CPSC 1060/1061: Introduction to Programming in Java

Lab 02 – Variables and Primitive Data Types; Basic Input/Output

Introduction

Variables in Java are memory locations that have been given names in the program and which can be assigned values. Each variable can only store values of a given data type, and the type of the variable must be declared (specified) before the variable is used. A variable's type can be one of Java's 8 primitive data types (**`boolean`**, **`byte`**, **`short`**, **`int`**, **`long`**, **`float`**, **`double`**, **`char`**), or it can be a class (the latter are sometimes called *reference types*).

In this lab, you will practice working with variables and Java's primitive data types. You will declare variables, initialize them, and make assignments to them. You will also practice working with arithmetic expressions (and assigning the results to variables), writing formatted text to the console (**`System.out`**), and reading formatted input from the keyboard (**`System.in`**) using the **`Scanner`** class.

Lab Objectives

By the end of the lab, you should be able to:

- declare and initialize variables involving Java's primitive data types;
- declare named constants;
- assign values to variables involving arithmetic expressions;
- write statements to perform basic output (using **`System.out.print`** and **`System.out.println)`** and input (using **`Scanner`** and **`System.in`**)
- include basic comments in programs.

Prerequisites

The lab covers materials from the course textbook, and so it assumes that you have read the appropriate sections of the textbook dealing with the programming concepts and lab objectives found in this lab. You may use your course textbook as you work on this lab assignment.

What to Submit

In the lab, you will create a small Java programs. A source file (ending in **`.java`**) should be submitted to the course website. Further instructions are included at the end of this document.

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. Most statements should be defined in the **`main`** method of the class.

Part I

Please follow the guidelines below.

- At the top of the Java file, and for every file for every lab and project you submit this semester, agree to and include the comment below containing the class name, your name, submission date, and the program's purpose.

```
/*
* [Class name here].java
* Author:  [Your name here]
* Submission Date:  [Submission date here]
*
* Purpose: A brief paragraph description of the
* program. What does it do?
*
* 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 any source other than the course webpage
* or the course textbook. I recognize that any unauthorized
* assistance or plagiarism will be handled in accordance
* with the policies at Clemson University and the
* policies of this course. I recognize that my work is based
* on an assignment created by the School of Computing
* at Clemson University. Any publishing or posting
* of source code for this project is strictly prohibited
* unless you have written consent from the instructor.
*/
```

- Declare the following named constants of type **`double`** 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.
- Declare a variable of data type **`int`** called **`hoursPerWeek`**.
- Declare the following variables of type **`double`.**
  - **`grossPay`**
  - **`netPay`**
  - **`federalTax`**
  - **`stateTax`**
  - **`medicare`**
  - **`socialSecurity`**
- Assign the value 40 to **`hoursPerWeek`**.
- Compute **`grossPay`** by multiplying **`hoursPerWeek`** times the **`PAY_PER_HOUR`**.
- Compute **`federalTax`** by multiplying **`grossPay`** by the **`FEDERAL_TAX_PERCENT`** and divide that result by 100; assign the resulting value **`federalTax`**.

- Compute the **stateTax** by multiplying the **grossPay** by the **STATE_TAX_PERCENT** and divide that result by 100 and assign this value to the variable **stateTax**.
- Compute the social security contribution by multiplying the **grossPay** by the **SS_PERCENT** and divide that result by 100 and assign this value to the variable **socialSecurity**.
- Compute the **medicare** contribution by multiplying **grossPay** by **MEDICARE_PERCENT** and divide that result by 100 and assign this value to the variable **medicare**.
- Compute the **netPay** by subtracting the **federalTax**, **stateTax**, **medicare** and **socialSecurity** from **grossPay**.
- Display the **grossPay**, **federalTax**, **stateTax**, **socialSecurity**, **medicare** and **netPay** on the screen.

The output of the program should look similar 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
```

You should format the output text in columns using the **\t** escape sequence as discussed during class.

Compile and run your program. Afterwards, edit your program by changing the value assigned to the variable **hoursPerWeek**. Sample values are shown below.

```
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
```

Part II

Modify **`NetPay.java`** to ask the user for the number of hours worked.

- At the very top of the file before the declaration of your class, add the below statement.

  ```
  import java.util.Scanner;
  ```

  This tells the compiler that you will be using objects and methods from the Scanner class.
- In the **`main`** method, create a Scanner object called **`keyboard`** to read from **`System.in`**. Include the code for this immediately after the variable declarations.
- Add a print statement to prompt the user to enter the number of hours per week.
- Afterwards, add a read statement that reads in the hours per week and assigns this value to **`hoursPerWeek`**. The **`keyboard`** object should use the **`nextInt`** method of the Scanner class to read the value (since **`hoursPerWeek`** is an **`int`**).
- Check Listing 2.5 (pp. 95) of your textbook for more insights on requesting input from users.

Compile and run your program. A run of your program should look like this:

```
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.

Submission and Grading

After you have completed and thoroughly tested your program, upload and submit **`NetPay.java`** to our course website (if you have any questions about how to submit this assignment, then ask your lab instructor for help at least 24 hours before this assignment is due). Always double check that your submission was successful on our course website (if you have questions about how to do this, then ask your lab instructor for help).

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, if they do not compile, or if the submitting student has an unexcused absence for a single lab period devoted to the assignment, then a grade of 0 will be assigned. (Note: students who show up to their first lab period and ***personally*** show their lab TA that they finished and submitted the week's lab assignment may be excused from their second lab period for that week.)
- If the required comment for all labs describing the program and the academic honesty statement is not included at the top of the file, then 10 points will be deducted.
- If a (single) source file name is incorrect, then 10 points will be deducted.
- 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. Both the calculations and formatting of the output must be correct in order to receive credit for each test case.