CMPE 160 – Introduction to Object Oriented Programming

Spring 2021 - Project#1 - Communication Simulation

Due Date: ***9 May 23.59***

Instructor: Tuna Tuğcu

TA : Yiğit Yıldırım, Ersin Başaran

Contact SA : Sinan Kerem Gündüz

keremgunduz077@gmail.com

Introduction:

We are all living in a world that technological revolutions occur almost every day. One of the most important revolution was ,of course, the Communication Systems. Today, you're going to implement a simple version of a communication system.

There will be Customers, Operators, and Bills that belong to the Customers. Customers can talk to each other, message each other, or they can connect to the internet via their mobile phones. Operators have their unique costs for the corresponding actions that will be given to you in the input. Every customer will have their unique Bill that stores and do the necessary actions about their spending.

You're going to take the input as an input file, do the operations, and will print out the necessary information to the output file. There will be no System.in arguments, in other words, there will not be any input given with the keyboard while the program runs.

Note that there will be multiple classes, (Not like in CMPE150) Therefore, think a little bit about the structure before starting to implementing program.

Field and Method names that given to you in this Document must be identical in terms of names and structures in your Project. That doesn't mean that you can not add extra Methods or Fields.

Classes and Implementation Details:

There will be 4 classes interacting with each other in this Project.

- Main
- Customer
- Operator
- Bill

Note that, it will be better to do the necessary calculations via using corresponding methods in the Customer, Operator or Bill class, not in the main Class. Some parts of the main class are given to you(input – output oerations) but, other 3 classes are given completely empty. You're expected to fill these classes with the given directions.

Main.java:

Main Class is for the general input - output operations. You are going to read an input file that contains directions about the simulation and do the several actions, then you will print out the desired results into the output file. These directions will be explained in a detailed way, later on this document. Name of the input and output files will be given as arguments of the program. For the input-output test cases, we are going to use your main class.

There will be 2 arrays in the main method as follows:

- Customer[] customers
- Operator[] operators

You are going to store the corresponding data in these arrays. All of these arrays must be initialized and, all of the elements in these arrays must be set to "null" as default. You're going to edit these arrays whenever a new object is created.

Customer.java:

Customer Class must have the following fields, exactly as named below:

- int ID
- String name
- int age
- Operator operator
- Bill bill

Customer class must have the following methods with the exact parameters:

- A constructor with five parameters: int ID, String name, int age , Operator operator, double limitingAmount
- void talk(int minute, Customer other) for customers to talk via the operator. Other is the another customer, mainly the second customer.
- void message(int quantity, Customer other) for customers to send message, amount is the number of messages to be sent. Other is the another customer, mainly the second customer.
- void connection(double amount) for customers to connect the internet. Amount is the number of data as MB.
- Getter and setter methods for age,operator,and bill. Ex: getAge (), setAge(int age)

Operator.java:

Operator class should have these fields with the exact names:

- int ID
- double talkingCharge
- double messageCost
- double networkCharge
- int discountRate

Operator class should have the following methods with the given parameters:

- A constructor with 5 parameters: ID, talkingCharge, messageCost, networkCharge, discountRate
- double calculateTalkingCost(int minute, Customer customer) for calculating the total amount to pay for talking.

- double calculateMessageCost(int quantity, Customer customer, Customer other) for calculating the total amount to pay for talking.
- double calculateNetworkCost(double amount) for calculating the total amount to pay for talking
- Getter and setter methods for talkingCharge, messageCost, networkCharge, discountRate

Bill.java:

Bill class must have the following fields:

- double limitingAmount
- double currentDebt

Bill Class must have the following methods with the exact parameters:

- Constructor with 1 parameter: limitingAmount. Note that you should initialize the currentDebt as zero.
- boolean check(double amount) is for checking whether the limitingAmount is exceeded or not.
- void add(double amount) is for adding debts to the bill.
- void pay(double amount) is for paying the bills with the given amount.
- void changeTheLimit(double amount) this method is for changing the limiting Amount.
- Getter methods for limitingAmount and currentDebt

Input Format:

You're going to read the input file line by line or token by token. First 3 lines will be integers. In the first line, the number "**C**" represents the number of customers will be in this Project. The second line has the number "**O**" which tells you that the number of Operators in the Project. In the third line, the number "**N**" will represent the number of events that are going to be simulated.

Next **N** lines are going to be operations with the given rules.

1-Creating a new Customer

2-Creating a new Operator

3-A customer can talk to another customer

4-A customer can send message to another customer

5-A customer can connect to the internet

6-A customer can pay his/her bills.

7-A customer can change his/her operator

8-A customer can change his/her Bill limit

Input 1 : Creating a new Customer

This line contains a String name, an int age, an int ID as operator ID, double as the limiting amount.

1 <name> <age> <ID> <limitingAmount>

Example : 1 Ahmet 20 4 100 (This creates a customer with the name Ahmet, age 20, and uses the operator with the ID 4, and has a bill with 100 limitingAmount)

Note that, id of the customer should be in the order of creation. To illustrate, first created customer must have ID 0 and the location in the array of that customer should be on his/her ID. Also there is no operation to create a bill object, therefore you need to create in the Customer class and store the bill object whenever you are going to create a customer.

Input 2 : Creating a new Operator

This Input line is followed by 3 doubles and 1 integer respectively, double talkingCharge, messageCost, networkCharge, int discountRate

2<talkingCharge><messageCost><networkCharge><discountRate>

Example : 2 1.2 0.48 0.15 15 (This creates an Operator object with 1.2/minute talking charge, 0.48/message messaging cost, 0.15/1MB network usage charge, %15 discount rate that will be applied to the proper customers)

Creation of an Operator object. Again, ID of the operator must be in the order of creation.

Input 3 : A customer talks to another customer

This line is followed by 3 integers that represents respectively, ID of the first customer, ID of the second customer , time as an integer.

3<1stCustomerID><2ndCustomerID><time>

Example: 3 4 5 17 (Customer with ID 4 and 5 are talking for 17 hours. Note that for talking operation, Customer ID 4 and 5 will both pay the talking charge according to their operators charge rates.

Input 4 : A customer sends message to another customer

This line is followed by 3 integers that represents respectively, ID of the first customer, ID of the second customer , number of messages that Customer 1 has sent.

4<1stCustomerID><2ndCustomerID><quantity>

Example: 4 7 2 6 (Customer with ID 7 sends 6 messages to customer with ID 2, for the messaging operations, only the sender customer is going to pay the messaging cost.)

Input 5 : A customer connects to the internet

This line is followed by an integer and a double, ID of the customer and amount of internet that the customer uses in MB.

5<CustomerID><amount>

Example: 5 0 23.4 (Customer with ID 0 uses 23.4 MBs of internet)

Input 6 : A customer pays his/her bills

This line is followed by an integer and a double, ID of the customer and amount of money that the customer wants to pay for his/her bill.

6<CustomerID><amount>

Example: 6 0 45.0 (Customer with ID 0 pays 45 liras to his/her bill)

Input 7 : A customer changes his/her operator

This line contains 2 integer as the CustomerID and the OperatorID

7<CustomerID><OperatorID>

Example : 7 3 2 (Customer with ID 3 changes his/her operator to the Operator with ID 2)

Input 8 : A customer changes his/her Bill limit

This line contains 1 integer and 1 double for, the CustomerID and the new limit

8<CustomerID><amount>

Instructions:

- Most of the calculations must be in the Customer, Operator, and Bill classes and in their methods.
- For the talking charge, charge must be the amount of minutes times the operator's talking charge per minute. If the Customer's age is below age 18 (18 is excluded) or higher than 65 (65 is excluded), than the operator applies a discount with the corresponding discount rate that given.
- For messaging, cost must be the quantity of the messages times the operator's message cost per message. If the 2 customers are using the same operator, then the operator applies a discount with the corresponding discount rate that operator has.
- For internet usage, calculations are straight forward, amount of MBs * the network cost per MB.
- Before the actions, you should check the limit of the bill, if someone would exceed the limit after the actions, then no actions must occur.
- We assume that every customer has enough money to pay their bill for the desired amount.
- ID of the Customers, Operators, and Bills should start from 0.

- You can declare extra fields or extra methods. But the given ones in this document must exist in your program and their parameters and names must be the same with the ones that given.

Output Format:

You should calculate the followings and print them out to the output file.

1-For each operator, you should print out, amount of time that they serviced for talking, number of messages sent via that operator, amount of internet usage in terms of MB that operator provided. Operator <ID of the Operator> : <talking time> <n Of messages> <MBs of usage>

Example :

Operator 0 : 16 4 29.2

Operator 1 : 10 7 29.4 (for a program that contains 2 Operators.)

2-For each Customer, How much Money that they spend for paying their bills and the current debt at the end of the simulation in their bills. Customer <ID of the Customer> : <total Money spent> < current debt>

3-ID of the Customer that talks the most and the amount of time in terms of minutes. < name of the Customer> : <talking time> (if 2 Customers are equal, then print out the one that has smaller ID)

4- - ID of the Customer that sends messages the most and the number of messages. <name of the Customer> : <number Of Messages> (if 2 Customers are equal, then print out the one that has smaller ID)

5 - ID of the Customer that connects the internet the most and the amount in terms of MBs . <name of the Customer> : <connection amount> (if 2 Customers are equal, then print out the one that has smaller ID)

- Double variables should be printed with 2 digits after the decimal points, if the third decimal point is 5 then you should round it to the up. (for

1.5019999 print out 1.50, for 8.879999 print out 8.88, for 1.245 print out 1.25, for 1.255 print out 1.26)

Example Input/Output:

3 3 12 2 0.5 0.48 0.15 18 2 0.43 0.47 0.18 20 1 aaa 20 1 100 2 0.55 0.20 0.35 10 1 bbb 16 0 200 1 ccc 48 1 150 4 2 0 7 5 0 14.4 4 0 1 3 3 1 2 13 6 0 2.5 7 1 2	Operator 0 : 13 0 0.00 Operator 1 : 13 10 14.4 Operator 2 : 0 0 0.00 Customer 0 : 2.50 1.50 Customer 1 : 0.00 5.33 Customer 2 : 0.00 2.63 bbb : 13 ccc : 7 aaa : 14.40

Some Extra Remarks:

- To emphasize again, methods and the fields given in this document should be implemented exactly as they are, in terms of names and parameters.
- You should adjust the visibility and accessibility of the fields. All of the variables shouldn't be declared as public. Also there will be grading just for using necessary accessing modifiers.
- We are going to grade your implementation using JUnit tests as well as the input/output grading.
- We highly encourage you to add comments to your project in order to increase readability in the case of an objection. Note that that is not an essential requirement, but it is an important ability to getting used to preparing documentation for an engineer.
- There will not be any erroneous test cases. Ex: there will not be a case that customer ID 4 tries to message while there were only 3 customers. But, you should also consider and check possible validity checks.

- You're going to read the inputs from a file, and print out the results into a file by using a PrintStream object. This part of the code is already given to you in the main.
- **Important :** Note that the deadline is strict for this project. You are going to submit your codes via the teaching codes system and system are going to close at **9 May 23.59**

Good luck.