Final Project

To Be Uploaded by Sunday March 29, 2020

- the names of the members of your group
- a maximum of two people per group
- all group members will get the same mark for the project
- cannot add group members after March 29, 2020

To Be Handed In by First Class After Sunday April 5, 2020

- a printed copy of all code
- use NetBeans to print out the java code with line numbers
- only one member hands in the code, all member names must be displayed

Note: before you print out your code you must

- use Source > Format to style the code
- make sure that the code does not run beyond the red line
  - the red line indicates the right side of the page
  - code that goes beyond the red line will cause the print out of unnecessary paper
- marks will be deducted for unformatted and excessive paper

To Be Submitted by Sunday April 5

- a zipped copy of your NetBeans project to the dropbox by Wednesday
  - use File > Export Project > To ZIP
  - the name of this file should be yourName.zip
- only one member uploads the code
  - member names should be displayed

Projects submitted after Wednesday April 8, 2020 will not be marked.

To Be Demonstrated

- must demonstrate the program to me
- it will not be marked until you have demonstrated the program to me
- if you are unable to explain the code, your mark is ZERO
- if you do not show me a running version of the program, your mark is ZERO

Projects not demonstrated to me by Thursday April 9, 2019 will not be marked

Project Overview

- a JavaFX project that works with two sequential files, a Customer.dat file and an Order.dat file
- the user should be able to
  - review and make changes to existing orders
  - review and make changes to existing customers

    - add a new order or customer
    - review all orders of a given customer
    - review all orders of a given product

Project Structure

Should contain a separate folder that contains six files

- Customer.dat
- Customer.java
- CustomerFile.java
- Order.dat
- Order.java
- OrderFile.java

Customer.dat

- a sequential file that keeps track of Customer data
    - customerID, name, street, city
- should be in the format of:
    - C101, Fred Budd, 1 Main Street, Hamilton
- contains a minimum of five customers at start up

CustomerFile.java

- has a minimum of two methods
- one method returns an ArrayList of Customer objects based on the data from Customer.dat
    - a Customer object for each Customer record
- another method transfers the Customer data from the ArrayList to Customer.dat

Customer.java

- has private variables for each piece of data found in Customer.dat
    - customerID, name, street address and city
- has appropriate get and set methods for its private variables
- has only one constructor that sets the customerID

Order.dat

- a sequential file that keeps track of Order data
    - orderID, customerID, product, shipping method
- should be in the format of:
    - O10, C101, shoes, express
- contains a minimum of five orders at start up

OrderFile.java

- has a minimum of two methods

- one method returns an ArrayList of Order objects based on the data from Order.dat
    - an Order object for each Order record
- another method transfers the Order data from the ArrayList to Order.dat

Order.java

- has private variables for each piece of data found in Order.dat
    - orderID, customerID, product, shipping method
- has appropriate get and set methods for its private variables
- has only one constructor to set the orderID

Project Requirements

When the project starts up

- all the data found in Customer.dat is loaded into an ArrayList containing Customer objects
    - the data transfer should use methods of the CustomerFile class
- all the data found in the Order.dat is loaded into an ArrayList containing Order objects
    - the data transfer should use methods of the OrderFile class
- the first record of the Order.dat file should appear in appropriate text fields
    - one textfield for each data field
- first, next, previous, last buttons should appear to allow for basic navigation
- only CustomerID should be displayed in a TextField
    - loaded from the ArrayList containing Order objects
    - customer name, street and city should not be displayed

Failure to use textfields for navigation will result in a mark of zero for the project.

When the project is running

- the user should be able to navigate through the order records
    - is actually navigating through an order object ArrayList
    - click the next button, should see the next order
    - click the previous button, should see the previous order
    - click the last button, see the last order
    - click the first button, see the first order

      Note: only display customerID for the customer
        - name etc. is not to be shown
- the user should be able to modify an existing order
    - is actually modifying the order in the order ArrayList
    - the user sees the record to be changed
    - the user then clicks an update button
        - can only change the product or the shipping method
        - cannot change customerID or orderID

        - has an option to confirm the update or cancel the update
            - if confirmed, then the updated record should appear
            - if cancelled, then the original record should still appear
- the ability to delete an existing order
        - is actually deleting an order in the order ArrayList
    - the user sees the record and clicks a delete button
        - has an option to confirm the delete or cancel the delete
            - if confirmed, then the next record should appear
            - if cancelled, then the record should still appear
- the ability to add a new record
        - is actually adding an order to the order ArrayList
    - the user clicks an add button
    - empty text fields appear and the user types in the data
    - the user clicks an update button
        - has an option to confirm the add or cancel the add
            - if confirmed, then the new record should appear
            - if cancelled, then the previous record should appear

Note: when adding a new order

- must make sure that the orderID is unique (does not already exist)
- must verify whether the customer is a new customer or a returning customer
- if a new customer then a Customer Stage appears that allows the user to enter the new customer data
    - make sure that the customerID is unique
    - must add the customer to an ArrayList of customer objects
    - when the program ends, the ArrayList updates Customer.dat

Searches

- the user should be able to display
    - all the Orders on a separate Stage
    - all the Customers on a separate Stage
    - search all the orders of a given customer to be displayed on a separate Stage
    - search all the orders of a given product to be displayed on a separate page

Finally

- for all the above operations, proper exception handling should be used
    - thus if adding a record that requires an integer and user types in a String, exception message should appear
- when the user closes the project, any changes made by the user should be reflected in the sequential file
    - the data is transferred from the ArrayLists to the appropriate sequential file
    - the data transfer should use the appropriate methods of the OrderFile and CustomerFile classes