SIT771 Object Oriented Development

Pass Task 5.2: Lists

Overview

In this task you will create a simple program which will create a List, and allow the user of the program to manipulate the values stored in the list.

Submission Details

Submit the following files to OnTrack.

- The Program Code (*Program.cs*)
- A screenshot of the running program

You want to focus on lists, loops, and the different techniques we have to manipulate lists.

Instructions

We're going to create a program which will allow the user to add values to a list, print the values, and print the sum of all the values. Part of this program will be a simple user interface.

1. In the `Program` class, add a static list of doubles:

   ```
   private static List<double> _values = new List<double>();
   ```

   this is the list of doubles we'll be working with in the program.

2. Add `ReadInteger(string prompt)`, and `ReadDouble(string prompt)` to the Program class. These should be identical to those in 5.1P.

3. Add a public enum named `UserOption`. Give it the values/options of: `NewValue`, `Sum`, `Print` and `Quit`. This enumeration will be used later on in the program to create the user interface menu.

   ```
   public enum UserOption
   {
     // values
   }
   ```

4. One of the actions our user will be able to take is to add a new value into the list. Let's create the method that will do this. Here is something to get you started:

```
public static void AddValueToList()
{
  // code goes here
}
```

Remember, we can use the Add() method on the List class to add items to a list, and we can use the `ReadDouble()` method to get a value from the user.

5. The second action our program can perform is printing values to the screen. To do this, create another `public static void` method named `Print()`. In Print(), use a foreach loop to iterate over `_values`, printing each value to the console.

6. Finally, we need to be able to *sum* the values. This is yet another `public static void` method, which should do the following:

    - Create a local double variable called `sum` and initialise it to 0.
    - Use a foreach loop to iterate over each element in `_values`.
    - Each iteration, *add* the value in each element to the sum.
    - Write `sum` to the console.

7. Now that we have the functionality of our list implemented in our code, let's add the code which will allow the user to interact with the program!

    - Let's take a look at the ReadUserOption() method:

```
public static UserOption ReadUserOption()
{
  Console.WriteLine("Enter 0 to add a value");
  Console.WriteLine("Enter 1 to add a sum all values");
  Console.WriteLine("Enter 2 to print a sum all values");
  Console.WriteLine("Enter 3 to quit");

  int option = 3;
  Int32.TryParse(Console.ReadLine(), out option);

  return (UserOption)option;
}
```

You can see it simply presents the possible options to the user, and then reads the option from the user. Note that by default, the method will return `3` (exit).

    - Create the `Main()` method. The Main method should loop until the user opts to quit, each time it should `switch (ReadUserOption())` and use the returned `UserOption` value to call the appropriate method.

8. Create a screenshot of your program running, and upload it to OnTrack along with the code files for this task.!