PROGRAMMING ASSIGNMENT 1

Issue Date : 08.03.2024 - Friday
Due Date : 24.03.2024 - Sunday (23:00)
Advisor: R.A. Aslı TAŞGETİREN
Programing Language : Java 8

1 Introduction

In this experiment you are going to gain experience in basic concepts of Object Oriented Programing, mainly Classes and Objects, by developing firmware of a simple machine.

2 Gym Meal Machine

Some people (e.g. bodybuilders, athletes, dancers etc.) have different dietary needs, to provide a convenient place for them to get their meals LiveLong Gym has decided to build a machine (i.e. Gym Meal Machine (GMM)) that will allow them to choose their meals based on the nutrients they want. With this machine members can freely choose their meals based on its protein, carbohydrate, fat, or calorie content, and they can also choose a specific item with its slot number.

In this experiment, you are expected to implement firmware of GMM using Object Oriented Programing. There are two parts to this experiment, first GMM should be loaded with meals/products, then it can take orders from members of the gym. These will be explained thoroughly in 2.1 and 2.2 respectively.

2.1 Loading the Gym Meal Machine

While loading the GMM, these rules should be followed:

- Loading should be done row by row, from left to right, using the **Product.txt** (see 3.1 for more information) file,
- GMM has 6 rows and 4 columns,
- If a slot already has a product then it should only be loaded with more of that product,
- Each slot has 10 product capacity. If a slot is full, the next empty slot or next slot that has the same product but is not full should be used,
- If all the slots are full, fill() function should write an informational message (e.g., "INFO: The machine is full!") and return -1.

While loading the GMM you are also expected to calculate product's calorie value, since it is not included in **Product.txt**. To calculate the calorie of a product you should use this formula[1]:

$$calorie(kcal) = 4(kcal/g) * protein(g) + 4(kcal/g) * carbohydrate(g) + 9(kcal/g) * fat(g)$$

2.2 Purchasing from the Gym Meal Machine

After GMM is filled, members can buy products. This is done according to these guidelines:

- Members can choose what they want by giving money and entering a nutritional value they want (e.g, protein, carbohydrate, fat, calorie), or they can directly choose a slot number. (Slots are numbered according to their filling order, i.e., 0.1.2.3/4.5.6.7/8.9.10.11 etc.),
- While choosing meals according to nutritional value, it is acceptable to return first product that is within -+5 range from requested value,
- GMM only accepts 1, 5, 10, 20, 50, 100 and 200 TL as money,
- Purchase details will be provided in **Purchase.txt** (see 3.2) file,
- After the chosen meal is given to members, any leftover change should also be returned,
- If there is any problem with a member's order, an information message should be written (e.g., "INFO: This slot is empty, your money will be returned." etc.), money should be returned, and functions should return -1.

3 Input Output Format

You will have two input files, which are Product.txt and Purchase.txt, and one output file which is GMMOutput.txt.

3.1 Product.txt

This input file includes a listing of all the products that will be loaded to GMM. It includes every product's name, price and its nutrition values such as protein, carbohydrate, and fat. It does not include calorie content of the product as you are expected to calculate that yourselves, see 2.1.

Format of Product.txt:

Name[tab]**Price**[tab]**Protein**[space]**Carbohydrate**[space]**Fat**

Example lines form Product.txt:

```
Chicken Wrap    18      32 45 28
Beef Sandwich   20      33 28 27
Fried Rice      15      11 90 8.2
```

3.2 Purchase.txt

This input file lists some purchases made from GMM. Each purchase has type of the payment method, all the money that were inserted into machine, the value type they are making their choice with (PROTEIN, CARB, FAT, CALORIE, NUMBER) and the value they want with their choice.

Format of Purchase.txt:

Type[tab]**Money_1**[space]...[space]**Money_n**[tab]**Choice**[tab]**Value**

Example lines form Purchase.txt:

```
CASH    5 10    PROTEIN 30
CASH    20 1 1 1 1      NUMBER  5
CASH    20 10   CARB    30
```

3.3 GMMOutput.txt

In output file you are to include:

- Contends of GMM machine after the loading is done. It should be in the same order as how the loading was done. (between ——Gym Meal Machine—— and ————-, see bellow),

  Format of GMM rows:

  product name[(]**product calorie**[,]**number of products in the slot**[)]___

  (If the slot is empty, put ___ instead of product name, and 0s instead of product calorie and number of products in the slot.)

  Example look of GMM:

```
-----Gym Meal Machine-----
Chicken Wrap(560, 7)___Protein Bar(211, 10)___Beef Sandwich(487, 5)___Cookie(153, 10)___
Fried Rice(478, 10)___Water(0, 10)___Cheese Sandwich(367, 10)___Energy Drink(148, 6)___
Protein Shake(163, 6)___Smoothie(175, 10)___Water(0, 9)___Fried Rice(478, 3)___
Cookie(153, 1)_____(0, 0)_____(0, 0)_____(0, 0)___
___(0, 0)_____(0, 0)_____(0, 0)_____(0, 0)___
___(0, 0)_____(0, 0)_____(0, 0)_____(0, 0)___
----------
```

- Purchase input (INPUT), the product that were chosen according to input (PURCHASE) and the leftover cash from the purchase (RETURN),

```
INPUT: CASH     20 10   CARB    30
PURCHASE: You have bought one Beef Sandwich
RETURN: Returning your change: 10 TL
```

- Any informational message about the issues that might have happened during purchases (INFO),

```
INPUT: CASH     5 10    PROTEIN 30
INFO: Insufficient money, try again with more money.
RETURN: Returning your change: 15 TL
```

- And lastly, the state of the GMM one more time.

Execution and Test

While testing the input files (Product.txt, Purchase.txt and GMMOutput.txt) should be given as arguments, *in that order*. You should do the following steps:

- Upload your java files to your server account (dev.cs.hacettepe.edu.tr)
- Compile your code (javac8 *.java, or javac8 Main.java)
- Run your program (java8 Main Product.txt Purchase.txt GMMOutput.txt)
- Control your output file GMMOutput.txt, it should have same data and format as the one given to you with your assignment.

Grading Policy

Task	Point
Clean code	15
Coding standard	5
Class design	10
Comments in JavaDoc Style	10
Correct output	60
Total	100

Submit Format

File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

```
- b<studentid>.zip
     - <src>
        - Main.java, *.java
```

Late Policy

You have two days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You have to submit your solution in deadline date + two days, otherwise it will not be evaluated.

Notes and Restrictions

- **Since the main objective of this experiment is to gain experience on Object Oriented Programing, any solutions that does not use Classes and Objects will not be accepted.**
- While designing your Classes you should pay attention to various characteristics of their attributes, such as which data type they should be, whether they should be changeable or if they should be only assigned once, whether there should be restrictions to its access

etc. And again while defining methods of your class, you should pay attention to which methods should be under which classes. These will be graded under "class design" task.

- Do not miss the submission deadline.
- You must obey given submit hierarchy and get score (1 point) from the submit system, if not as stated in BBM 104 Laboratory Rudiments you will lose 10% of your grade.
- Save all your work until the assignment is graded.
- Compile your code on DEV server before submitting your work to make sure it compiles without any problems on our server.
- Source code readability is a great of importance for us. Thus, write READABLE SOURCE CODE, comments and clear MAIN function. This expectation will be graded as “clean code”.
- Regardless of the length, use UNDERSTANDABLE names to your variables, classes and functions. The names of classes, attributes and methods should obey Java naming convention. This expectation will be graded as “coding standards”.
- You must use JavaDoc commenting style for this project. Use comments to explain design and algorithm choices you have made in your project. There is no need to state the obvious, (e.g, a variable named price is price of something), be brief, follow coding standards and write clean code.
- You can ask your questions through course’s piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.

References

[1] https://en.wikipedia.org/wiki/Atwater_system#Modified_system