2022 Fall -- CSCI 1301L
Lab 02: Basic Computations

Introduction

This lab covers the knowledge of basic computation in Java. You will create a Java program that can compute the pay and taxes based on inputted information. The program should give the output of computed results.

Lab objectives

After completing this lab, you will be able to:
(1) declare and initialize variables of different types,
(2) explain the need and process multiple whole number and floating-point data types,
(3) calculate the resulting value from arithmetic expressions, and
(4) perform basic input and output operations.

Assignment

Part-1:

Create a class called **NetPay** in a file, **NetPay.java**. This file is used to compute a person’s gross and net pay based on their hourly wage, working hours, and tax withholdings.

All statements should be defined in the **main** method of the class. The only exception is that you may (but are not required to) declare constants before the main method. Note that you will make sure that all inputs and outputs are consistent with the provided examples. For this purpose, you should write and compile your Java source code in our designated environment, which is with the latest version of Eclipse, JDK 11, in either a Microsoft Windows or a MacOS operating system.

The output of your program should look like this:

```
Hours per Week:     40
Gross Pay:          290.0
Net Pay:            225.765

Deductions
Federal:            29.0
State:              13.05
Social Security:    17.98
Medicare:           4.205
```

In order to ensure that your program output is consistent with the example given above, we suggest you consider to make the outputs line up on the right-hand side, use the tab character '\t' in your print statements. Note that you may need different numbers of '\t' characters in each line.

For the calculation, *Net Pay* is calculated as the result from *Gross Pay* minus all deductions (Federal, State, Social Security, and Medicare). The *number of hours per week* will be used to compute *Gross pay, Net pay, and Deductions*.

You use the most appropriate data type for each variable.

Hours per week is your input to the program. For Part-1, you will type it into the program's source code. For Part-2, such a value will come from the user's keyboard input.

For this assignment, you will declare the following named constants and set their values as follows:

```
FEDERAL_TAX_PERCENT: 10.00
STATE_TAX_PERCENT: 4.5
SOCIAL_SECURITY_PERCENT: 6.2
MEDICARE_PERCENT: 1.45
PAY_PER_HOUR: 7.25
```

Pay attention that the first four value are "percentage" values. To calculate deductions, you will use the *Gross Pay* multiplying the percentage (xx%) to get these results.

When you finish the program all above, you will change the value of *number of hours per week* to some different numbers within the code. For example, when you change it to 40, we expect the following output:

```
Hours per Week:     40
Gross Pay:          290.0
Net Pay:            225.765

Deductions
Federal:            29.0
State:              13.05
Social Security:    17.98
Medicare:           4.205
```

Part-2:

Modify **NetPay.java** to make the program accept user's input for the *number of hours per week*. in the example below. The value 100 shown in red is from the user's keyboard input. This means that in your source code, we MUST NOT see an assigned value of 100 in the variable *number of hours per week*.

```
Hours per Week:     100
Gross Pay:          725.0
Net Pay:            564.4125

Deductions
Federal:            72.5
State:              32.625
Social Security:    44.95
Medicare:           10.5125
```

Since your revised program is going to accept a user's keyboard input, think what additional codes will be added and what Java classes will be loaded and used. For the output, a user should first see "Hours per Week" in the output. Then the user will be able to input a number in the same line following "Hours per Week". Once the user finishes its input of the number and click enter, the remaining part of the output will show up onto the terminal screen.

Before you make the assignment submission, you should test your program with different inputs. At this stage, you do not have to consider negative numbers, invalid inputs such as characters, very large numbers such as 99999999999, etc. We only consider common scenarios in the real world. That said, your source code must be able to compile and there should not be any syntax error in the code. Your program may have a different number of digits after the decimal point (e.g. 564.4124999999999). This difference is due to the imprecision of decimal point numbers, but it is acceptable in your submission as long as such a difference is <0.01.

At last, understand the following statement for Academic Honesty and add it into the top of your source code to submit. The following lines should be added ABOVE the original first line of code.

```
/*
 * [CLASS/FILE NAME].java
 * Author: [YOUR NAME]
 * Statement of Academic Honesty:
 *
 * The following code represents my own work. I have neither
 * received nor given inappropriate assistance. I have not copied
 * or modified code from anywhere other than the authorized
 * sources. I recognize that any unauthorized sharing, assistance,
 * or plagiarism will be handled in accordance with both the
 * University of Georgia's Academic Honesty Policy and the
 * policies of this course. I recognize that my work is based on
 * an assignment created by the Department of Computer
 * Science at the University of Georgia. Any publishing or posting
 * of source code at any time for this project is prohibited.
 */
```

Replace [CLASS/FILE NAME] with the required name according to the assignment instruction.
Replace [YOUR NAME] with your actual name.

Submission instruction

After you have completed both Part-1 and Part-2 and thoroughly tested your program, upload and submit the file *NetPay.java* reflecting **the completion of Part-2** to eLC. Do not submit Part-1 of the assignment as Part-1 is only a middle step for this assignment.

Grading

A score between 0 and 10 will be assigned.

- You may earn 3 points for lab attendance.
- You may earn 7 points if your program passes various test cases that we use for grading. Some test cases will use values taken from the examples in this document and **some test cases will not**. You should come up with additional test cases to check that your program is bug free.

(1) Points are AWARDED when your program passes at least 4 test cases that are determined or chosen by the grader/grading program.

(2) Points are DEDUCTED when your program fails to meet certain requirements:

- (-1 point) If the source file(s)/class(es) are named incorrectly (case matters!)
- (-1 point) If your source file(s) have a package declaration at the top
- (-1 point) If any source file you submit is missing your Statement of Academic Honesty at the top of the source file. All submitted source code files must contain your Statement of Academic Honesty at the top of each file.
- (-1 point) If you have more than one instance of Scanner in your program. Your program should only have one instance of Scanner.
- (-1 point) Inconsistent I/O (input/output) that does not match our instructions or examples exactly (unless otherwise stated in an assignment's instructions). Your program's I/O (order, wording, formatting, etc.) must match our examples and instructions.
- If your (-1 point) comments or (-1 point) variables are "lacking". "Lacking" means that your grader can find any lines of code or variables that take more than 10 seconds to understand, and there is no comment, or the variable name does not make sense (variable names like b, bb, bbb, etc. will almost never be acceptable).
- (-1 point) Indentation is not consistent throughout your source code (e.g. you should NOT use a combination of tabs and spaces in the files.)

Special notice regarding the submission:

A. Late submission penalty. Points will be deducted from the original grade. If your submission is after the posted deadline…

1. within 48 hours: -2
2. after 48 hours: assignment will not be accepted.

B. If the source code file is missing, partially or completely broken, unable to open (including using a wrong file format), unable to compile, irrelevant to the assignment, purposely made to contain malicious codes, or determined to be an academic misrepresentation or a case of plagiarism, you will receive 0 grade for this assignment.