**CIS 345 – Business Information Systems Development II – Fall 2014**

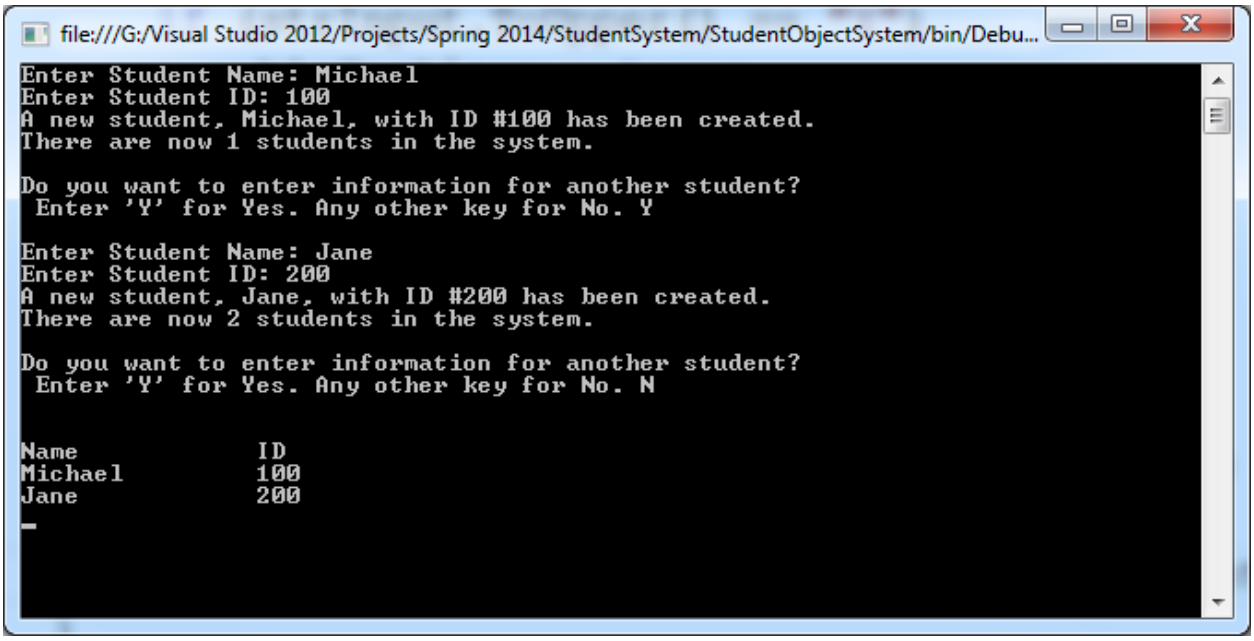
In-class Exercise 7: Objects Exercise

Due on Blackboard: Tomorrow, Thursday, October 16, by 10:00 PM

This exercise is meant to give you practice in using objects, constructors, and static and non-static methods. It also involves properties, methods, arrays, looping, instance, static, and local variables, etc.

Create a project, StudentSystem, and within:

- Have one class called StudentSystem
  - *Rename Program.cs as StudentSystem*
- Create one class called Student
  - *Work on this class first while leaving StudentSystem empty*



# Student Class

| **Student** |
| --- |
| - studentCount: int<br>+ <<property>> Name : string<br>+ <<property>> ID : string<br>+ <<property>> StudentCount : int {readonly} |
| +<<constructor>>Student (studentID: string)<br>+<<constructor>> Student (studentID: string, studentName: string) |

You may use auto-implemented properties for Name and ID to do it quickly. (However – it is okay if you create them manually by writing out get and set acessors).

Constructor Logic

1. Constructor method with only the ID as a parameter
    - The constructor should assign the ID received as a parameter to the instance variable representing ID within the class. *If you have a private variable for ID, you would store the ID in that variable. If instead you have only a public property for ID, you would store the ID using that property.*
    - Increment the student count by 1.
    - Write to the console that a new student with xxx ID has been created.
    - It should also output to the user the total number of students.

2. Constructor method with ID and Name as parameters.
    - The constructor will assign the ID and Name to the instance variables (similar to part 1).
    - Increment the student count by 1.
    - Write to the console that a new student with xxx name and ID has been created.
    - It should also output to the user the total number of students.

# StudentSystem Class

| **StudentSystem** |
| --- |
| - studentRoster: Student[] |
| + Main(args: string[])<br>- LoadStudents(): void |

LoadStudents Method

- Allocate 10 elements to the studentRoster array.
- Declare a boolean variable, continueLoop and initialize it to false.
- Declare local variables for storing name and ID. Name them appropriately.

- Create a do-while loop, and keep looping till continueLoop is true. Within the do-while loop,

    - Set continueLoop to false. *We are assuming that the user does not want to enter any more students.*
    - Ask the user for the name and ID of a new student. Store them in the appropriate variables you created.
    - Instantiate a new student – call the constructor and pass it the necessary arguments. Store the reference to the newly created student in the studentRoster array. Use the count property from the Student class as the index location.
    - Ask if the user would like to create another student. Read input, compare for "Y" and if the user entered "Y", assign true to continueLoop. *We are allowing the user to change our assumption of no more students.*

- Loop through all the students in the studentRoster array and print out a list of students. *Remember to use the count variable from the Student class.*

Main Method

- Call LoadStudents

**Learning Exercise**

Rather than assigning the reference to the newly created Student instance directly to the array, store it in a temporary reference, say, tmpStudent. Work on tmpStudent to read/write Name/ID. Then use that tmpStudent reference to store the new student into the array.

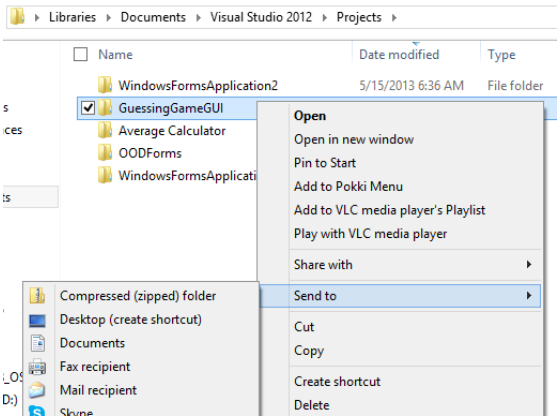
Similarly, when looping through the array to read information, do not access properties directly. Take the array element and store a reference to it in tmpStudent. Then use tmpStudent rather an accessing objects directly from the array.

*One advantage of using a temporary reference is that you avoid a lot of typing when you need to refer to one object over and over. This is particularly useful when your object is contained within another object, which might be contained in another object, and so on. The other advantage is that once you store a reference into a temporary reference, you are assured that you are using the reference every single time. On the other hand, if when we retrieve it directly from the array over and over, it is possible to make an error and end up working with some different unintentionally.*

## 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 7, Jane C. Smith, CIS 345, Tuesday 9:00 AM"
- Class files are called Student and StudentSystem
- Your Visual Studio project is called StudentSystem.
- Zip filename is: FirstNameLastName_Inclass8.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.