Hacettepe University

Department of Computer Engineering

BBM104 Introduction to Programming Laboratory II

Programming Assignment I

Due Date : 11.03.2022 – 23:59

Advisor : Research Assist.

Calorie Calculation for Healthy Life

Introduction

In this experiment, you are expected to gain knowledge on basic JAVA programming. The program you are going to develop will deal with variables, loops, string operations, class system, file read and write operations. Besides the programming task, you will also learn to comply with coding standards.

1. Problem Definition

In this experiment, you are expected to write Java code that calculates the calories by considering taken and burned calories during the day for the healthy life of the people. You will be given three text files as follows:

1.1 Text for information of people (people.txt)

This text file includes personal information of each person, which are person ID **(personID)**, name **(name)**, gender **(gender)**, weight **(weight),** height **(height)** and date of birth **(dateOfBirth)** as shown in the following table. Every item in the file is separated with a **tab** character. This text file contains up to 100 items.

[person ID] *tab* **[name]** *tab* **[gender]** *tab* **[weight]** *tab* **[height]** *tab* **[date of birth]** *newline*
[person ID] *tab* **[name]** *tab* **[gender]** *tab* **[weight]** *tab* **[height]** *tab* **[date of birth]** *newline*

Example content of person.txt

12345	ahmet	male	78	175	1987
12346	ahmet	male	92	189	1990
12378	gizem	female	61	172	1986
……..					

1.2 Text for food (food.txt)

This text file includes information of foods, which are food ID **(foodID)**, name of food **(nameOfFood)** and calorie count **(calorieCount)** as shown in the following table. Every item in the file is separated with a **tab** character. For each food, 1 portion is 100 grams, and the calorie count in the table is calculated for 1 portion. ID of fruits groups start with 10.., ID of meal groups start with 11.., ID of dessert groups start with 12.., ID of nuts groups start with 13... and they consist of a 4-digit number. This text file contains up to 100 items.

[food ID] *tab* **[name of food]** *tab* **[calorie count]** *newline*
[food ID] *tab* **[name of food]** *tab* **[calorie count]** *newline*

Example content of food.txt

1001	apple	57
1101	spaghetti	131
1102	lahmacun	185
1201	baklava	521
…….		

1.3 Text for sport activities (sport.txt)

This text file includes information of the sport, which are sport ID **(sportID)**, name of sport **(nameOfSport)** and calorie burned **(calorieBurned)** as shown in the following table. Every item in the file is separated with a **tab** character. The calories burned for each sport are calculated for 60 minutes. ID of sport activities starts with 20.. and they consist of a 4-digit number. This text file contains up to 100 items.

[sport ID] *tab* **[name of** sport**]** *tab* **[calorie burned]** *newline*
[sport ID] *tab* **[name of sport]** *tab* **[calorie burned]** *newline*

Example content of sport.txt

2001	swimming	400
2002	running	300
2013	tennis	275
…..		

2. Calculation of daily calorie needs

People's daily calorie needs (dailyCalorieNeeds) vary by gender, age, height, and weight. Therefore, it will be calculated separately for men and women as follows:

$$\boldsymbol{Calculation\ for\ Men} = 66 + (13.75\ x\ weight\ (kg))\ +\ (5\ x\ height\ (cm)) - (6.8\ x\ age)$$

$$\boldsymbol{Calculation\ for\ Women} = 665 + (9.6\ x\ weight\ (kg))\ +\ (1.7\ x\ height\ (cm)) - (4.7\ x\ age)$$

The daily calorie needs **(dailyCalorieNeeds)** should always be rounded to the closest integer value. (Ex: 1234.4 →1234, 1234.8 →1235, 1234.5 →1235, 1234.49 →1234)

3. Text for input (command.txt)

Each line of the input file named as command.txt consists of either person ID **(personID)**, food ID **(foodID)** and the number of portions **(numberOfPortion),** or person ID **(personID)**, sport ID **(sportID)** and sport duration **(sportDuration)** as shown in the table below**.** During day, a person may add food ID that is eaten and sport ID that is done into this file. The **print(personID)** command should print the current calorie status of the specified person in command.txt file to monitoring.txt file. The **printWarn** command should print the person who take more than dailyCalorieNeeds end of the day considering calorie taken and burned. The **printList** command should print calorie statuses of all people given in command.txt file to monitoring.txt file. The expected output format is given in Section 4.

[person ID] *tab* **[food ID]** *tab* **[number of portions]** *newline*
[person ID] *tab* **[sport ID]** *tab* **[sport duration]** *newline*
…..
print (personID) *newline*
[person ID] *tab* **[sport ID]** *tab* **[sport duration]** *newline*
printList *newline*
printWarn *newline*
…..

Example content of command.txt

```
12345    1001    2
12378    1002    3
…..
print (12345)
12345    2001    45
printWarn
12378    1001    1
printList
……
```

4. Text for output (monitoring.txt)

You are expected to write output of your program to a text file named as monitoring.txt for persons specified in command.txt file. This text file should include the following information for each person in order as shown in the following table: name **(name)**, age **(age)**, daily calorie needs **(dailyCalorieNeeds),** calories taken **(caloriesTaken)**, calories burned **(caloriesBurned**) and result **(result)** for command of the print**(personID)**, **printWarn** and **printList**. For the **printWarn** command, if no person take more than **dailyCalorieNeeds** end of the day, print "*there is no such person*" to moninoring.txt. If the result is a number less than zero, it means that a person has taken less calories than they should take during a day. On the other hand, if the result is greater than zero, a person has taken more calories than they should take during a day. The result **(result)** should always be rounded to the closest integer value. Also, the output file should include **person ID (personID),** calories taken, name of the food **(nameOfFood)**, calories burned and name of sport **(nameOfSport)** to keep track calories burned and taken for a given person in the input file. Every item in the file separated with a **tab** character

[person ID] *tab* **has** *tab* **taken** *tab* **[calories taken]kcal** *tab* **from** *tab* **[name of food]** *newline*

*************** *(There will be 15 stars) newline*

[person ID] *tab* **has** *tab* **burned** *tab* **[calories burned]kcal** *tab* **thank to** *tab* **[name of sport]** *newline*

*************** *(There will be 15 stars) newline*

[name] *tab* **[age]** *tab* **[daily calorie needs]** *tab* **[calories taken]** *tab* **[calories burned]** *tab* **[result]** *newline*

*************** *(There will be 15 stars) newline*

[there] *tab* **[is]** *tab* **[no]** *tab* **[such]** *tab* **[person]** *newline*

*************** *(There will be 15 stars) newline*

……..

Example content of monitoring.txt

```
12345 has taken 200kcal from apple
**********
12356 has burned 100kcal thanks to tennis
**********
ahmet 27 1897kcal 2300kcal 404kcal -1kcal
**********
ahmet 27 1897kcal 2300kcal 404kcal -1kcal
gizem 25 1789kcal 1900kcal 430kcal -319kcal
**********
There is no such person
**********
……..
```

5. Example content of input and output file

In this experiment, you will be given an input file (command.txt) as below and you are expected to create an output file as shown below (monitoring.txt) by considering this given input file. The values in the example content of files given above (section 1.1, 1.2 and 1.3) are taken into consideration in this input and output files.

command.txt

```
11234 1001 3
11235 1002 2
11237 1009 4
11234 1104 2
11235 1113 2
print(11234)
11235 2003 60
11234 2004 60
printList
12239 1015 1
```

monitoring.txt

```
11234 has taken 261kcal from elma
**********
11235 has taken 196kcal from muz
**********
11237 has taken 236kcal from kavun
**********
11234 has taken 964kcal from kilistava
**********
11235 has taken 1250kcal from kuzucevirme
**********
murat 35 1776kcal 1225kcal 0kcal -551kcal
**********
11235 has burned 476kcal thanks to basketball
**********
11234 has burned 102kcal thanks to billiards
**********
murat 35 1776kcal 1225kcal 102kcal -653kcal
ayse 36 1374kcal 1446kcal 476kcal -404kcal
mehmet 37 1721kcal 236kcal 0kcal -1485kcal
**********
12239 has taken 64kcal from nektarin
```

Execution and Test

You will use the Java Platform as described in the. The input files (command.txt) should be given as an argument. Upload your java files to your server account (dev.cs.hacettepe.edu.tr)

- Upload your java files to your server account (dev.cs.hacettepe.edu.tr)
- Compile your code (javac *.java, or javac Main.java)
- Run your program (java Main command.txt)
- Control your output file (monitoring.txt) and format.

Grading Policy

Task	Point
Submitted	1
Clean code	10
Coding standards	10
Calculation of the calorie taken/burned part of the output	15
Print(personID) command (output)	15
PrintWarn command (output)	19
PrintList command (output)	30
Total	**100**

Submit Format

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

<student id>.zip

<src>
- Main.java, *.java
-people.txt
-sport.txt
-food.txt

Late Policy

You have three days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90, 80 and 70 for each late submission day). You have to submit your solution in deadline date + three days, otherwise it will not be evaluated.

Notes and Restrictions

- Do not miss the submission deadline.
- 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”.
- The names of input files are as given in this document (files will use in assignment (fixed files): *people.txt, food.txt, sport.txt*. Input file: *command.txt*. Output file: *monitoring.txt*)
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.