National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 212: Object Oriented Programming

Class: BSCS-6ABC

Lab 12: Files & Streams

Date: May 30, 2017

Instructor:

Dr. Anis ur Rahman / Dr. Mian M. Hamayun

Learning Objectives

The learning objective of this lab is to understand and practice the concept of text file handling in Java.

Lab Task #1 (File Matching)

In commercial data processing, it's common to have several files in each application system. In an accounts receivable system (to keep track of the amounts owed to a company by its credit clients), 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 **AccountRecord** (provided at the end) to include method **combine**, which takes a **TransactionRecord** object and combines the balance of the **AccountRecord** 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. 1 and 2. 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.

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.

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**".

```java
// Fig. 17.4: AccountRecord.java
// AccountRecord class maintains information for one account.
package com.deitel.ch17; // packaged for reuse

public class AccountRecord
{
   private int account;
   private String firstName;
   private String lastName;
   private double balance;

   // no-argument constructor calls other constructor with default values
   public AccountRecord()
   {
      this( 0, "", "", 0.0 ); // call four-argument constructor
   } // end no-argument AccountRecord constructor

   // initialize a record
   public AccountRecord( int acct, String first, String last, double bal )
   {
      setAccount( acct );
      setFirstName( first );
      setLastName( last );
      setBalance( bal );
   } // end four-argument AccountRecord constructor

   // set account number
   public void setAccount( int acct )
   {
      account = acct;
   } // end method setAccount

   // get account number
   public int getAccount()
   {
      return account;
   } // end method getAccount

   // set first name
   public void setFirstName( String first )
   {
      firstName = first;
   } // end method setFirstName

   // get first name
   public String getFirstName()
   {
      return firstName;
   } // end method getFirstName
```

```
50
51     // set last name
52     public void setLastName( String last )
53     {
54        lastName = last;
55     } // end method setLastName
56
57     // get last name
58     public String getLastName()
59     {
60        return lastName;
61     } // end method getLastName
62
63     // set balance
64     public void setBalance( double bal )
65     {
66        balance = bal;
67     } // end method setBalance
68
69     // get balance
70     public double getBalance()
71     {
72        return balance;
73     } // end method getBalance
74  } // end class AccountRecord
```

Hand in

Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single compressed/archived file named Lab_12_<Your CMS_ID. Your_NAME >.zip (without angle brackets) that contains ONLY the following files.

1) All completed java source files representing the work accomplished for this lab. The files should contain author in the comments at the top.
2) A plain text file named **README.TXT** that includes a) author information at the beginning, b) a brief explanation of the lab, and c) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.
2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
3. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the programming/other assignments.