Lehrstuhl für Computergestützte Modellierung und Simulation
TUM School of Engineering and Design
Technical University of Munich

Professional Software Engineering

Assignment 2
Submission Deadline: 27th November

1 Instructions

In this assignment, a git repository https://gitlab.lrz.de/prose/assignment-2 is provided with a Visual Studio solution containing two projects -

- **SimpleShop**
- **SimpleShop.Test**

You need to fork this repository to your lrz-gitlab which can later be used to clone this to your local system. You can then open the solution(.sln) file in the Visual Studio to work on it. Final solution has to be uploaded on the **moodle**.
The assignment consists of 3 tasks - analysing code and commenting, writing NUnit tests, and programming methods, classes, etc. to fulfill given NUnit tests. These tasks are not ordered by difficulty level but subtasks within each task are provided with a rating, as comments, on a scale of 1 to 3 - 1 being easy and 3 being hard. Inside each subtask, the order is chronological. Start with the first task and work you way down.
Note: Keep your forked gitlab project access 'private' otherwise other people would be able to see your solutions.

2 Background

SimpleShop project implements an invoice generating program for a shop. It takes orders from an input TAG file by reading items, customer name, prices, etc. from valid tags, calculates VAT, discounts and total price depending on type of customer, and finally prints an invoice to the console in the end.
It parses tags from input file to custom struct *KeywordPair* which are then stored as data objects to be used inside the program.

Valid tags look like: *<Keyword>value</Keyword>*
Examples: *<ItemNumber>1</ItemNumber> <ItemName>Burger</ItemName>*

If both tags at the start and end of a keyword are valid, *ShopParser* creates a keyword pair. For instance in the above case, by reading keyword "ItemNumber" and storing it as key, and corresponding value as "1". Similarly, both tags for "ItemName" are also valid. So it also gets parsed into a keyword pair with value "Burger".

Invalid tag: *<CustomerName>James T. Kirk<CustomerName>*
ShopParser doesn't parse this because "/" is missing in the ending tag.

Note: When trying to run **SimpleShop** project directly from Visual Studio, the program returns
"This is not how you use this shop!"
This is because it doesn't have the reference to input TAG file. Therefore, you need to run it using a command line program (cmd or powershell). To do that, you first need to navigate to the *SimpleShop* folder and then specify a path to TAG file "SampleOrder.tag" provided inside the *SimpleShop.Test* folder after "dotnet run" command as follow:
dotnet run your_repo_directory_path \SimpleShop.Test \SampleOrder.tag

3 Tasks

Code Analysis

In the first part of this assignment, you have to analyse and understand the given code snippets. Two independent functions are provided inside the file **CodeSnippets.cs** in the project **SimpleShop.Test**. You are required to go through the code and reason about it. You should formulate a general idea about what each function is doing and when it should be used. Generally, this shouldn't be more than 50 words. Additionally, you should explain the code with comments.
Hint: Try to formulate the comments as short and concise as you see fit. While rough comments can be enough, you may find yourself wanting more steps for analysis. In this case, maybe, fine grained comments are the way to go.

NUnit Tests

In the file **WriteTestsHere.cs** in the project **SimpleShop.Test**, there are 12 logically empty NUnit tests (all are asserting a default Fail at the moment). Complete these tests such that all of these 'Pass' so that the code written in **SimpleShop** project is tested. You are **not** allowed to modify the signatures. Also, provide the comments on what is being tested.

Programming

Turning the things around, your goal in this task is to provide function, method and class implementations based on the given NUnit tests in the file **ToBeWritten.cs**. These implementations should be written inside the **SimpleShop** project.
Again, you are not allowed to modify the given signatures. However, you are allowed to add new classes, methods and functions. Use a step by step approach to pass one test after another. Also, pay attention to the comments provided in **SimpleShop.cs** (this is the main program file for **SimpleShop** project) and other project files.