**CSCI 1301: Introduction to Computing and Programming**
**Lab 02 – Computing Taxes and Pay**

## Recommended Group Brainstorm (NO computers during this time)

Good programmers think before they begin coding. Part I of this assignment involves brainstorming with a group of peers with no computers to talk about a strategy for solving this week's lab. Breakup into groups based on your seating (3-4 people per group) and brainstorm about how to solve the problems below. Make sure everyone understands the problem and sketch out potential ways to move toward a solution. You may find it helpful to look over the required readings for this week.

**Note: Brainstorms are to help you get started with the lab and get you in the habit of designing before you code. You won't submit them to eLC.**

## Exercise I - Computing Taxes and Pay

Create a class called **`NetPay`** to compute a person's gross and net pay based on their hourly wage, hours worked, and several withholdings. All statements should be defined in the **`main`** method of the class (except for declarations of constants).

The output of the program should look similar to 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
```

Implementation Details:

- To make the outputs line up on the right-hand side, you will need to use the tab character '\t' in your print statements. You may need one or more '\t' characters in each line. Play around with it until the output lines up.
- You will use the number of hours per week to compute the gross pay, net pay, and the deductions. You will need to create a variable for each value above. Brainstorm the proper data types.
- Hours per week is your input to the program. For exercise 1, you will type it into the program. For exercise 2 (below), it will come from the keyboard.
- You will also need to declare named constants and initialize them to the values shown.
  - **`FEDERAL_TAX_PERCENT`**: 10.00.
  - **`STATE_TAX_PERCENT`**: 4.5.
  - **`SS_PERCENT`**: 6.2.
  - **`MEDICARE_PERCENT`** 1.45.
  - **`PAY_PER_HOUR`** 7.25.

Questions to consider:

- How do we compute gross pay? Store the result in the variable created for gross pay.

- How do we compute federal tax? Think about using gross pay and federal tax percent.
- Compute state tax, social security contribution, and medicare using similar methods as federal tax.
- How do we compute net pay?
- Display the values on the screen in the order listed above.

Once your program is implemented, compile and run your program. Afterwards, edit your program by changing the value assigned to the variable for hours per week. Do this in the code for now. We will do user input in part 2. Additional sample executions are shown below. Your program may have more digits after the decimal point shown than the examples below, and that is okay as long as your program's output is within 0.01 of the answer shown.

```
Hours per Week:     30
Gross Pay:          217.5
Net Pay:            169.32375

Deductions
Federal:            21.75
State:              9.7875
Social Security:    13.485
Medicare:           3.15375
```

---

```
Hours per Week:     35
Gross Pay:          253.75
Net Pay:            197.544375

Deductions
Federal:            25.375
State:              11.41875
Social Security:    15.7325
Medicare:           3.679375
```

---

```
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
```

---

```
Hours per Week:     45
Gross Pay:          326.25
Net Pay:            253.985625

Deductions
Federal:            32.625
State:              14.68125
Social Security:    20.2275
Medicare:           4.730625
```

---

### Exercise 2:

Modify **`NetPay.java`** to **ask the user for the number of hours worked** (it is no longer typed into the program). Note: you do not need to submit exercise 1 – only submit a single java file after completing both exercises.

- Which preexisting Java class did we use to read in keyboard input? Reference the textbook or class slides and borrow the necessary code from there (you don't have to memorize this code).
- Don't forget to prompt the user to enter the number of hours per week.

Compile and run your program. A run of your program should look like the example below (**the blue (bold) is the user's input – it is not being output by the program**). Your program may have more digits after the decimal point shown than the examples below, and that is okay as long as your program's output is within 0.01 of the answer shown.

```
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
```

You should run your program with various inputs, and test that your program is producing correct outputs. For every programming lab and project this semester, you are responsible for ensuring that your program doesn't contain any syntax, runtime, or logical errors. Always test, test, and retest the programs you create.

## Statement of Academic Honesty Requirement

You must include the Statement of Academic Honesty comment at the top of the program's source file. Copy and agree to the entirety of the text contained in the file **StatementOfAcademicHonesty.txt** on the Labs and Projects webpage, and fill in the class name of your **`.java`** file, your name, submission date, and the program's purpose. In the future, every Java source file of every lab and project you submit **must** have a comment such as this at the top of every source file. Otherwise, points will be deducted from your assignment.

## Additional Requirements

These are things that make the graders lives easier, and ultimately, you will see in the real world as well. Remember the teaching staff does not want to touch your code after they gave you requirements; they want to see the perfect results they asked for! Here is a checklist of things you can **lose points** for:

- (10 points) If the source file(s)/class(es) are named incorrectly (case matters!)

- (10 points) If your source file(s) have a package declaration at the top
- (10 points) 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.
- (10 points) If you have more than one instance of Scanner in your program. Your program should only have one instance of Scanner.
- (15 points) 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.
- (100 points) If the source file(s) are not submitted before the specified deadline's late period ends (48 hours after the deadline) or if they do not compile.
- (25 points) Late penalties will be deducted as per the course syllabus.
- If your (10 points) comments or (10 points) variables are "lacking"
    - Here, "lacking" means that you or a TA 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)**
- (10 points) Indentation is not consistent throughout your source code
    - Refresh your memory of indentation patterns in chapter 2 in the course textbook
    - Be careful of a combination of tabs and spaces in your files (use one or the other)!

If any of the above do not make sense to you, talk to a TA, or ask on Piazza!

## eLC Submission and Grading

After you have completed and thoroughly tested your program, upload and submit **`NetPay.java`** (not the .class file – that's the byte-code) to ***eLC*** in the Lab 2 Assignment Dropbox. Always double check that your submission was successful on ***eLC*** and double check that you submitted the correct version of your completed program.

The lab will be graded according to the following guidelines.

- A score between 0 and 100 will be assigned.
- If the source file(s) are not submitted before the specified deadline's late period ends (48 hours after the deadline) or if they do not compile, then a grade of 0 will be assigned.
- 25 points will be deducted for labs submitted late (note: labs can only be submitted up to 48 hours late, after 48 hours, a grade of zero will be assigned).
- The programs will be evaluated using various test cases. **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. Feel free to share any tricky test cases you find on Piazza. Both the calculations and formatting of the output must be correct in order to receive credit for each test case.
- Points will be deducted for not following any of the aforementioned instructions or requirements.