EEM 480 Homework 3

In this homework you are required to write a simple database program of our COM480 E-trade company. Here in general what is necessary is to write a java program which will be used by COM480 computer department. The program basically keeps a customer information and his/her trade information. Our computer department engineers require a following type of structure.

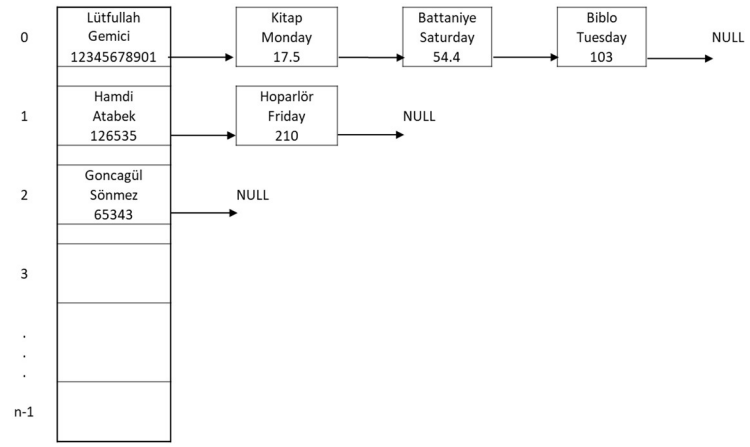

Figure 1 Example Database

In this requirements the customers will be kept in Customer class as :

```
public class Customer{
    private String Name;
    private String Surname;
    private int ID;
    private Item Link;

    @Override
    public String toString(){
        String retstr = "Name " + Name;
        retstr = retstr + " Surname " + Surname;
        retstr = retstr + " ID " + ID;
        return retstr;
    }
}
```

The items will be kept in Item class as :

```
public class Item {
    String ItemName;
    String Date;
    float Price;
    Item Link;
}
```

The Data Base Interface is given as :

```
public interface DB_Interface {
    public void addCustomer(Customer newCustomer);
    public void listItems(int ID);
    public Customer getNewCustomer(String Name, String Surname, int ID);
    public void addNewItem (Integer ID, String ItemName, String Date, float
Price);
    public Float getTotalTradeofCustomer(int ID);
    public Float getTotalTrade();
    public void readFromFile(String path);
    public Customer search_Customer(int ID);
}
```

When you are implemented the necessary methods :

`addCustomer` add the customer object to the customer array.

`getNewCustomer` gets an information of new customer from user. The information will contain Name, Surname and ID.

`addNewItem` adds the new item to the linked list of corresponding array location which contains the ID. If ID is not found, IDNotFoundException has to be thrown.

`getTotalTradeofCustomer` gets the ID of the user and finds the total amount of expenses of her/him and put the result on screen.

`getTotalTrade` finds and shows the total amount of trades of the company.

`listItems` list all the items that customer with ID to the screen.

`search_Customer` returns the customer object of ID

`readFromFile` read the trade text data from file. The file must be as given below as example:

E:\Mydrive\Mydata.txt

```
Lutfullah Gemici 13456
13456 Biblo Tuesday 103
Hamdi Atabek 126535
13456 Battaniye Saturday 54.4
Goncagul Sonmez 65543
126535 Hoparlor Friday 210
13456Kitap Monday 17.5
```

As you have noticed if a line starts with an integer it belongs to an item, otherwise it belongs to a customer. When the file given in example has been read the database given in figure 1 is created.

You are required to write the EEM480Database class which implements the DB_Interface. If the following code is run :

```
public static void main(String[] args) {
     // TODO code application logic here
     EEM480DataBase MyDataBase = new EEM480DataBase();
     Customer DummyCustomer = new Customer();
     MyDataBase.readFromFile("e:\\MyData.txt");
     Float exps = MyDataBase.getTotalTradeofCustomer(13456);
     System.out.println(MyDataBase.search_Customer(13456) + " Total Expense : " + exps);
     System.out.println(" The Total Trade : " + MyDataBase.getTotalTrade());
     MyDataBase.listItems(13456);
     Customer newc = new Customer();
     newc = MyDataBase.getNewCustomer("Ali", "Veli", 4950);
     MyDataBase.addCustomer(newc);
     MyDataBase.addNewItem(4950, "Karburator", "Monday", 145.8);
     MyDataBase.addNewItem(4950, "Laptop" , "Tuesday", 2340);
     System.out.println(" The Total Trade : " +
             MyDataBase.getTotalTrade());
     MyDataBase.listItems(4950);
}
```

The program will produce the following output.

The output :

```
The content of file has been read
Name Lutfullah Surname Gemici ID 13456 Total Expense : 174.9
The Total Trade : 384.9
Lutfullah Gemici 13456 Item List :
Kitap Monday 17.5
Battaniye Saturday 54.4
Biblo Tuesday 103
The Total Trade : 2870.7
Ali Veli 4950 Item List :
Laptop Tuesday 2340
Karburator Monday 145.8
```

Rules for HW Submission

. You must write your HW in NetBeans environment.
. You must write a report with name "**Report_HW3.pdf**" explaining your HW (purpose, how did you solve it, algorithm etc.) and what you the environment you used (NetBeans, for example). The person who read your report can easily use the class you have written.
. Discuss the result you have obtained.
. Submission should be in the form of a zip/rar. When extracted, the result should be a single folder with the name "HW2".
. Do not forget to put your report into the zip/rar file.
. The name of your project will be "**Name_Surame_HW3**. e.g. *Lutfullah_Arici_HW2*. If you do not obey the rule I will not grade your homework.
. You must bundle your whole project folder into your HW3.zip file.
. If I extract your project file, then import to my environment and if it doesn't work, you will be graded on 30 not 85. (Double check. It saves life)
. Do HW by yourself. Be honest.

Grading Criteria:
Implement a class that has fields for all the data contained in one data set (i.e. an integer, a floating-point number, a character, and a string), and member functions that perform the required computation and output. Data fields of your class should be declared as “private” members of the class, so you will need to implement get() and set() functions to access them. Member functions that perform computation and output should be “public” members of the class. Your program should then be modified so that it uses this class to store the data in a data set and to generate the required output.

A fully working program is worth 80 points. A useful and descriptive report file is worth 10 points. Useful comments throughout your program are worth 15 points. Clear, easy to read syntax (i.e. following coding style guidelines) is worth 10 points.

EEM 480
Coding Guidelines
What follows is a list detailing standards to be followed whenever you are programming for EEM 480 - Variable and function names are to be descriptive of what the variable represents, or what the function does (i.e. “int numDataSets” instead of “int n”). - The first word in the name of a variable or function is to be written in all lowercase letters. The first letter of each subsequent word is to be capitalized. Special characters (underscores and the like) should not be used except for in special cases (i.e. “int someReallyLongVariableName”). - Names of classes follow the same rule, with the exception that the first letter of the first word is also capitalized (i.e. “class MyClass”). - Proper spacing and indentation should be used (see examples):

```
//GOOD
for(int loopCount = 0; loopCount < 10; loopCount++)
{
       if (loopCount >= 4) //countdown from 5 to 0
       {
              System.out.print( 9 – loopCount );
       }
}
//BAD
for(int loopCount=0;loopCount<10;loopCount++)
{
if(loopCount>=4) //countdown from 5 to 0
{
System.out.print( 9 – loopCount );
}
}
```

- Class data members should be declared as “private” members of the class unless there is a compelling reason to do otherwise (and occasionally, there will be). - Class function members should generally be declared as “public” members of the class (though there will be more frequent exceptions to this rule than to the above rule).
- Global variable declarations should be avoided except in rare instances (for example, having a global variable that specifies whether “debug” mode is on or off).
- Generally speaking, a function that does computation should produce no output (except possibly for debug output), and a function that generates output should perform little to no relevant computation. Instead, have the function that performs computation return data (or a data structure) that can then be passed to the function that generates your output. The purpose of this is to keep your modules as specific and reusable as possible.
- Every function in a program should include comments detailing, at a minimum, what the function does, what the parameters are, and what the return value is.

- In instances where a descriptive enough variable name cannot be found while still using a reasonable amount of characters, additional comments are to be used to explain what the variable is used for.
- Particularly confusing pieces of code are to be accompanied by additional comments explaining what the code is doing (if it's confusing to you, as the person who wrote it, think how much worse it's going to be for a complete stranger trying to just look at it and understand what's going on).