Scalable Data Infrastructures - Code Exercise 05

Overview

Now that you know a little something about methods, arguments and parameters, and return values, this code exercise will allow you to take something you already know how to do and encapsulate it into a method

Instructions

It's always best to start with a problem analysis. While one is not required to be submitted for this code exercise, it would benefit you greatly to begin by examining the requirements and creating a bulleted list of the things you'll need to complete the code. You can then begin by creating a new solution with the following naming format: *LastName_FirstName_CE05*.

In the Main method, you will need to create a user input so that the user can enter a number. The number can be anything you like, such as a cost, an age, number of books you own. Just be sure to indicate what the number represents when asking the user to enter it. This user input will then be sent to a custom method for validation.

You will need to create this custom method to validate the user input. You can give it any name you like, but it should be relevant to the task it will perform. In this case, it will be a method that validates user input. It will need to accept a string as an argument and return a correct data type for the number entered by the user. If you allow the user to enter a floating point number, the method should not validate and return an integer. The code inside this method should validate if the string it receives can be parsed to the proper number data type. If it can, the method should then return the number value to the Main method. If it cannot, the method should contain a validation loop that will ask the user to try again to enter a proper number. In other words, the code will contain the validation loop and it will run while the user is not entering proper number data. Once the user enters a proper number, that number is returned to the Main method.

Once the number is validated and returned to the Main method, output that number to the console in a meaningful way. Outputting the number in the custom method will result in a loss of points.

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.

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: Code Exercise 5

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	**5**	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.
User Input	**10**	ReadLine is set up correctly with a corresponding WriteLine that contains descriptive text to indicate what the user must do.	ReadLine is set up correctly, but the text of the WriteLine is not descriptive.	User input is gathered in the custom method only, not in the Main method.	ReadLine is set up correctly, but there is no WriteLine to indicate what the user must input.	ReadLine is not used or is not set up properly.
Custom Method	**20**	A custom method is created an includes all the requirements of the method signature, including a descriptive, PascalCase, active-verb name with correct return and parameter.	A custom method is created, but the name is not descriptive or not an active verb or not in PascalCase.	A custom method is created but indicates an incorrect return type in the method signature.	A custom method is created but is missing a return type in the method signature.	**No custom method exists in the submitted project. Zero (0) for the entire exercise.**
Argument/Parameter	**15**	A parameter of the correct data type exists as part of the custom method and has a descriptive name.	A parameter of the correct data type exists as part of the custom method, but the name is not descriptive of the data it will hold.	A parameter is part of the custom method but it is not a string data type.		The custom method contains no parameter.
Return	**15**	A return is included in the custom method and is of the correct data type.	A return is included in the custom method, but the data type is incorrect.	A return is included in the custom method, but the data type does not match the data type indicated in the method signature.		The custom method does not return a value.
Validation	**10**	The user input is validated with the proper loop, parsed to the proper data type, and the validation is encapsulated in the code block of the custom method.	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 method.	The user input is not validated.
Method Call	**10**	The method is properly invoked with the correct argument data being sent to the custom method and with a returned value variable in place to capture the value returned from the method.	The method is invoked, but no returned value variable is used to capture the value returned from the method.	The method is invoked, but no argument is sent into the method to be validated.	The method call does not send an argument value and no return value is captured.	The custom method is not invoked.
Output	**10**	Output is meaningful and done in the Main method.	Output is done in the Main method, but the text is not meaningful to the user.			The output is not done in the Main method.