.

Zip the entire top-level folder by right-clicking the folder and selecting Send to | Compressed (zipped) folder.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Make sure you check the following. Your grade is dependent on all these criteria being met.

- You have included your name as a comment within your class.
    - e.g. " // In-class 9, Jane C. Smith, CIS 345, Tuesday 9:00 AM"
    - Your project is called GuessingGame
- Zip filename is: FirstNameLastName_Inclass9.zip

Verify your zip file before you submit

- Check for actual class files being present in the folder before you zip it.
- Check your zip file size after zipping – if it is 1K, it likely contains only a shortcut.
- Uncompress your zip file before submitting and verify that files are present.
- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.
- Make sure you have submitted your file and not just saved a draft on Blackboard. A blue clock indicates a submission in progress, i.e. a draft, not a submission.