Seneca College

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

JAC444 **Submission date:** **Oct 31, 2021**

Workshop 5

Description:
The following workshop lets you practice basic java coding techniques, creating classes, methods, using arrays, inheritance, polymorphism, Exceptional Handling, Java I/O.

Task 1 – (a)

In commercial data processing, it's common to have several files in each application system. In an accounts receivable system, for example, there's generally a *master file* containing detailed information about each customer, such as the customer's name, address, telephone number, outstanding balance, credit limit, discount terms, contract arrangements and possibly a condensed history of recent purchases and cash payments.

As transactions occur (i.e., sales are made and payments arrive in the mail), information about them is entered into a file. At the end of each business period (a month for some companies, a week for others, and a day in some cases), the file of transactions (called "**trans.txt**") is applied to the master file (called "**oldmast.txt**") to update each account's purchase and payment record. During an update, the master file is rewritten as the file "**newmast.txt**", which is then used at the end of the next business period to begin the updating process again.

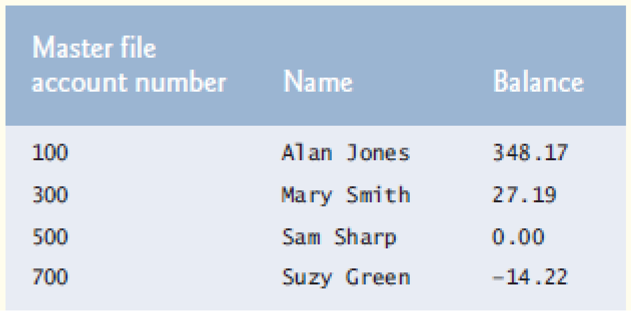
File-matching programs must deal with certain problems that do not arise in single-file programs. For example, a match does not always occur. If a customer on the master file has not made any purchases or cash payments in the current business period, no record for this customer will appear on the transaction file. Similarly, a customer who did make some purchases or cash payments could have just moved to this community, and if so, the company may not have had a chance to create a master record for this customer.

Write a complete file-matching accounts receivable program. Use the account number on each file as the record key for matching purposes. Assume that each file is a sequential text file with records stored in increasing account-number order.

- Define class **TransactionRecord**. Objects of this class contain an account number and amount for the transaction. Provide methods to modify and retrieve these values.
- Modify class *Account* from the Week 7 Lecture in class to include method *combine*, which takes a **TransactionRecord** object and combines the balance of the Account object and the amount value of the TransactionRecord object.

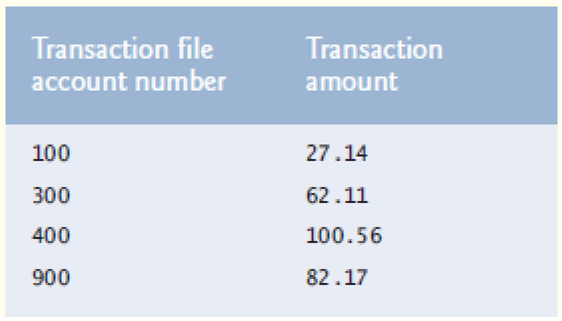
- Write a program to create data for testing the program. Use the sample account data in **Figs. A and b**. Run the program to create the files **trans.txt** and **oldmast.txt** to be used by your file-matching program.

- Create class FileMatch to perform the file-matching functionality. The class should contain methods that read oldmast.txt and trans.txt.
- When a match occurs (i.e., records with the same account number appear in both the master file and the transaction file), add the dollar amount in the transaction record to the current balance in the master record, and write the "newmast.txt" record. (Assume that purchases are indicated by positive amounts in the transaction file and payments by negative amounts.)
- When there's a master record for a particular account, but no corresponding transaction record, merely write the master record to "newmast.txt".
- When there's a transaction record, but no corresponding master record, print to a log file the message "Unmatched transaction record for account number…" (fill in the account number from the transaction record). The log file should be a text file named "**log.txt**".



Master file account number	Name	Balance
100	Alan Jones	348.17
300	Mary Smith	27.19
500	Sam Sharp	0.00
700	Suzy Green	-14.22

Sample data for master file (Fig a)



Transaction file account number	Transaction amount
100	27.14
300	62.11
400	100.56
900	82.17

Sample data for transaction file (Fig b)

Continue to next page…

Task 2 – (b)

It's possible (and actually common) to have several transaction records with the same record key. This situation occurs, for example, when a customer makes several purchases and cash payments during a business period.
Rewrite your accounts receivable file-matching program from **Task – 1 (a)** to provide for the possibility of handling several transaction records with the same record key.
Modify the test data of CreateData.java to include the additional transaction records in Fig. c.

Transaction file account number	Transaction amount
100	27.14
300	62.11
300	83.89
400	100.56
700	80.78
700	1.53
900	82.17

Additional transaction record (Fig c)

Workshop Header

/**
Workshop #
Course:*<subject type> - Semester*
Last Name:*<student last name>*
First Name:*<student first name>*
ID:*<student ID>*
Section:*<section name>*
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:*<submission date>*
**/

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Document all the classes properly
- Do Not have any debug/ useless code and/ or files in the assignment

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- You are supposed to create video/ record voice/ detailed document of your running solution.

  (50%)

  - Screen Video captured file should state your last name and id, like Ali_123456.mp4 (or whatever the extension of the file is)
  - Detailed document should include screen shots of your output, have your name and id on the top of the file and save the file with your last name and id, like Ali_123456.docx (or whatever the extension of the file is)
- A word/ text file which will reflect on learning of your concepts in this workshop.

  (30%)

  - Should state your Full name and Id on the top of the file and save the file with your last name and id, like Ali_123456.txt

- Submission of working code. **(20%)**
    - Make sure your follow the "**Code Submission Criteria"** mentioned above.
    - You should zip your whole working project to a file named after your Last Name followed by the first 3 digits of your student ID. For example, **Ali123.**zip.
- Your marks will be deducted according to what is missing from the above-mentioned submission details.
- Late submissions would result in additional 10% penalties for each day or part of it.

Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any