Lab 3: Lists

Copyright ©2022 – Howard Community College All rights reserved; Unauthorized duplication prohibited.

This lab will continue with our TV store software that was started in Lab 2. As such, the first thing that must be done is to fix any issues that were found in Lab 2.

In this code, you are creating a customer database. The code will read in an initial file with existing customer data. The code will create a class that will contain an unordered list to hold this customer data. The code will allow the user to add, remove, update, display and save this list to a file. The code will also update the customer purchase code so that only customers in the list can purchase a TV.

There must be a javadoc at the beginning of the code with your name, the due date, a description of the program and a version number. The description is NOT the title of the lab. It needs to explain what is being done in the program. These are comments only.

Follow the sample output screen shots to determine how your program must run and then display the data. The first three lines of output must be the name of the Lab, the copyright statement, and the title of the program. This must be done via a call to a static method.

The TV class is unchanged.

The Customer class has only two changes:

1. The class must implement the Serializable interface in the Class header
2. A new constructor that takes the name and the account number as the only parameters

The Menu Interface has several changes:

1. The interface must contain all the final variables for the Customer Data submenu

The code must add a new class called CustomerData. CustomerData must implement an Iterable<Customer> and implement Serializable. The CustomerData class has only one instance variable:

1. A List of type Customer that holds all of the customer data

The CustomerData class must have:

1. A default constructor that instantiates a LinkedList of type Customer
2. An iterator method that returns the list as an iterable list.
3. A method to add a customer
4. A method to remove a customer
5. A method to update the name of a customer given an account number
6. A method to find a customer given an account number; this method must return a Customer object
7. A method to return the size of the customer list
8. A method to display the contents of the customer list in a formatted form; see the output for the format. The formatting must use printf and NO SPACES.

In the driver program:

1. Instantiate a new object of type CustomerData
2. Add a static method to
   1. Open and read the data file CustFile.txt
   2. Use the new two parameter Customer constructor to create a new Customer object
   3. Add the Customer object to the CustomerData list
   4. Return the full CustomerData object to main() once the file read has finished
   5. Display an appropriate error message if the file cannot be found
3. Add one menu option
   1. Customer Update
4. Customer Update must display a submenu with the following options:
   1. Add a customer
   2. Delete a customer
   3. Update the name of an existing customer
   4. Save the customer data to a file
   5. Display the customer list
   6. Return to the main menu
   7. Code must validate input to the menu and display appropriate error if an invalid entry is attempted via a try catch block; code must use final variables only, no hard coded numbers
   8. The code will continue to allow the user to update the customer list until the user specifically states to return to the main menu
5. Add a customer
   1. Display a title of what action is happening
   2. Ask the user for the customer name
   3. Ask the user for the customer account number
   4. If the account number already exists, display an error message, and require the user to reenter an account number; this must continue until a unique, new account number is entered
   5. Once there is a new account number, instantiate a new Customer object and add it to the CustomerData list
6. Remove a customer
   1. Display a title of what action is happening
   2. Ask the user for the customer account number
   3. If this account does not exist, display an error message, and require the user to reenter an account number; this must continue until an existing account number is entered
   4. Remove the list element with that account number from the CustomerData list
7. Update the name of an existing customer
   1. Display a title of what action is happening
   2. Ask the user for the customer account number
   3. If this account does not exist, display an error message, and require the user to reenter an account number; this must continue until an existing account number is entered
   4. Ask the user for the new name for this account number
   5. Update the name in the list for this account number; do NOT do this via removing and adding a list item
8. Save the customer data to a file
   1. Ask the user for the name of a file
   2. Create the file; display appropriate errors if necessary
   3. Write the entire customer data list to the file in the exact same format as the CustFile.txt that was provided with the assignment; display appropriate error is necessary
9. Display the customer list
   1. Display the entire customer list in a formatted manner; see the screen shot for the format
10. Return to main menu
    1. Must test that if something was changed and the user did not save to a file
    2. If nothing changed, return to main
    3. If something changed and the user saved, return to main
    4. If something changed and the user did not save, ask the user if they want to save before leaving
       1. If they answer y or Y, return to the Customer Update menu
       2. If they answer n or N, return to the Main Menu
       3. If they answer anything else, display an error and require that they enter in y, Y, n or N
11. Customer purchase should change to add the following:
    1. Display the customer list
    2. Ask the user to enter in the account number or ‘none’ (this can be in any form)
    3. If there is a match for the account number, use the name from the list and proceed as previously written (i.e. do not prompt the user for a name)
    4. If there is no match, ask for the name, add the new customer to the CustomerData list and proceed as previously written
    5. If none is entered, ask the user for a new account number (test for uniqueness) and a name; add the new account number and name to the customer list and proceed as previously written

Extra Credit:

If the user enters in a new Customer from the Customer Purchase option of the main menu, and the user never goes into the Customer Update sub menu and saves new customer list to a file, the program should check for this on exit from the program and ask the user if they really want to exit without saving the change to the file. If they do want to exist without saving, end the program. If they do not, redisplay the main menu.

Additional Requirements:

1. There should be NO hard coded values in the code. All final variables should reside in the interface file.
2. The code must check for the valid input of all numeric input entries. This includes not only checking for validity but checking that an integer is entered.
3. The code must use a stack for the TVs in inventory, a queue for the customers and an unordered list for the customer data. The code must use the Java stack, queue and list collections; do not use an array.

Submit your .java code files (the driver class, the TV class, the Customer class, the CustomerData class and the interface) via Canvas. Please use a .zip file for the submissions.

Screen Shots:

The new menu items**:**

**Graphical user interface, text, application, email

Description automatically generated**

Customer Sub Menu:

**Text

Description automatically generated**

Display the customer list (values immediately after reading):

**Table

Description automatically generated**

Adding a customer:

**Text

Description automatically generated**

Removing a customer:

**Text

Description automatically generated**

Changing the customer name:

**Text

Description automatically generated**

Exit without Saving:

Table

Description automatically generated

Save to a file:

Text

Description automatically generated

Return to Main Menu after Save:

Table

Description automatically generated

Existing Customer Purchase:

Table

Description automatically generated

New Customer Purchase:

Text

Description automatically generated with medium confidence