Code documentation for "Checkpoint _1"

The task was to create a console program that would receive different article numbers with letters on the left side and then a hyphen and a number that is between 200-500. When you want to end the entry of articles, you must enter exit and the program must accept all forms of wording of this word such as ExIt. When you finish editing, a list of all article numbers in ascending order should be presented to the user before the program ends.

In the event of incorrect entries, the user must be notified with a warning text in red!

Has also chosen to set global parameters for different lengths on the two sides of the article number.

To avoid locking myself in too much with a predefined array with a number of fields, I have chosen to use List which is completely dynamic on this front!

The code itself is very self-explanatory in my opinion.

Code documentation for "Checkpoint _1_OOP"

I chose to make the project a little more advanced as I had a little to do, but at the same time wanted to challenge myself a little on this front. The principle is the same, but I have chosen to make a menu system and that the register is also saved as a text file in JSON format.

The program tries to follow the structure of MVC and at the same time be an OOP project.

In the Model directory, I have chosen to put all functions that process data and save this information in a global list based on my own defined class. Some functions also request information from the user so I do not know where it is right to actually put these functions. But there is some verification of the entered data so it feels quite right in my opinion. Others probably have different opinions, etc.

View contains functions that are primarily intended to present information to the user that is retrieved from the global list.

Control contains a specific function to verify the structure of the item number more served as SKU (stock keeping unit). There are a couple of global prams you can set for the structure of how this SHOULD be formatted.

Catalog Control:

File: ValidateSKU.cs

verifyArtId(string temp_string)

In parameter : Article id to validate syntax for.

Retrun type : Boolean

Verify input string if it validates all condition that are set in the code.

Catalog Model :

File: Product.cs

addProducts()

In parameter: NULL

Retrun type: NULL

Function for adding new products to our register. If the memory is empty, it checks if there is a local file on the computer and possibly loads it so that the user sees all the products that may be in their range.

The function also checks that we do not already have the same article number in the register!

deleteProduct ()

In parameter: NULL

Retrun type: NULL

Function to be able to delete a product from the register by entering these article numbers.

editProduct()

In parameter: NULL

Retrun type: NULL

Function for editing the name of a product (and other data) that exists in the register. If the product is not available, you can enter an existing number or end the function.

exitProgram ()

In parameter: NULL

Retrun type: NULL

Function to quit the program itself in a nice way, the user asks if it wants to save its register if this has not been done with changes

readArticles()

In parameter: NULL

Retrun type: NULL

Loads all products from a JSON encoded file into memory for further handling by the user!

saveRegisgter()

In parameter: NULL

Retrun type: NULL

Writes the registry to a JSON encoded text file.

Folder : View

File : Views.cs

showProductInfo()

In parameter : NULL

Retrun type : NULL

Displays detailed info about a specific SKU.

showProducts()

In parameter: NULL

Retrun type: NULL

Function for presenting all products in the register, primarily those sorted by article number and secondarily article names.

showMenu ()

In parameter: NULL

Retrun type: NULL

Function that renders a menu item in the console to the user, and only accepts valid selections. The menu options are in a List to easily adjust the order of the various menu options. However, there will be a bit of a problem with the Switch case, which will be a bit longer as you have to manually edit which functions are to be connected to the selected alternative.

Had it been a real product then I would have chosen to work against a database and possibly OOP. And made sure that it had worked on the web from the beginning and that it would work just as well on mobiles as desktops, as well as in several different languages and permissions in the system. But now the idea was just a simpler test to see if we have taken in different knowledge and so on.

Sorry for some misspelling, time booked at an optician. And not fully perfect program, would do more testing if if it would be a real product.

/ Anders Wallin 2021-12-27