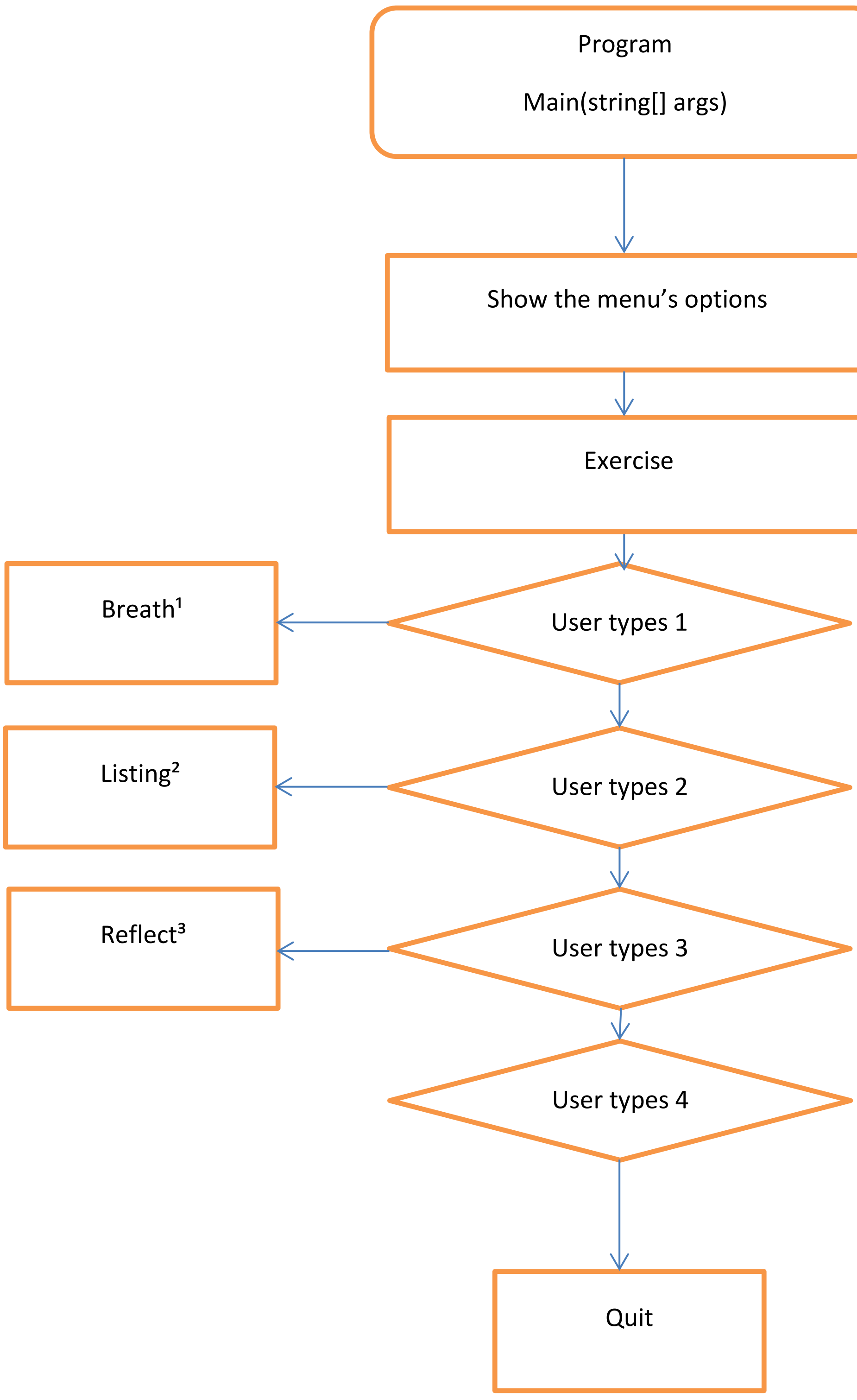
Week 07 Unit 04 Activity Design



The flowchart above represents of how the mindfulness program is supposed to work.

[1]The breath layer works with two private strings: _breathIn and _breathOut, also, it has a constructor method, called Breath, with two other methos: GetReady() and DoBreathing, both being void.

[2]The Listing Layer works with a string called _message, which stores the standard message, gathered with a List<string> _prompts that will store the questions. And just like the breath layer, has a constructor method, with only one method, called DoListing, which is void.

[3]The Reflect Layer works with two strings: _prompts and _currentPrompt, with the string _prompt being an array of it. Also has a constructor method, and only one method called DoReflection, which is void.

Program Flow:

When the program is played, it shows a menu with four options: breathe exercise (1), reflection exercise (2), listing exercise (3) and quit (4).

When the user types 1, it shows a message asking how many seconds he/she wants to spend in this exercise (the same happens for reflection and listing exercise), after it, instructions like when the user should breathe in and breathe out appears, with this happening until the time is over, and after each breathe in and breathe out, the console shows how many time left until the end of the exercise.

When the user types 2, some instructions related to thinking appears in the console, after some seconds, follow-up questions appears asking the user to think about what he/she thought.

When the user types 3, it shows a random question for the user answer, then, the program gives the option to the user to save both the question and the answer in a specific file.

All the layers cited above, inherit some attributes, like time countdown, the starting message of each exercise and the final message of the Exercise layer.

In C#, there are three ways of how to work with codes:

Inheritance[1];

Encapsulation[2];

Abstraction[3];

[1]Inheritance is a good tool when you want to create a new class from an existing class. This is very useful when you want to practice reusability in C#, also, it is a key feature of OOP (Object-Oriented Programming).

It is not so difficult to use Inheritance, especially when you don't want to waste your time typing the same code multiple times.

For example, there is a class called A, and you create a class called B, but you want to work with the attributes of the class A in class B, so, you can do by this way: B : A, this two points means that everything in class A will be inherited by class B.

[2]Encapsulation in C# can be defined as a method of preventing unauthorized access to certain kinds of data.

For example, you want to make more secure your code you can do encapsulation through the accessing methods, which can be private, protected, public and internal.

Also, encapsulation works with getters and setters, with the getters being responsible of returning a specific value, and the setters for defining a specific value.

[3]Abstraction is the process of hide the critical part of the code and show only the required information for the user.

And to finish, when you create an abstract class, you are simply doing abstraction in it.

Have you understood? Programming can be fun when you understand it.