# PE-17: Multiple Forms

## Objective

Create a **number guessing game** to practice creating additional forms in one project, opening them programmatically and getting them to communicate. Start with a new Windows Forms project in Visual Studio.

## Setup: The Parameters Form

Form1 (or whatever you name it) will let the user set up the guessing game parameters. Set up it up as follows:

* A start button
* Several text boxes for user input (with appropriate labels):
  + The **low end** of the random number range
  + The **high end** of the random number range
  + Pre-populate the text boxes above with default values of 1 to 100

Here is the pseudocode for my Start Button click handler:

private void StartButton\_Click(object sender, EventArgs e)

{

bool bConv;

int lowNumber = 0;

int highNumber = 0;

// convert the strings entered in lowTextBox and highTextBox

// to lowNumber and highNumber Int32.Parse

// if not a valid range

if ( ??? )

{

// show a dialog that the numbers are not valid

MessageBox.Show("The numbers are invalid.");

}

else

{

// otherwise we're good

// create a form object of the second form

// passing in the number range

GameForm gameForm = new GameForm(lowNumber, highNumber);

// display the form as a modal dialog,

// which makes the first form inactive

gameForm.ShowDialog();

}

}

## Setup: The Game Form

Create a second windows form class called GameForm. (Right click the project name in the Solution Explorer and select Add/Form (Windows Forms)…). Be sure to select “Windows Form” instead of “Class” when creating it, as that will create the designer file appropriately and give you the required using statements.

It should have the following controls:

* A text box for the user’s **current guess**
* A label for **output** (whether the current guess is high or low)
* A **guess button** the user can click
* A **status strip** that includes a **progress bar** that counts down while the game is in progress (similar to MyEditor)
* A **timer** control that ticks once per 500ms

The GameForm class itself will need a parameterized constructor to accept the custom data for the game (low and high numbers). It’ll also need to do things like generate a random number, start the timer, etc.

This code will generate a random number between 1 and 10:

Random rand = new Random();

int nRandom = rand.Next(1, 11);

The timer’s tick event will need to update the progress bar accordingly and potentially cause the game to be over after 45 seconds has elapsed.

## Playing the Game

This part will require you to add code to both form classes.

When the program starts up, the user should be able to start the game or change the values in the text boxes to customize the game. When the user clicks start, verify the game parameters are valid:

* There’s a valid value for the “low” number (it’s actually a number)
* There’s a valid value for the “high” number (it’s actually a number)
* Low is less than high

If the input is invalid for any reason, pop up a message box with a useful error message. Do not start the game until the input is valid. (Use the KeyPress event handler and Char.IsDigit() to only allow the user to type digits in the TextBoxes.)

Once you’ve verified the data is valid within the Click event handler, create a new GameForm and show it to the user. Using the following code will disable the first form, so that the user can’t start a second game while the first is active:

GameForm gameForm = new GameForm(lowNumber, highNumber);

// ShowDialog() creates the form as a Modal dialog

// which disables the parent form while the Modal dialog is active

gameForm.ShowDialog();

At this point, the user should type numbers into the guess text box and click the guess button. Update the output label after each guess, letting the user know if they were high or low. Something like “Your guess of 56 was HIGH”.

* If the user guesses the correct number before the timer runs out, stop the timer and pop up a congratulatory message box.

Use the MessageBox class to display dialogs to the user. For example:

MessageBox.Show($"Woohoo, you got it in {nGuesses} guesses!");

* If the user fails to guess the number, pop up a failure message box.
* In either case, once the user closes the message box, close the form itself by calling this.Close( ).

The user should be able to play again as many times as they’d like.

You should set the AcceptButton for the Game form so that the user can simply press Enter for each guess.