COMP 210 – Data Structures and Analysis (Sections 1 & 2)

Assignment #2 – Java Warm-up!

Issue Date: September 12th, 2023.

Due Date: Tuesday, September 19th, 11:55pm

Marks: 5

In this Assignment, you will be required to complete a Java program to process information about manufacturing data.

Don't stress if something doesn't work — come to the office hours and/or post on Piazza! If you feel entirely stuck and unable to resolve your issue, email Qiwei Zhao: qiwei@cs.unc.edu, and the LAs team will work together to help you!

Part A. Prerequisites and Setup

1. Java Development Environment:

Before proceeding, please ensure that you have already installed the required software such as Java, and IntelliJ.

2. Java Project Setup in IntelliJ:

- Launch IntelliJ IDEA.
- In your earlier created **COMP210** Project, right click on the src directory and select "New Package." Name your package "**assn02**".
- Locate this **assn02** directory, and save the "**JavaWarmUp.java**" starter code that came with this assignment inside the **assn02** directory.

At this point, you have the required setup for assn02. For the remainder of the assignment, you should modify the **JavaWarmUp** class to meet the specifications outlined in part B. This means that given an input of the form described in part B, it produces a report containing the information as shown in part B. You can define additional helper methods or helper classes that you think might be useful, but there is not one expected solution.

Part B. Write a program to process information about manufacturing facility data.

1. Requirements for your program.

Your program will create a database for a manufacturing facility that assembles various categories of electronic products such as phones, laptops, smart watches, etc. It will have a series of entries representing the items in this plant. You will initially be asked to enter some sample item entries and subsequently, the program should output some statistics of the entries.

The program should work as follows:

- The first input will be an integer that indicates how many items will be entered into the database.
- Following this will be information about each item on a single line in the following form:

MM/DD/YY HH:MM Category Fee Quantity Time Cost

Where:

MM/DD/YY represents the date of assembly.

HH:MM is the assembling time in hours and minutes in a 24-hour format.

Category represents a single-word descriptor of the category of the item. The following 3 types of categories must be supported and may be predefined in an array: “phone”, “laptop” and “smart_watch”.

Fee is the *per unit* assembling fee charged for this item (e.g. $12.18 per unit).

Quantity represents the number of items in this *batch*.

Time represents the *total time* required for producing this *batch* of items in the assembly line. The factory wage for producing items is **$16** an hour.

Cost represents the *total cost* of producing this *batch* of products in the assembly line.

<u>Here is an example of what your input might look like:</u>

```
6
6/8/22 19:32 laptop 41.73 593 384.5 1607
5/9/22 22:26 phone 20.79 3606 1795.0 2252
9/21/22 14:34 laptop 49.36 1525 1044.6 1779
8/6/22 11:26 phone 20.91 5401 1958.6 2381
9/15/22 1:38 smart_watch 12.99 1046 158.4 1756
9/6/21 5:09 smart_watch 12.18 670 100.5 1728
```

<u>Your program should produce a report that indicates:</u>

- **The highest and lowest per unit assembling fee** and the corresponding date, and time. In case of multiple productions with the same per unit assembling fee, it should report the production batch that comes later in the input (not necessarily later by date, but simply later as provided to the program at entry).

- **Some statistics by category**: specifically, your program needs to print the total number of products assembled for each category, the **average assembling** fee per unit, and the average net profit per unit.

Important notes: The average assembling fee is a weighted average that needs to consider the quantity of items per batch. The average net profit per unit is the average profit per product. This is based on the total fee charged minus the costs including the cost of time spent (time multiplied by worker's hourly rate).

Here is what your output should **exactly** look like (including decimal places) for the above input:

```
9/21/22
14:34
laptop
49.36
9/6/21
5:09
smart_watch
12.18
phone
9007
20.86
13.68
laptop
2118
47.22
34.83
smart_watch
1716
12.67
8.23
```

Important Notes on the Autograder:
For your output, each piece of data should occupy its own line. Do not add extra lines, breaks, spaces, or punctuation. Doing so might prevent the autograder from correctly recognizing your answers and could result in a grading error.

For the highest and lowest values, you should sequentially output their corresponding date, time, category, and price. The order of the categories should be determined by the sequence they appear in the data. For each category, make sure to output as per the above guidelines. And remember, always begin by naming the category first.

Please note that the autograder for this assignment is extremely picky and your output must exactly match the format in the examples.

2. Starter code, Hints and Notes:

To achieve the overall goal of the assignment, we have provided you the starter code that you should use. Within the starter code you have sections labeled as "TO DO" that provide you details of what you need to complete.

Feel free to use Java documentation on the internet to help you! When getting up to speed on a new programming language this is the normal thing to do. If you use a substantial amount of code from any documentation website, please cite it in a comment.

3. Submit to Gradescope for Grading

All that's left now is to hand-in your work on Gradescope for grading!

Before doing so, you need to know that before an assignment's deadline you can resubmit work as many times as you need to without penalty. Portions of assignments are autograded and will provide near-immediate feedback. We want you to resubmit as many times as it takes you in order to earn full autograding credit!

Login to Gradescope and select the assignment named "Assignment 2 - Java Warmup" You'll see an area to upload a zip file. To produce a zip file for autograding, return to IntelliJ.

Mac Users
Along the bottom of your window, you should see an option to open a terminal integrated into IntelliJ. Type the following command (all on a single line):

zip -r assn02_submission.zip assn02. (Your path should be set to the src file)

In the file explorer pane, look to find the zip file named "assn02_submission.zip". If you right click on this file "Open in -> Finder" on Mac, the zip file's location on your computer will open. Upload this file to Gradescope to submit your work for this assignment.

Windows Users
Navigate to your course workspace in a File Explorer window. Then right click on the src folder in your exercises directory and compress the directory into a zip folder. You can name it "assn02_submission.zip". Please pay attention, the level of the folder needs to be the same as the description. Compression software on Windows may introduce additional folder structure, and the file naming must be the same.

<u>When you upload it to Gradescope, please delete any files that showed up in the **src**/ folder that were not actually part of **assn02**. Specifically, in this assignment, the file that needs to be kept is the **JavaWarmUp.java**.</u>

Autograding will take a few moments to complete. For this exercise there will be no "Human graded" component, but in future exercises and projects there may. If there are issues reported, you are encouraged to try and resolve them and resubmit. If for any reason you aren't receiving full credit and aren't sure what to try next, come give us a visit in the office hours!