CSCI 1301 – Lab 13

September 27, 2018

1 Code From Today's Lecture

Please find in this archive[1] the code from today's lecture, commented. It contains a couple of questions, make sure you know how to answer them before moving on.

2 Writing A Tip Calculator

Your friend learned that you now know how to program, and would like to know if you could develop the following program:

> The idea would be that your program would ask me how much the check is, how many we are, and what percentage we want to leave as a tip. Your program would then print how much we need to pay (assuming we split equitably).
>
> What would be great is if you could round things up (I often pay cash, so I don't want to have to carry small change with me): so for instance, if we are two and the check is \$23, plus 15% tip, $\$23 + (\$23 \times 0.15) = \$26.45$, we need to pay \$13.225 each, but I'd rather have us both pay \$14 and leave a small extra tip. Oh, and can you make it look nice, and display all the information at the screen? How much we left as a tip, what percentage it makes, etc.

2.1 Thinking

Think about your friend's demands, and how you could write this program (for now, you can think of it as being entirely in a `Main` method, you do not have to construct a class). Write on a piece of paper the steps needed. The general structure should be:

1. Gather the data
2. Compute
3. Display the result of the computation

Try to decompose those three steps further: what messages you need to display at the screen to gather the data, how many numbers, or variables, you need to manipulates, what is the exact computation, what you need to display at the last step, etc. Make sure you understand how to do the computation, without thinking about the tools you actually have in C#.

2.2 Coding

Now is time to think about the actual code you need to write. You should have all the tools you need to complete the steps you identified at the previous exercise. The "rounding part" could be problematic, but you can use the `Math` class, and more specifically the `Math.Ceiling` method: if you give to that method as an argument a floating point value, it will "round it up" (using the same datatype: that is, `Math.Ceiling(2.3)` will return `3.0`, i.e. *a float*).

[1] DetailledAccount.zip