Programming 120 – F22 - Final

You have a client who has a two classes they teach. They’ve provided data, but don’t have any way to interact with it. They’re a bit old school so they’re fine using a console for their app. We’ve broken down the features the teacher wants.

1. Display All The student’s information and grades, including the average of both classes
2. See if a student is enrolled by entering their first and last name
3. Lookup a student by either passing in a student id ( index), or their first and last name
4. Update students information ( First, Last, CSI Grade, GenEd Grade ) using a school id ( index )
5. Display all students who have an average grade between two numbers ( a min and max value )

In the future, someone else will maintain your code. So, make your you properly comment on your code and have the variable names make sense.

We’ve broken down a list of methods your code should have in order to make this work, but you can create others as needed. The code is yours to write. Good luck.

Required Code:

Rubric

|  |
| --- |
| Basics – 30 Points |
| Commented header at top of code ( name, date, assignment )  Comments throughout code explaining functionality  Proper use of try / catch  Proper use of TryParse()  Use a switch for the Main menu  Main Menu loops until user exits  Your name displayed at the top of your app on run, with Professor.  Ex. Professor { your name }  (“Professor Hannah Angel”) |

|  |
| --- |
| Menu Functionality – 60 points *Works properly and loops ( sub menus don’t need to loop )* |
| 1 – Display All Students with a formatted string  2 – Check Student Enrollment  3 – Student Look Up  4 – Update Student Info  5 – Display Students with an Average Grade  e – Exit |

|  |  |
| --- | --- |
| Methods Required – 80 points | |
| ***Display***  void FormatStudent(int index)  void DisplayStudents()  void DisplayStudentsWithGrades(double min, double max)  ***Search***  bool StudentEnrolled(string fName, string lName)  int StudentIndex(string fName, string lName)  ***Math***  double GradeAverage(double csi, double gen)  double ClassAverage(double[] classes) | ***Functional***  void UpdateStudent(int index)  ***GUI***  void Menu()  void CheckStudentEnrollment()  void StudentLookUp()  void UpdateStudentInfo()  void ViewStudentWithGrades() |

|  |
| --- |
| Test Code – 30 points *Use these values to check your code is working properly* |
| Method: StudentEnrolled(“Samuel”, “Vimes”); - *Result: true*  Method: StudentEnrolled(“William”, “Cram”); - *Result: false*  Method: StudentIndex (“Ford”, “Prefect”); - *Result: a positive int ( not -1 )*  Method: StudentEnrolled(“William”, “Cram”); - *Result: -1*  Method: GradeAverage(Aurther Dents CSI Grade, Aurther Dent Gen Ed Grade); - *Result: 38*  Method: DisplayStudentsWithGrades (101, 103); - *Result: Havelock Vetinari* |

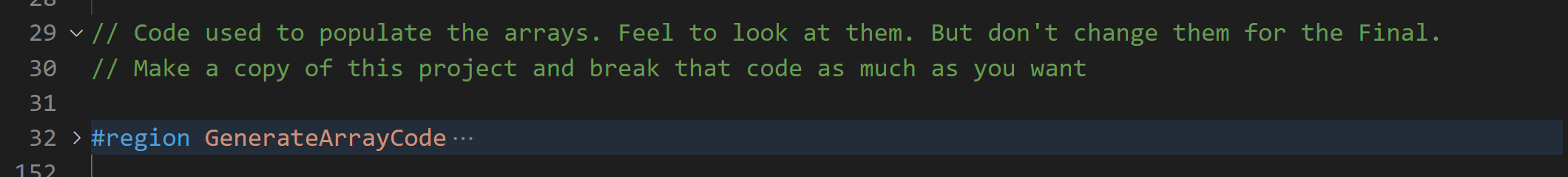
Provided Code

There are 4 arrays provided for the final. There is code written to populate them when you run your app. You will interact with these 4 arrays for your displays, searches, math, and functionality methods. You don’t need to create new arrays for this assignment.

Text

Description automatically generated

At the bottom, you’ll find a “region” called GenerateArrayCode. Click on the arrow next to it will show you the code used to populate the arrays. This should not be changed for the final. BUT you should definitely play with it in your own time to see what it does. Feel free to ask questions.



Display Methods

*These methods are used to display information about the students.*

* *void FormatStudent(int index)*   
  Take an index as an argument. Displays students information.

Ex. Index – First

Name – Last Name – CSI: CSI Grade – GenEd: GenEdGrade – Average: Average Grade



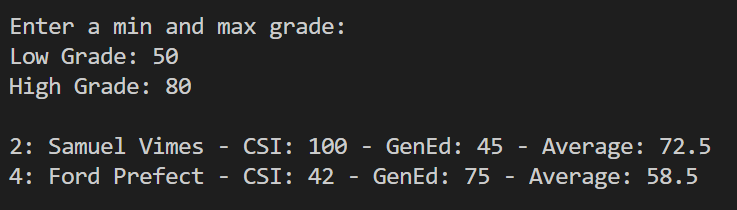
* *void DisplayStudents()*  
  Displays all students with proper formatting

Also Displays Number of enrolled students  
Average grade of CSI Class  
Average grade of Gen Ed Class  
Average grade between both classes

Text

Description automatically generated

* *void DisplayStudentsWithGrades(double min, double max)*  
  Takes two arguments, a min grade and a max.   
  Displays all students with an average grade in that range.



Search Methods

*These methods are used to search the arrays for student information. Uses linear search*

* *bool StudentEnrolled(string fName, string lName)*

Create a method that checks to see if a student is enrolled.

*Linear Search: Loop through the first and last name arrays. Compare fName against the element in the first name array, and lName in the last name array.*

*Result: If firstname AND lastname are found at the same index, return true. Else return false*

* *int StudentIndex(string fName, string lName)*

Create a method that searchs for a students information.   
Return their index if the student is found.

*Linear Search: Loop through the first and last name arrays. Compare fName against the element in the first name array, and lName in the last name array.*

*Result: If firstname AND lastname are found at the same index. Return the index, else return -1.*

Math

*These methods are used to perform calculations*

* *double GradeAverage(double csi, double gen)*

Takes two values

Returns the average of those two values

Values should be a CSI Grade and a Gen Ed grade

This can either be used for one student, or you can pass in the values from Class Averages to get the total class average

*Result: When you pass in one students csi grade and gen ed grade, should return their average.*

* *double ClassAverages(double[] class)*

Takes a double array ( these can be either the csi grades or genEd grades. The math is the same )

Using a loop, sum all the grades

Using .Length, return the average.

*Result: When you pass in a class array, it should show the average of the entire class.*

Functional

*These perform specific tasks*

* *void UpdateStudent(int index)*

Takes an index

Using FormatStudent(index) display the current students information

Ask the user for a new first name

Ask the user for a new last name

Ask the user for a new CSI Grade ( use try parse to validate )

Ask the user for a new CSI Grade ( use try parse to validate )

*Result: When you display the students, your new student should have replaced the previous one.*

GUI

*These methods are used to control your menu structure*

* *void Menu()*

Creates your main menu structure. Display {school} database at the top – You can put what you want

Display Professor {your name}

Menu Options

1. Display All Students
2. Check student enrollment
3. Student Lookup
4. Update Student Info
5. Display Students with Average Grade
6. Exit app

Ask user to enter an option

Loops until user chooses to exit app

Text

Description automatically generated

* *void CheckStudentEnrollment()*

Display “Check for student enrollment”

Asks for a first name

Asks for a last name

Display if a student is enrolled or not ( the message your display is your choice )

Return to the main menu afterward ( don’t call Menu());

Text

Description automatically generated

* *void StudentLookUp()*

Display “Display Student Information”

Use your StudentIndex() method to find the student.

If found, use your FormatStudent() method to display student information

Else display the student wasn’t found. ( message is your choice )

Return to the main menu afterward ( don’t call Menu());

Text

Description automatically generated

* *void UpdateStudentInfo()*

Give the user an option to update student by index or student name

If by index, take an int.

If by student name, use the StudentIndex() method to find the index.

Use the UpdateStudentInfo(index) method by passing in the student index. This should prompt the user for the new information.

Return to the main menu afterward ( don’t call Menu());

Text

Description automatically generated

* *void ViewStudentWithGrades()*

When called, asks the user for a low and a high grade

Using DisplayStudentWithGrades(min, max), display a student with averages between that range of values.

Return to the main menu afterward ( don’t call Menu());

*Text

Description automatically generated*

Tips:

There are a lot of small moving pieces, but none of them are too big or complex. Review Linear Search, working with arrays, and methods, and our fundamentals and you’ll be fine.

Like building a house of cards, the order that you build your code can make your life easier or harder.

1. Build your Math methods first and test them.  
   *Reason: These methods don’t rely on any of your other code to work properly, outside of the arrays.*
2. Build your FormatStudent(). Get that working properly with one student. Use GradeAverage() to display the students’ average in the formatted display.  
   *Reason: Being able to display your information is key to testing and figuring out if your code is working.*
3. Build your DisplayStudents(). This should loop through your list and use FormatStudent() to display their information. Afterward, you can display all class information.  
   *Reason: Now that you can display all your students, you have the foundation for troubleshooting the rest of your code as you go.*
4. Etc…

Save the Main Menu for last. Build in small bite-size chunks.

* Functional Code: Build the code that does something ( finds a student, changes a grade, Display students )
* Selection Code: Build the code that lets you ask the user for their input, then pass that into your Functional Code, and display results
* Menu: Build the code that lets the user interact with our Selection code by choosing from an option.

Final Result