



Assignment 7 - File I/O (for Chapter 17)

[Submit Assignment](#)

Due Sunday by 11:59pm **Points** 50 **Submitting** a file upload **File Types** java

After studying Chapter 17 and its example of Product Manager application, plus 4 .java files discussed in Module 8, you should be able to write a Java program to handle data stored in text files.

This assignment requires you to create only one Java file, **Assignment7.java**, to read data from two text files, process these data to generate and print information, and also save updated data in a new text file. Different from past assignments, you will not receive detailed instructions and/or specifications about the program logic and structure but basically a simple overall file requirements listed below.

1. Your Assignment7.java must be saved in a specific folder you create for this work in disc, say, C:\2510\homework\Assignment7\, but not on your desktop.
2. The following two input text files contain data for the above program to read and process. They must be downloaded and saved in a folder named '**Ass7_Data**', which you create by hand but must be at the same level in the disc as where Assignment7.java is located. For example, if the program is saved in C:\2510\Assignment7\, these two files should be saved in C:\2510\homework\Ass7_Data\
 - **Orders.txt**  (2155 lines of data)
 - **Customers.txt**  (91 lines of data)
3. In no case you will modify the name and contents of above two data files.
4. Customers.txt contains 91 customerIDs, which are sorted and each is a 5-char string on its own line. Customers of this file are companies, not individuals, who make business with us.
5. Orders.txt contains 2155 lines of customer order data. In this file, each line represents one line item of certain customer order that contains 21 tab-separated data fields defined below, where EmployeeID represents the ID of one of our employees who handles the order, Freight means the shipping and handling fee, and Discount values like 0.05 or 0.2 means 5% or 20% discount. Instead of using Java date/time API of java.time package (introduced in Chapter 15), all date values in this file can be treated and processed as strings.
 - CustomerID
 - CompanyName
 - ContactName
 - ContactTitle
 - Address
 - City
 - Region
 - PostalCode
 - Country
 - Phone
 - Fax
 - OrderID

- OrderDate
 - RequiredDate
 - ShippedDate
 - EmployeeID
 - Freight
 - ProductID
 - UnitPrice
 - Quantity
 - Discount
6. Because normally an order consists of more than one line item, you will find several lines together in Orders.txt make one order. For instance, the very first three lines belong to order 10248 of customer VINET, and line 4 and 5 together define order 10249 of customer TOMSP. Therefore, you see customer and order data are repeated across lines that belong to the same order.
7. You will implement **Assignment7.java** program, which acquires two input file names, Orders.txt and Customers.txt, that are entered at command line (i.e., in args[] of main()) and reads both files completely as shown in the screenshot below.

```
C:\2510\Assignment7>javac Assignment7.java
C:\2510\Assignment7>java Assignment7 Orders.txt Customers.txt
-----begin of Assignment7-----
File ../Ass7_Data/Orders.txt is read successfully with 2155 lines
File ../Ass7_Data/Customers.txt is read successfully with 91 lines
(Q1) Total customers in orders file = 89
(Q2) Total orders in orders file = 830
(Q3) 2 customers with no orders = FISSA PARIS
(Q4) Largest order = $16387.5, which is order 10865 of customer QUICK
(Q5) Smallest order = $12.5, which is order 10782 of customer CACTU
File ../Ass7_Data/SIMC.txt is created (254 lines) but only 200 lines are updated
-----end of Assignment7-----
C:\2510\Assignment7>
```

8. In case of missing file name(s) or incorrect file name(s) are entered, the program must handle the errors or exceptions with a readable message and stop its execution. The next screenshot provides three sample tests and their messages for your reference. You may implement them in your own way using try-catch or if-

else, etc., but displaying the same error messages as shown in the screenshot is recommended.

```

Administrator: Command Prompt

C:\2510\Assignment7>java Assignment7
-----begin of Assignment7-----
.....Did you forget to enter file names at command line?
Please re-run this program.
-----end of Assignment7-----

C:\2510\Assignment7>java Assignment7 ABC.txt Customers.txt
-----begin of Assignment7-----
.....No file of ../Ass7_Data/ABC.txt can be found.
Please check if such a file exists and re-run this program.
.....Reading ABC.txt failed
-----end of Assignment7-----

C:\2510\Assignment7>java Assignment7 ABC.txt XYZ.txt
-----begin of Assignment7-----
.....No file of ../Ass7_Data/ABC.txt can be found.
Please check if such a file exists and re-run this program.
.....Reading ABC.txt failed
-----end of Assignment7-----

C:\2510\Assignment7>

```

9. The first console screenshot above is not just a sample but requirement, meaning your Assignment7.java must execute to process the data files and display information as shown in the screenshot, including output lines between "begin of Assignment7" and "end of Assignment7".
10. You will lose credits of this assignment if any **hard-coded data** except file name are written in your program code. Be sure you understand that, when your program is tested and graded, contents of either or both files can be modified, i.e., another pair of testing files could be used during grading and the output figures based on these files would be different from the above first screenshot. In fact, you are encouraged to modify the two input data files for your own testing.
11. Your program, Assignment7.java, must first display two lines to indicate the total lines read from the input files. See the first screenshot above for how they should be displayed.
12. Your program, Assignment7.java, must answer five queries defined below and display their answer in the form as shown in the first screenshot.


Q1: There are 2155 order lines in the Orders.txt, but these orders belong to only a small number of customers (or companies). How many customers (or companies) can be found from Orders.txt? Do not count the same customer twice. The answer should be 89 customers.

Q2: How many different orders can be found from Orders.txt? Do not count the same order twice. The answer should be 830 orders.

Q3: Customers.txt contains 91 customers in total. Compared to Orders.txt, where only 89 customers are found. That means, there should be two customers not appeared in Orders.txt because they never place any orders. Who are these customers? The answer should be FISSA and PARIS.

Q4: Every line item's total amount is calculated by 'unit price * quantity * (1 - discount)'. Accumulating the total of line items of the same order gives the grand total amount of that order. If we rank orders in terms of their grand total amount, then, which order is the largest order? What's its order ID and who is the customer?

Q5: Similar to Q4, except for the smallest order.

12. Before end, your program, Assignment7.java, will write data to a new text file, which is created at run-time, not by hand, and must be named '**SIMC.txt**' in the same location as the other two input text files. In case the file already exists, your program will replace or overwrite its contents in each execution. This file contains a **subset** of Orders.txt, which must be orders of customers in countries of [Spain](#), [Italy](#), [Mexico](#), and [Canada](#), i.e., SIMC, and, if an order line item's discount is originally zero, it is updated to be **3%** in SIMC.txt, and if the original discount is 20% or higher, it is decreased by **1%** and saved in SIMC.txt. Lines that satisfy these conditions will be copied, modified, and saved with only 8 fields, which are CustomerID, Country, OrderID, OrderDate, ProductID, UnitPrice, Quantity, and Discount. All data fields must be tab-separated.
13. As seen in the first screenshot, the previous writing file operation needs to be concluded with a message to indicate how many lines are copied from Orders.txt and how many of them whose discount value is updated. For your reference, a correct sample of the output file is provided here [SIMC.txt](#) . No need to upload your output file for grading.
14. Either you use additional classes and/or methods to complete this assignment is up to you. If additional classes are used, they are expected to be included in Assignment7.java with no package statements. It is suggested that you use 'readWriteFile.java' we discussed in Module 8 as a good reference to help complete this work.
15. When finished, upload only Assignment7.java for grading.