# IT 230 Module Three Coding Activity Documentation- DebugFixMethods

**Name:** Grace Tay

**Date:** November 7, 2016/ TryParse finished on Nov 13, 2016

**Class:** IT 230

**Module:** 3

|  |  |
| --- | --- |
| **1.** | Insert a copy of your of the ZIP file of all of your Visual Studio project files here so that it can be loaded and run in another Visual Studio:  *Attached Separately* |
| Insert here a copy of your \*.cs source code text you used here (copy and paste source code here, do **not** simply insert \*.cs files): |
| **2.** | Insert a screenshot here of the output that resulted from running your program, showing your last name as the first printed text to the screen: |
| **3.** | Explain the design of your program, the steps you took to complete it, and how you coded it:   1. First, I extracted the file. I copied and pasted it into a new solution title “TayDebugFixMethods” and pressed “Build Solution” 2. I found six bugs in the “Error List”: 3. While you can click through the error list to go through each error, I started with the first red line I saw (indicator of an error). That was in the WritePrompt(); line in the run function. It took me a second, but I realized that when the function is declared later on, it is written as Writeprompt. Since C# and most languages are case sensitive, this will cause an error. I changed the function to say WritePrompt in both areas. 4. The next red error underline is seen in the run function when an integer (choice) is being passed into the WriteChoice function. This directs me to the last function in the code, WriteChoice(). The code is written as ‘void WriteChoice(choice)’, which I changed to (int choice), to specify the data type in the parameter. I then changed Choice to choice in the WriteLine statement to match the casing of the parameter. Typically variables are lower case, so I choice to stick with the lower case choice. 5. After that, there was one last error in the error list regarding the inability to convert from ‘string’ to ‘int’. In the ReadChoice() function, I added Convert.ToInt32 to the return line. 6. At this point, there are no errors in the list. So, I ran the program to see if it matched the intended output. The program run successfully and matched what was asked in the instructions. 7. I added Grace Tay’s version in the Main function, to prove it is my output. 8. The first image in section 2 is what was due. I expanded on the program in the second screenshot. Unfortunately, there is a problem if you type in anything except for the numbers they expect, so I added TryParse and a while loop. |
| **4.** | Reflect on this experience and the lessons you learned from it:  Errors 1 and 3 indicate how important it is to watch capitalization. The smallest letter being off can cause issues. I had this error in a website I developed this weekend and it caused a headache and 404 error. Here, it can cause the program to not run. Errors 2 and 4 are more about data types and making sure the parameters you pass in or the data type you are returning are accurate. The biggest lesson was an expansion I did to the project. I tried the TryParse mentioned in the Discussion 3 grading feedback. You stated: “I would suggest using the TryParse method to check for a numeric value before you execute your convert method.  If the input is not numeric prompt the user for an additional value.” The if statement was not quite right, because if you typed in a letter a second time, the program didn’t run well. Instead, I have a while loop that will keep running until an integer is typed in. |