Assignment 1

Practical Tasks in Advanced Java Part 1 (15%)

DUE DATE

Session 11

OBJECTIVE

- Apply fundamental programming concepts to design a program.
- Develop the program logic.
- Employ JavaFX concepts learned in the sessions so far.
- Manipulate and access files in Java.
- Perform search and sort operations in Java.
- Use XML serialization.
- Perform multiple transactions on the content of a file.

DESCRIPTION

For each of the following problems, start by developing the program logic, (Flowchart or algorithm [pseudocode]), then develop each solution in Java code.

INSTRUCTIONS

1. File Matching

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.

a. Define class `TransactionRecord`. Objects of this class contain an account number and amount for the transaction. Provide methods to modify and retrieve these values.

b. Modify class `Account` in Fig. 15.9 in "15.5.1 Creating a Sequential File Using XML Serialization" in the *Java How to Program, Early Objects* (11th Ed.) e-book 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.

c. Write a program to create data for testing the program. Use the sample account data in Figs. 15.16 and 15.17 in "Exercises" in Chapter 15: "Files, Input/Output Streams, NIO and XML Serialization" in the e-book. Run the program to create the files `trans.txt` and `oldmast.txt` to be used by your file-matching program.

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

Fig.1 Sample Data for master file

d. 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`".

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

Fig.2 Sample Data for transaction file

2. File Matching with Multiple Transactions

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 the previous question 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 the below chart.

Account number	Dollar amount
300	83.89
700	80.78
700	1.53

Fig.3 Additional transaction records

3. File Matching with XML Serialization

Recreate your solutions for questions 1 and 2 using XML serialization. You may want to create applications to read the data stored in the .xml files — the code in "15.5.2 Reading and Deserializing Data from a Sequential File" in the e-book can be modified for this purpose.

SUBMISSION INSTRUCTIONS

Your submission must include a document with the logic for each problem in this assignment as well as each problem's source code.
Work must be submitted in the correct file type and be properly labelled as per the College naming convention:

NAME_COURSE_ASSIGNMENT. E.g. XuXiaLing_FM50D_A01.

GRADING CRITERIA

Assignment Value: **15%**

Grading Criteria	Grading
File Matching • Part a • Part b • Part c • Part d	/50
File Matching with multiple transactions	/20
File Matching with XML serialization	/30
TOTAL	**/100**