COMP 248

ASSIGNMENT 1

DESCRIPTION: In this first assignment, you will have to write two fairly simple programs. The first will focus on the material from Chapter 1, while the second will build on the material from Chapter 2. Note that you will be graded both on the correctness of the code (i.e., it must compile and produce the appropriate results), as well as on the use of the style and programming conventions discussed in class.

PROGRAM 1: We will assume that you are the editor for a book of poetry. Today, you will be working on a short poem by some guy name Shakespeare. The poem goes something like this:

Shall I compare thee to a summer's day
Thou art more lovely and more temperate
Rough winds do shake the darling buds of May
And summer's lease hath all too short a date

Your job is to (i) provide some basic statistics on the poem, and (ii) to personalize it a little. You find this stuff terribly challenging so you decide to write a Java program to do it for you. So you will do the following two things:

1. You will calculate the number of characters in this poem, as well as the average number of characters per line.
2. You will substitute the word "thee" in the first line with your own name.

You will then print your work to the screen. It will look something like this (assuming your name is "Sammy"):

```
Poem analysis:
The poem has 165 characters
The average length per line is 41.25 characters

The new poem is:
Shall I compare Sammy to a summer's day
Thou art more lovely and more temperate
Rough winds do shake the darling buds of May
And summer's lease hath all too short a date
```

There are a few things to keep in mind.

1. You want your code to be flexible so that it works for other cases. So your poem, plus the old string ("thee"), the new string (your name), and the number of lines in the poem, may be different in the future. We don't want to have to search through your code to change these things in the future so you should define them in one place as constants. (The marker may also want to change these things to make sure that your code still works).
2. In a more powerful program, we would use a *loop* construct to move through the poem line by line. We haven't covered loops yet, so you should treat each line of the poem as a separate string and process it accordingly.
3. As discussed in class, the String class has many methods for String processing. You may notice that the String class contains a "replace" method that could be used here. However, we have not discussed the *regex* (i.e., Regular Expression) parameter that this method uses. We have also mentioned that the StringBuffer class can be used for modifying Strings but we haven't covered that in detail either. So the bottom line is this: You will only use the String class for the text in the poem, and you will only use the methods presented in the class slides (Chapter 1, Section 2). In short, you must identify the part of the String that you want to replace and then extract the remaining pieces. You will then reassemble these parts, along with the new text, into a new String.

PROGRAM 2: You have found a summer job working at a pizza shop. Your primary function is to accept orders and return the proper change to customers. Your calculator is broken and you never learned how to use the cash register, so you have decided that two weeks of Java instruction has prepared you to write a program to do this for you.

In this case, of course, you will need to obtain some input. Basically, you need to know what the customer has ordered. You have three options: Slice #1 (Cheese, $1.99/slice), Slice #2 (All Dressed, $2.99/slice), or Slice #3 (Vegetarian, $2:49/slice). Some people are hungrier than others so you also have to know how many slices of each have been ordered.

Once you know this, you can calculate the costs. Basically, you will add up the cost of the order, then add the tax – currently set by the Quebec Pizza Monitoring Agency at 43.5% - to get a final result. You will then have to make change for the customer. In short, they will give you a bill and you will return the change. Of course, you need some kind of receipt so you will have to show your calculations.

Input will be required from the keyboard. Because we haven't talked about conditional statements – that could be used to distinguish between the pizza types – we will use a simpler approach. In short, you will enter the number of slices for each type (0 is a possible entry). Once the entries are made, a simple receipt is displayed on the screen (you don't have to worry about printing anything). Then the customer will give you a bill (i.e., an integer representing the number of dollars). You then enter the bill amount and, voila, your change is calculated. The whole process might look like this:

```
How many Cheese slices do you want?
1
How many All Dressed slices do you want?
2
How many Vege slices do you want?
3



 ** Receipt **

Type of Slice                  Quantity   Price      Total
Cheese                         1          1.99       1.99
All Dressed                    2          2.99       5.98
Vege                           3          2.49       7.47

SubTotal                                             15.44
QPMA Tax                                             6.72
Total                                                22.08

Money given to you by customer?
 30


** Change **
Total: 22.08
Money from customer: 30
Change: 7.92
 7 x 1.00
 3 x .25
 1 x .10
 1 x .5
 2 x .1
```

A few notes are in order.

1. As was the case for Program 1, both the price per slice and the tax rate could change in the future so this should be easy to update.
2. There is a lot of formatting here so you are free to use any of the techniques discussed in class.
3. Change is listed as a dollar amount, plus the number of quarters, dimes, nickels, and pennies.
4. Because we have no ability just yet to do any kind of error checking, we will assume that the bill given to you by the customer is always large enough to cover the total cost (otherwise you would smack the customer and ask for a bigger bill).
5. In practice, we might use the functions in the MATH library when doing calculations of this sort. But we haven't talked about that and it is not to be used in this assignment. Instead, you will use the techniques discussed in class to do the calculations (defining the right types, assignment statements, type casting, parentheses, etc). This is one of the main objectives of this program.

6. Because we aren't using the Math libraries, there may be small rounding errors in the calculations (e.g., off by one cent). That's okay since these things could easily be corrected in a real program.

So that's it. The programs are not that large…perhaps less than 100 lines for the two programs combined. In fact, my assignment description is longer than the code that you will write. But you have to think about the problems a little bit, especially the second one, since the solution involves more than just getting the syntax right.

DELIVERABLES: You will submit two sources files, one for each question. NO other files should be submitted! For grading purposes, the marker will just compile your source files and run them to check the output. The files will be named **Q1.java** and **Q2.java**. Each file will contain one class, with the same name as the file. Each class will have one method called **main**. Once you are finished writing your code, you will combine the two source files into a single zip file called:

A1_name_ID.zip

where "name" is your name and "ID" is your student ID. So if your name is John Smith and your Student ID is 12345, then you would create a zip file called

A1_John_Smith_12345.zip

The file should be uploaded to the Moodle server via the Assignment 1 link on the web page. You are free to submit multiple versions (new versions just overwrite the last one) until the deadline. You may still submit after the deadline but the server will record your submission time and an appropriate penalty will be assessed (as per the syllabus). Absolutely NO submission will be accepted by email.

NOTE: It is your responsibility to verify that your assignment has been properly submitted. So please check the contents of your submission after you upload it to make sure that it is valid.

Good Luck!