Scalable Data Infrastructures - Project 04

Overview

This project is a synthesis to include many of the concepts and tools you've learned and worked with this month, including methods, Lists, input, output, and string methods. Because it is a large project, you will want to give yourself plenty of time to work out the problem in a problem analysis as well as time to create and debug your code.

Instructions

To start, think of a List of four items. It might be four items you own or four items you want to sell. But, they should be related in some way. As with other projects, you can decide what the items are (cars, gaming systems, books, etc.). You will then think of prices for each of these items. So, before you start writing your code, know what the four items are in for one List and the prices for each item for a second List.

Now, create a new Visual Studio project with the following naming convention: *LastName_FirstName_PR04*.

The first thing you can add to this project (after your comments) can be two Lists with the data you've decided to use. Remember that the data type of the Lists should reflect the data they are going to hold.

Create a custom method that will output the items in your Lists. You will need to send both Lists to this custom method as arguments so they will be available to the method. Within the method, use a single loop to output the elements of both Lists to the console in a meaningful way that allows the user to select an item. That is, you'll output each List item along with a number the user can use to select the item, such as the following:

1. Subaru Outback: $20000
2. Ford Focus: $17000

and so on for each item and price in your List. That is all this method should do, which is to build a menu from which the user can select an item.

After outputting the elements to the Console, you will then ask the user to enter the number of the item they wish to remove from the Lists. This user input will be done in the Main method. After the user inputs the number, you will send that value to a custom method that will validate that they entered a number and that the number is within the bounds of the menu presented. In other words, they need to enter a number from the menu. Once the number is validated, it should then be returned to the Main method.

In the Main method, the code will use the number entered to remove that item from the List. Remember that you need to remove the item from one List as well as the corresponding dollar value from the other List.

Next, ask the user for a new item to add to the List. Then ask for the cost of the new item. Both will need to be added to the Lists as the first items in the Lists after the inputs have been validated. To validate these user inputs, you will need to create two additional validation methods, one to validate the string and one to validate the floating point number (you should then have three validation methods: one to validate an integer within the bounds of the menu, one to validate strings, and one

to validate floating point numbers). Send each user input into its proper validation method, return the value back to the Main method, then add the validated input to the Lists.

Finally, invoke the method to output the items in the List one final time to allow the user to see the items on the List.

Things to Consider

Look to your problem analysis to verify that you've included all the requirements of the problem. If you ask your instructor or a lab specialist for assistance on the code, he/she is going to ask to see your problem analysis first. If you didn't do that, we are going to tell you to complete that task before we will look at your code.

Verify that you have all the required custom methods in the project and that each custom method has the appropriate arguments, parameters, and returns as this is an important aspect of the overall project.

Make sure you're including comments at the top of the code to include your name, class and term, and the assignment name and that you have meaningful comments for each line of code in the project.

Finally, remember that you must compress the entire project folder for submission. Submitting only the Program.cs file or the .sln file will result in a 0 for the activity.

Rubric: Project 4

Minimum Project Requirements

These requirements must be satisfied before any points are awarded. Failing to meet these requirements will result in a zero (0) grade.

1. Project must run when instructor compiles it.
2. The submission must be submitted in the proper format as defined in the FSO activity.
3. You will lose 5 points if the project does not follow the naming convention described in the activity's documentation.

Topic	%	Excellent (100%)	Acceptable (80%)	Good (50%)	Fair (25%)	Poor (0%)
Coding						
Comments	**5**	Comments exist at the top of the code to include name, class and term, assignment, and date, and each line of the code is properly commented.	Missing the initial comments with name, class and term, assignment, and date, but the rest of the code is commented properly.	Missing up to four line comments, but some comments present.	Missing more than four comments	No comments in the code.
Syntax	**10**	There are no syntax errors, including correct line and **formatting according to the style taught**.	There are no syntax errors, but the code does not follow the style taught.	Project code contains minor syntax errors but is easily fixed.	Project code contains more major syntax errors but are easily fixed.	Project code does not run.
Methods	**25**	All required methods are created and include all the requirements of the method signature, including a descriptive, PascalCase, active-verb name with correct return and parameter.	Custom methods are created, but the names are not descriptive or not active verbs or not in PascalCase.	Missing up to one required method.	Missing more than one required method, but there is a custom method in the project.	**No custom methods exist in the submitted project. Zero (0) for the entire project.**
Arguments/Parameters	**15**	All required arguments/parameters are present and of the correct type with descriptive names.	One missing argument/parameter or incorrect data types used.	Missing more than one required argument/parameter.		The custom methods contain no parameters.
Returns	**15**	All required return values are present, assigned, and of the correct type. All returned values are stored in return value variables.	One missed, unassigned, or mistyped return values, or up to one returned value not stored in a variable.	No returned values are stored in return value variables.		The custom methods do not return values.
Validation	**10**	The user inputs are validated with the proper loop, parsed to the proper data type, and the validation is encapsulated in the code block of the required custom methods.	The user input is not converted/parsed to the correct data type.	User input is validated with a conditional, not a loop.	User input validation is contained in the Main method, not in the custom methods.	The user input is not validated.
Method Calls	**10**	The methods are properly invoked with the correct argument data being sent to the custom methods and with a returned value variable in place to capture the value returned from the methods.	The methods are invoked, but missing at least one returned value variable to hold returned values or missing at least one argument in the method calls.	The methods are invoked, but no arguments are sent into the methods or no returned values are properly stored in the Main method.		The custom methods are not invoked.
Output	**10**	The correct loop is used to output the List elements, output is meaningful and done in the proper methods.	Output is done in the proper methods, but the text is not meaningful to the user.	The wrong type of loop is used to output the data from the Lists or only one List's elements are output.	More than a single loop is used to output the elements of the Lists.	The output is not done properly or in a meaningful way.