In this lab, you will learn to handle Exceptions and to write data into a text file.

Problem statement: LoanRegister.csv has some basic data about loan accounts as shown in Figure 1. It has four data-items for each account: account number, principle amount, rate of interest, and number of years in the loan term. Your program needs to read this data, sort it in increasing order of EMI amount, calculate monthly EMI, and then print all data into another file named ProcessedLoans.csv. It also prints the Processed loans report. Important things to note:

1. LoanRegister.csv has seven records with bad data in 2nd record with O instead of 0 (Fig.1)
2. ProcessedLoans.csv has six records with 2nd record removed (Fig.2)
3. Program takes file name input. If the file does not exist, it simply prints a message (Fig.3)
4. If the file is found, it reads all data, reports which accounts have bad data, and then how many records were finally written into the ProcessedLoans.csv (Fig.4)

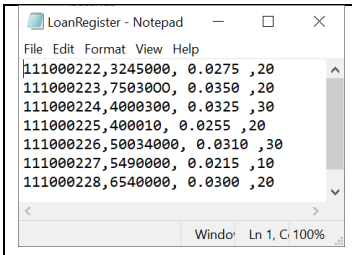

Figure 1: LoanRegister.csv

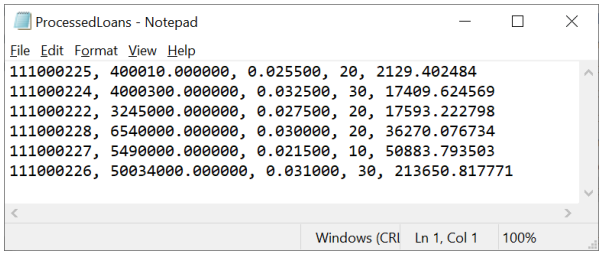

Figure 2: ProcessedLoans.csv

```
Enter file name
abc.csv
Invalid file name
```

Figure 3: Exception handling for invalid fie name

```
Enter file name
loanregister.csv
7 accounts read
Invalid data in account #111000223
6 accounts written

        ******** Processed Loans Report ********
  #. Account    Principle              Interest        Years    EMI
-------------------------------------------------------------------------
  1. 111000225  $     400,010.00        2.5500%         20      $  2,129.40
  2. 111000224  $   4,000,300.00        3.2500%         30      $ 17,409.62
  3. 111000222  $   3,245,000.00        2.7500%         20      $ 17,593.22
  4. 111000228  $   6,540,000.00        3.0000%         20      $ 36,270.08
  5. 111000227  $   5,490,000.00        2.1500%         10      $ 50,883.79
  6. 111000226  $  50,034,000.00        3.1000%         30      $213,650.82
```

Figure 4: Console I/O and Processed Loans Report

Solution Design: You are given LoanAccount.java, and LoanMaster.java. You need to make LoanAccount implement Comparable interface so that LoanAccounts can be sorted on their EMI values. LoanMaster.java has 6 methods, of which main() and printOutput() are fully coded. You need to code the remaining 4 methods:

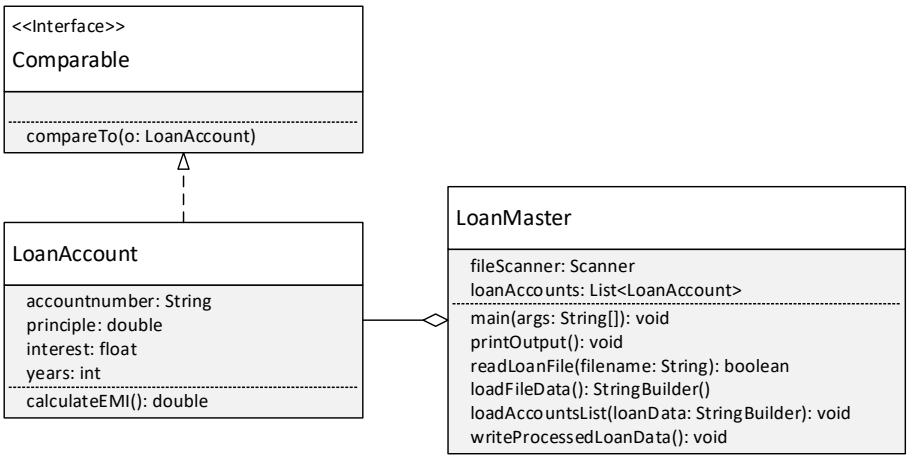

Figure 5: Class diagram

1. **readLoanFile()**: opens the file and attaches it to fileScanner. If successful, it returns true. If **FileNotFoundException** is thrown, it returns false.
2. **loadFileData()**: Uses fileScanner to read data line by line into a StringBuilder. Returns the Stringbuilder loaded with file data.
3. **loadAccountsList**(): Uses StringBuilder returned by loadFileData() to parse data line by line, split each line on commas, and create LoanAccount objects with the parsed values. Each LoanAccount object is added to loanAccounts array list. If any account data is 'dirty', then it handles the **NumberFormatException** so that a message is printed about which Account number's data is dirty (see Fig.4), and then moves on to process next line.
4. **writeProcessedLoanData()**: reads data from loanAccounts array list and writes it into ProcessedLoans.csv file. It also has EMI for each loan account, as shown in Fig.2

Instructions

1. Download LoanAccount.java, LoanMaster.java, TestLoanMaster.java, and LoanRegister.csv from Canvas
2. Create a package named lab9 in Labs project.
3. Import java files into this package and LoanRegister.csv into the 'Labs' project folder
4. Complete LoanAccount.java and LoanMaster.java.
5. Run LoanMaster.java to test console I/O
6. Run TestLoanMaster.java to test if all methods are coded as expected
7. If your program runs well, you should see a ProcessedLoans.csv in your Labs folder. You may have to refresh the view by right-clicking on Lab5 and selecting Refresh option. Open the file to see if it looks as expected.
8. Write your name and Andrew ID in the first line of LoanAccount.java and LoanMaster.java
9. Zip the two java files into **AndrewID-lab9.zip** and submit on Canvas.

Rubric:

- Test cases: 5 points (5 test cases for 1 point each)
- Console I/O: 5 points