Assignment 4 – Interfaces, Delegates, Events and Menus

Objectives

- OOP and Poymorphism
- Utilizing interfaces
- Working with delegates / Action<T>

Prior Knowledge

- Working with intrefaces
- Working with delegates and Action<T>
- Using collections
- Working with numerous projects

Excercise

You are asked to implement a class that will help a programmer to display and manage an hierarchical Menu for his console-based applications.
(For example, utilizing such a class, if existed, would have saved you implementing the Menu in assignment 3)

This class should provide a way for the programmer that uses this class (for example, the programmer that implements the Garage Management System from assignment 3), to build a menu for his needs, by defining/creating Menu items in the main menu, or as sub-items for any menu item.

Please name this class MainMenu.

The application in which we want to display a text-based menu will hold a reference to an object of type MainMenu.

An example for an hierarchical menu would be the top menu of Visual Studio.
The main window of Visual Studio displays a top Main Menu, which contains several menu items (File, Edit, View) which themselves hold sub-items. Cilcking some of the items results in executing an operation in Visual Studio (like "Build"). Clicking on others results in displaying the sub-items of that menu items and so on..

The MainMenu class will have a method called Show(). Calling display the first level of the menu items, and will practically start the main loop of the console-based application in which the user will be asked to input his choice of action until quiting.

In each menu level –

1. The current menu items of the current menu level will be displayed to the user
2. The user will be prompted to choose one of the options displayed in the menu
3. The user's input will be handled
4. The user's input will be validated, and the user will be prompted on any invalid/illegal input
5. Navigating to a sub-menu / Executing an action
   a. If the user chooses a menu-item which is a 'parent' of sub-items, the console will be cleared, and the sub-items will be presented to the user as the new current menu level, with the parent item as the title of this menu level and practically back to (1.)
   b. If the user chooses a menu-item which doesn't contain any sub-items, will result in executing the method in the system that this menu-item is

representing (i.e. Printing the list of vehicles in the garage). Choosing this type of menu-item will also first clean the console, and then notifying the system that the revelvant operation should be executed (the UI of the relevant operation is executed by the system, not by the menu component). When the operation execution is done, the current level of menu-items will be re-displayed an so on.

c. In each level, the user can choose a menu item that takes him back to previuos menu level, or that ends the menu loop if it is the top level.

The menu always displays the following:

- Title (in the first level, it's the main title of the menu, and in any sub-level, the title is the title of the parent menu item)
- A list of menu items of the current level (numbered. 1 based).
- A special menu item, numbered 0, titled "Back" ("Exit" if it's the top level)
- A prompt to the user that asks from a number 1… or 0 (for "back/exit")
- For example:

```
Drinks Machine
==============
1. Tea
2. Coffee
3. Juice
0. Exit
Please enter your choice (1-3 or 0 to exit):
>> 2
```

`<Console.Clear()>`

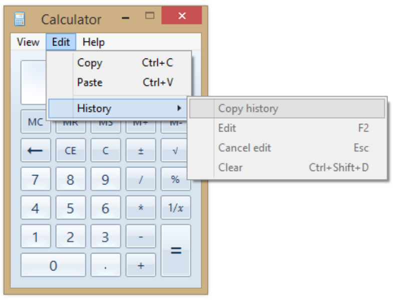
```
2. Coffee
==============
1. Americano
2. Espresso
3. Capuchino
4. Late
0. Back
Please enter your choice (1-3 or 0 to exit):
>> 3
```

<< the system, that uses you MainMenu class, will now display the user the relvant UI to make an espresso (amount of sugar, milk type, etc.) and upon completion will re-dislpay the last level>>

```
2. Coffee
==============
1. Americano
2. Espresso
3. Capuchino
4. Late
0. Back
Please enter your choice (1-3 or 0 to exit):
>> 3
```

Further elaboration

The MainMenu class should allow the programmer who uses it (for example, the programmer who implements the Garage Management System) to build a console-based hierarchical menu system for the needs of his application, with first-level menu items and sub-items for some of these items, instead of implementing this console-based hierarchical menu by him self.
The way to achieve this, is by holding a collection of menu items in the MainMenu class, and potentially holding a collection of menu items by each of these items, and so on.
In the system that utilizes this MainMenu class, you should have a method that build and initializes the menu structure of the system.
The operations that should be exdecuted upon selecting menu-items in the menu (the 'leaves' in the menu tree), should be executed by the relevant menu items.
Take for example the menu of the Calculator app (in this case: 3 Menu items in the first level, 3 sub-items that are the 'children' of the 2nd item, 4 sub-items that are the 'children' of the 3nd item etc.):



The Task

Your mission is to implement twice, once by utilizing interfaces, and once by utilizing delegates, the two techniques that solves the "employee notifies that he is sick" scenario. (Hint: A menu item is the employee, the garage management system is the employer).

1. **Using interfaces:**
   Which interface will be defined? Which operation(s) will it declare? Where (in which assembly) should you define that interface? What should be its access level? Who will implement it? Who will reference and use it?
2. **Using delegates:**
   Which delegate will be defined? In which project? Who will hold it an invoke it?

Your Visual Studio solution should have 3 projects:

1. **Ex04.Menus.Interfaces**
   This project will be the implementation using interfaces, and will be of output type of class library (.dll) and not an application (.exe)
2. **Ex04.Menus.Delegates**
   This project will be the implementation using delegates, and will be of output type of class library (.dll) and not an application (.exe)
3. **Ex04.Menus.Test**
   This project will be an implementation of an application (.exe) that demonstrates the use of the above two projects. The Main method of this exe is defined in the next page:

In the Main method of the test application you need to create **TWO** menus (one for each one of techniques), with 3 levels each.
The first menu will be displayed to the user which will navigate and operate it.
When the user will choose the 'Exit' option of that first menu, The second menu will be displayed to the user.

This will be the specification of the menus:
The top level will contain 2 menu items:

1. " Version and Spaces"
   A menu item which upon selection will display two sub-items:
   a. An action menu item titled: "Show Version"
      Upon selection will execute an operation in the Test project that will display the following text to the user: "App Version: 21.1.4.8930"
   b. An action menu item titled: "Count Spaces"
      Upon selection will execute an operation in the Test project that will ask the user to enter a text input and will output the number of spaces in the text.
2. "Show Date/Time"
   A menu item which upon selection will display two sub-items:
   a. An action menu item titled: "Show Time"
      Upon selection will execute an operation in the Test project that will display the current time to the user.
   b. An action menu item titled: "Show Date"
      Upon selection will execute an operation in the Test project that will display the current date to the user.

Note:
You are not expected to avoid code-duplications between the two .dll projects.
Treat them as two diffrenet assignments for this manner.

Please Note:

1. Design your system in the best manner of code, regarding dividing to classes and methods, inheritance and polymorphism.
2. These classes implement **console-based** menu system, therfor may indeed write/read to/from the console.
3. You may ask questions in the course's private Facebook group regarding the assignment.
4. You must comply with the coding standards, as stated in the relevant document, found on the course website. **Pay extra attention to the naming conventions regarding delegates and events**. Points will be deducted to whom ever does not comply with these standards.
5. Avoid cheating (Do not use other students assignments as a basis for yours. Refrain from copying the work of fellow students from your group or previous semesters. Cheaters will be caught and punished. Work independently!)

Good Luck!