**Lab 02 - Calculators**

**Jacob Heslop and Richard McCormick**

**September 19th, 2019**

**Section 03**

**Problem Statement**

For this lab, we were tasked with creating two program files, a Time Calculator and Tax Calculator. Each of these programs has different requirements. The Time Calculator converts an amount of seconds into seconds, minutes, hours, and days, and the Tax Calculator tells you how much in taxes you have to pay based on your marital status and income.

**Planning**

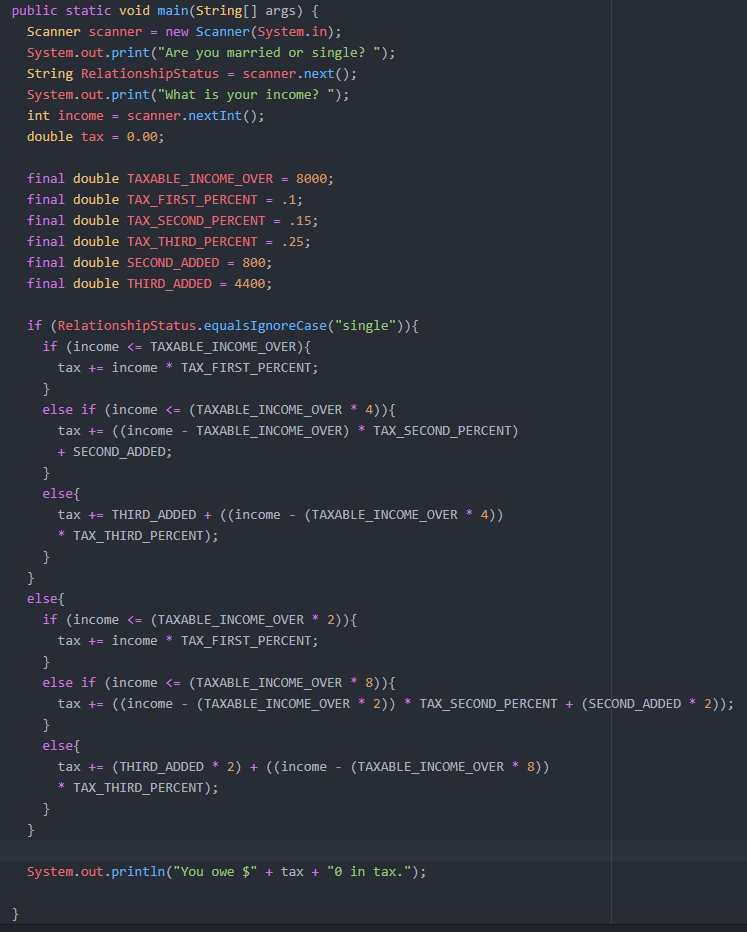
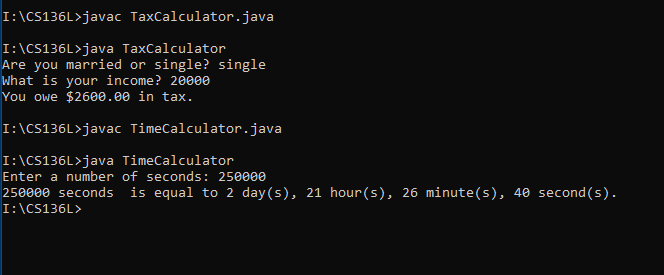
We started planning for this lab by completing the prelabs, and creating a written drawing of what we wanted our program to accomplish. This includes mapping out each possible path the program could follow, and anticipating what we would want to happen at each turn. This allows us to start programming with a map of where we want to go. The Time Calculator planning involved simply determining the number of seconds in a day, an hour, and a minute. For the Tax Calculator, we needed to determine each tax bracket, and it’s corresponding tax rate.

Once our map was complete, we began transferring our ideas into code.

**Implementation**

We started our program with the Time Calculator. To accomplish the program’s desired outputs, we used a series of if statements that would reduce the number of seconds by the largest denomination possible (i.e., 86,400 seconds, or one day) and then calculate the remaining number of hours, minutes, and seconds. If there is not enough seconds to subtract out the largest denomination, then the program selects the next largest denomination (3,600 seconds in an hour). This continues down to the minute and finally the second(s) level.

Part two of the lab is the Tax Calculator. This was slightly more difficult, as there are more conditions which had to be checked at each turn. We started by dividing the problem into two main if statements: single and married. This allowed us to write a working code block for the single status, which correctly calculated the tax rate at each income level. This working code block was then copied under the married status. The copied code is then tweaked so that the variables within are twice that of the single status, making it possible to avoid re-writing the entire statement.



**Refactoring**

We had to revisit our map for the Time Calculator. We had originally planned on setting up while loops which for each step would subtract the correct amount from the last one and add one to the next highest (subtracting 60 minutes from Minutes and adding an hour to Hour until minutes was less than 60). When the code was run we kept running into errors so we switched to if/else if statements and dividing remaining seconds. In much later hindsight, our problem probably occurred because we were trying to use the += function on variables that we had not given a value to but had called.

A major refactor we had to make in the Tax Calculator lab was one we did not expect to encounter at all. The program was not correctly summing the tax for Married status, and appeared to be going over the desired amount significantly. We spent nearly thirty minutes trying to find out what the cause was, before finally realizing that we had a semicolon at the end of the first if statement, causing both statements to run when only the second one was needed. By removing the semicolon, our program resumed functioning properly.