W. P. CAREY SCHOOL of BUSINESS
ARIZONA STATE UNIVERSITY

CIS 345 – Business Information Systems Development II – Fall 2014

Assignment 3

Due on Blackboard: Friday, September 26, by 10 PM

Create a Student Grade System using C# that stores a list of 5 names and corresponding scores and is able to look up scores by name. Utilize two different arrays to store the names and integers but keep the order of the names and scores synchronized i.e. if John's name is in index 1, then John's score should also be in index 1 in a different array.

Skills developed: Loops, method writing, method calls, handling return values, arrays, handling reference type variables, passing reference type variables as arguments, logic implementation.

Before you begin

Run the sample executable provided to you. Be familiar with the behavior of the program.

Read the full document and understand what each method does ***before*** beginning to program. Do not write methods before you know what they do.

Commenting, Style, and Documentation

This assignment (and any future assignment) requires you to use descriptive variable names and proper case conventions. Use *PascalCase* for methods and *camelCase* for local variables and parameters, e.g., "firstName", "CalculateAverage()", "averageScore", "LookupRecord()".

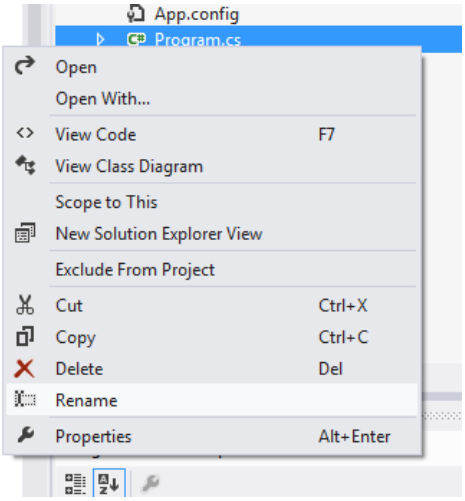
Describe major pieces of logic in English. Describe the intended use for variables. Don't inundate the page with comments but rather help a person unfamiliar with the code understand what you were thinking when you wrote it – think of documenting "why" you did something. With descriptive variables and method names, code becomes largely self-commenting.

In order to get full credit, your code must follow the programming conventions listed above. Programs that do not follow this, e.g., which name their variables as "a" and "b", will get a grade no higher than a B for a fully working application. *Why? 1) We are trying to adopt good programming practices 2) Not doing it results in code that is not easily maintained 2) If the specified standard is deviated from, it takes more time to understand code resulting in grading delays for the entire class.*

Note: You are not required do input validation.

Create a new Visual Studio 2012/13 project. Start with the project name "Assignment3". **Do not put in your name here.**

Rename your class name from Program.cs to GradeSystem.cs by right-clicking on the filename in Solution Explorer and clicking rename. Accept its recommendation to change the name of the class throughout the project code.



Your program should have the following static methods:

- *Method Name:* PopulateNameArray
  Purpose: Prompt the user and read student names and store them into an array of strings.
  Parameters: an array of strings called "names"
  Return Values: None

  <u>General method logic:</u>
  Tell the user that they will need to enter # names. Get the value of # from the array length.
  Loop:
  Prompt the user for one name
  Read the name and store it into the array you received as a parameter (it is called names). (*You will store the name inside an array element, not the array. Therefore, pick an array element number*).

  This method does not return any value.

- *Method Name:* PopulateScoreArray
  Purpose: To prompt the user and read 5 student scores and store them into an array of strings. This method will prompt the user to enter a score by name. e.g. "Enter John's score:"
  Parameters: an array of integers called "scores", and an array of strings called "names"
  Return Values: None

  General method logic:
  Tell the user that they will need to enter scores.
  Loop:
  Prompt the user to enter the score of the first student by providing the name of the student, which is read from the string array (called "names").
  Store the score for the student in the integer array (called "scores").
  You will store scores using the same index value as the name array.

  This method does not return any value.

- *Method Name:* FindStudentPosition
  Purpose: This method accepts an array of strings containing student names and it also accepts the name of a student. Its purpose is to go through the entire array and search for that student. Once it finds the student, it returns the location of that student in the array. This means that if the student name is found in index 5, it returns 5. If the student name is not found in the entire array, it returns -1.

  This method is NOT returning either a student name or a student score, but rather only his or her location in the array. It is also NOT printing anything to the Console.

  Parameters: a string called "name" and an array of strings called "stringArray".
  Return Values: Integer

  General method logic:
  Declare a variable to record the location called "studentLocation".
  Initialize its value to -1.
  Loop:
  Compare the contents of the current stringArray element with the student name.
  If the names are a match, assign the element number to studentLocation
  You found who you were looking for! Stop looking! Break out of the loop.

  Return studentLocation *(this is outside the loop.)*

- *Method Name:* Main
  Purpose: The Main method is where the program will actually execute. This method will maintain copies of the arrays storing student names and scores and it will call the other three methods that you wrote.

  General Method Logic:

  - Declare two ***local*** array variables in your method:

    One array should be of type integer and named scoreArray
    One array should be of type string and name nameArray

  - Initialize nameArray and scoreArray each with 5 elements. *This is also called allocating elements.*

  - Print a welcome message to the console window and then call the PopulateNameArray and PopulateScoreArray methods. Pass the appropriate arrays as arguments, "by value", to each method. *Note that arrays are reference types. You will NOT need to use the ref keyword.*

  - Clear the console screen by calling the Clear() method of the Console class. *Look up if the method is static or non-static!*

  - Prompt the user to enter a name for whom to look up a score. It should store that in a local variable (you will need to declare one!)

  - Call FindStudentPosition and pass the variable from the line above as an argument (you will also need to pass the nameArray!). Store the return value of the FindStudentPosition method in a variable called studentPosition.

  - If the return value stored in studentPosition is -1, it should tell the user that a student by that name does not exist in the database.
  - Otherwise, it should look up the score using scoreArray at the appropriate index (studentPosition) and print out the score along with the name of the student to the console.

Optional Enhancement (no extra-credit)

Write a method, *WriteCenteredLine*, to write centered text on the screen, such as the program heading. You should have a method that accepts a string as a parameter.

As you already know, you can insert preceding spaces by using {0,x} as a formatting instruction, where x is the number of spaces. Calculate x in your method by obtaining the width of the Console (Console.WindowWidth). Add the length of the given string (stringVariable.Length) you received as a parameter to the Console width, and divide the result by two. This final value should represent your x. Use value of x in your WriteLine within your formatting instruction braces.

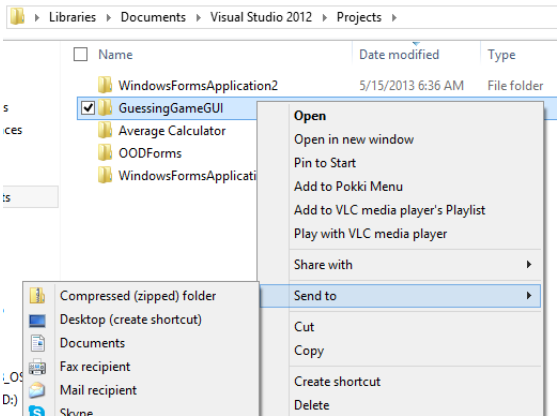
Call method as needed! Copy method into future assignments :-)

All of this can be done in two lines of code in the method ☺

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. " // Assignment 3, Jane C. Smith, CIS 345, Tuesday 9:00 AM"
- Class file is called GradeSystem.cs (rename from Program.cs).
- Your Visual Studio project is called Assignment2.
- Zip filename is: **FirstNameLastName_Assignment2.zip**
- Your code is commented and you are using all prescribed programming conventions.
- Your code utilizes PascalCase and camelCase as appropriate.

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.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.