CSE 114 Fall 2023: Assignment 3

Due: Sep 25, 2023 at 11:59 PM

Read This Right Away

This problem set is due at **11:59 pm on Sep 25, 2023 KST**. Don't go by the due date that you see on Brightspace unless you changed your default timezone to KST. By default, Brightspace uses US EST.

Directions

- At the top of every file you submit, include the following information in a comment
    - Your first and last name
    - Your Stony Brook email address
- Please read carefully and follow the directions exactly for each problem.
- Your files, Java classes, and Java methods must be named as stated in the directions (including capitalization). Work that does not meet the specifications will not be graded.
- Your source code is expected to compile and run without errors. Source code that does not compile will not be graded.
    - You can get partial credit if your program is incomplete but does compile and run.
- You should create your program using a text editor (e.g. emacs, vim).
- You must use the command-line interface to run your programs. That is, you must use the `javac` and `java` commands to do this assignment. Do not use IntelliJ IDEA yet.
- Your programs should be formatted in a way that is readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.

What Java Features to Use

For this assignment, you are *not* allowed to use more advanced features in Java than what we have studied in class so far.

What to Submit

Combine all your .java files from the problems below into a single zip file named **Homework1.zip** and upload this file to **Blackboard**.

Multiple submissions are allowed before the due date. Only the last submission will be graded.

Please do **not** submit `.class` files or any that I did not ask for.

Naming Conventions In Java And Programming Style In General

Please use these conventions as you establish your *programming style.* Programming professionals use these, too.

- **Names:** Choose informative names,
    - e.g. `hourlyRate` rather than `hr` for a variable name.

    - `CheckingAccount` rather than `CA` for a Java class name.
- **Class names:** We start a class name with an uppercase letter as I have in my examples: `Hello` not `hello` or `HELLO`. For multi-word names you should capitalize the first letter of each word,
    - e.g. `UndergraduateStudent`, `GraduateStudent`, `CheckingAccount`
- **Variable names:** We start a variable name with a lowercase letter,
    - e.g. `count`, `hourlyRate`, averageMonthlyCost
- **Method names:** Naming convention for method names is the same as that for variable names.
- **Use of white space:** Adopt a good indentation style using white spaces and be consistent throughout to make your program more readable. Most text editors (like emacs and vim) should do the indentation automatically for you. If this is not the case, see me and I can help you configure your setup.

I will not repeat these directions in each assignment, but you should follow this style throughout the semester.

Problem 1 (5 points)

Create a file named **WhatDay.java**

Write a Java program that will ask the user to type in a date and then determines the day of the week. There is a formula you can use to determine any day.

$$d = (q + \frac{13(m+1)}{5} + k + \frac{k}{4} + \frac{j}{4} + 5j) \% 7$$

(Note that all of the divisions are integer divisions).

- d is 0 through 6 representing Saturday (0), Sunday (1)… through Friday (6)
- q is the day of the month (1-31)
- m is the month (3: March, 4: April, …, 12: December, 13: January, 14: February)
- j is the century (the first two digits of the year)
- k is the year of the century (last two digits of the year)

Note that when the month entered is January or February, you must change the 1 for January to 13 or 2 for February to 14 and then subtract one from the year.

You can assume the user will only enter valid input. Assume the minimum year entered is 1600.

A sample run of the program is:

```
Enter the year (e.g. 1968): 2020
Enter month (1-12): 9
Enter the day of the month (1-31): 8
The day of the week is Tuesday
```

Submit **WhatDay.java**

Problem 2 (10 points)

Create a file named **Salary.java** (remember the class name will be the same as the filename).

A company has three kinds of employees:

- Corporate workers, who receive a set weekly salary
- Hourly employees, who receive a fixed hourly wage for the first 40 hours worked. For any additional hours, they receive 1.5 x their hourly wage for the overtime hours.
- Sales employees, who receive $1000 plus 12% of their weekly sales.

Write a program that calculates the weekly pay for an employee and prints on separate lines:
1) their gross weekly pay
2) the amount of taxes withheld, and
3) their net weekly pay (gross pay minus taxes)

The program prompts for what type of employee as shown below:

```
Select employee type: (c) corporate (h) hourly (s) sales:
```

Depending on the type of employee, the program asks for only the information required to determine that employee's gross weekly pay. For instance, for a sales employee the program needs to prompt only for the weekly sales (as a double) because the $1000 and 12% are fixed amounts and should be "hard-coded" in your program. Here's an example:

```
Select employee type: (c) corporate (h) hourly (s) sales: s ↲
Enter weekly sales: 4305.72 ↲
```

After determining the gross weekly pay, the program subtracts 12.5% for taxes and computes the net pay. Assume that any values entered by the user could contain a decimal amount. Use System.out.printf to format the three output values (gross pay, taxes, and net pay) with a dollar sign and exactly two digits after the decimal point for cents.

This is an excellent opportunity to learn how to structure code with methods. Try to move the calculations for each employee's gross pay into a method writing one method for each different type of employee. You can have the main program ask the employee type but then call the correct method to ask the remaining questions for that employee type and return a double value with their gross weekly salary. You will need to read input in each of the 3 routines so pass the Scanner object you create into each of the 3 methods and have them return the computed gross pay as a double value. If you want, write another method that, given the gross salary, returns the taxes to be withheld. Finally, have main print the output described above.

Below is an example run of the program:

Sample `printf` code: `System.out.printf("Total: %.2f", total);`

Note: Your program may assume that the user enters only valid input at the prompts.

Select employee type: (c) corporate (h) hourly (s) sales: c
Enter weekly salary: 4650.60
Gross pay: $4650.60
Taxes: $581.33

Select employee type: (c) corporate (h) hourly (s) sales: h
Enter hourly wage: 14.50
Enter hours worked: 52.5
Gross pay: $851.88
Taxes: $106.48
Net pay: $745.39

Select employee type: (c) corporate (h) hourly (s) sales: s
Enter weekly sales: 4305.72
Gross pay: $1516.69
Taxes: $189.59
Net pay: $1327.10

Submit Salary.java

Problem 3 (5 Points)

Create a file named Printjob.java (remember the class name will be the same as the filename). Write a program that asks the user to enter a string that encodes the characteristics of a print job. The string has this format:

"ColorType PaperSize Count"

ColorType , PaperSize, and Count are each separated by exactly one space.

The first part of the string represents the color of the printing. This cost is added to the cost of the paper itself:

Color Type (string)	Cost Per Sheet (₩)
"Grayscale"	5
"Color"	15

The second part of the string represents the size of paper used:

Paper Size (String)	Cost Per Sheet (₩)
"A4"	40
"A5"	20
"Letter"	30
"Legal"	25

Take in the user input in the main method, but then use a method named computePrintJobCost that takes three parameters: A String colorType, a String paperSize, and an integer count. The arguments should be in the order given. The method returns the cost to the main method. Call this method from main and print the total cost of the print job returned from computePrintJobCost in the main method.

You may assume that all user input will be valid (e.g. no invalid sizes, negative count, etc.). If the total end up with requiring a 5 won coin (e.g. 75), you should round up to the nearest unit of 10 won (e.g. 75 rounds to 80).

Several example runs are given below:

Enter colortype papersize and count: Color A4 4
Print job cost: 220

Enter colortype papersize and count: Grayscale Legal 53
Print job cost: 1590

Enter colortype papersize and count: Color A5 40
Print job cost: 1400

Hand in **Printjob.java.**

Problem 4 (5 Points)

Write a class Minimums. The class should contain 1 static function named min4. The method prototype is:

public static int min4(int num1, int num2, int num3, int num4);

It returns the lowest number of the 4 arguments.

You can do this with nested if statements. Better would be to create another function that compares two numbers with an if statement returning the smaller of the two. Then, call that function several times to get the least of the 4 values.

Here are a couple of sample runs:

Enter 4 integers on one line: -5 24 1111 -6
The smallest of these is: -6
Enter 4 integers on one line: 1 2 30 40
The smallest of these is: 1
Enter 4 integers on one line: -5 -20 -200 3
The smallest of these is: -200

Hand in **Minimums.java.**

Submission Instructions

Please follow this procedure for submission:

1. Place the deliverable source files (`WhatDay.java, Salary.java, Printjob.java, Minimums.java`) into a folder by themselves. The folder's name should be CSE114_HW3_<yourname>_<yourid>. So if your name is Joe Cool and your id is 12345678, the folder should be named 'CSE114_HW3_JoeCool_12345678'

2. Compress the folder and submit the zip file.
    a. On Windows, do this by clicking the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Blackboard for Assignment3.

   b. On Mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Blackboard for Assignment3.
3. Navigate to the course Blackboard site. Click **Assignments** in the left column menu. Click **Assignment3** in the content area. Under **ASSIGNMENT SUBMISSION**, click **Browse My Computer** next to **Attach Files.** Find the zip file and click it to upload it to the web site.