COSC2081 - Programming 1

Programming 1 Assignment
Technical Report

s3805912 - Ha Gia Bao
s3801962 - Vu Dang Phuc

Overview **2**

Technical Details **2**
- Structure 2
- Technologies 2
- Functionalities Specific 2
  - Lead Managements 2
    - Display 2
    - Addition 2
    - Delete 3
    - Editing 3
  - Reporting and Statistics 3
    - Number of leads by age 4
    - Number of interactions by potentials 4
    - Number of interactions by month 4

Challenges and Progresses **4**

Work Division and Responsibilities **4**

List of Figures and Charts **4**

Overview

This report aims to describe in detail how the Customer Relationship Management (CRM) was built, the technologies used and how functionalities work and interact by themselves and with each other to perform required tasks.

Technical Details

Structure

The system contains a main Lead class where all information a Lead contains eg.id, name, etc. is stored. Similarly, a class for storing interactions information is also presented. To manage them, 2 new classes were created to store the methods necessary to perform the necessary maintenance. A class was also created to store the methods performing user input validations. For processing and displaying the statistics, a separate class was introduced to store all the relevant functions. Finally, is a class responsible for the console UI, called Menu.

Technologies

The technologies used in this program includes:

- LocalDate: to handle the Lead's and Interaction's date.
- Scanner: to handle user's input.
- BufferReader and Writer: to handle the reading and writing of files.

Functionalities Specific

Lead Managements

Display

To display all information of leads as requested by the user, the method simply reads leads.csv line by line, separates each line by the comma into tokens and prints those tokens accordingly.

Addition

To effectively add a new lead and interaction, 3 main methods were created, the main addLead method which reads leads.csv and add them to the csv file; getLeadInput, which responsibles for handling user's input and storing them into lead objects; and getNextLeadID, whose function is to handle the leadId and makes sure that it is unique.

- **addLeads:** adds the new input into the csv file in proper format. It does this mainly through the use of FileWriter and its print functionality.
- **getLeadInput:** uses Scanner to handle the user's input and add each input into a lead object accordingly. It also applies all necessary validations to what the user puts in.
- **getNextLeadID:** this method requires an extra text file where it stores the id of the last line

in the csv file, this file is called leadIdTracker. This method works through reading the leadID, separating numbers from the id and handles it specifically. It does this by first incrementing the current leadID by 1, then checking the number of zeros needed for the next ID, for example, if the next id is 10 then there will be only 1 zero, if the next id is 100 then there will be no zero needed, etc. Then it will print the new id into a temporary lead tracker file called tempLeadIdTracker. Finally, the temporary file will replace the original tracker file.

Delete

The delete method works through the ways of rewriting the file to a new file, omitting the line needed to be deleted then delete the original file then change the new file's name to the original file's name. The flowchart below shows how it works in specific.(Figure: 1)

First, the input of the method includes the leadID of the lead needed to be deleted. Then a String variable called currenLine is created to store the contents of each line when it is read. BufferReader is then used to read the content of lead.csv. BufferWriter is used to write the new contents to a new file called tempLead.csv. A while loop is created to go through each line of the lead.csv file until it reaches the last line. In the loop, any file that doesn't contain the input leadID will be written as it is to tempLead.csv. If the line contains the input leadID, it will be skipped and will not be written into the new file. When the loop is completed, both files are closed, lead.csv is deleted and tempLead.csv is renamed to leads.csv.

In addition, if there is any interaction associated with the lead being deleted, the program will ask the user if they want to delete those interactions too. If the user selected yes, then all associated interactions will be deleted, if not then they will be kept as they are.

Deleting interactions works in the exact same way, but now the input is interactionID, the files being read and replaced is interaction.csv.

Editing

The logic of editing is very similar to the one used in deleting leads. The original file is also rewritten into a new file, the original file is deleted and the new file is renamed, but this time instead of omitting the line, it is edited instead according to what the user required. Validations are also applied to the new inputs.

This method takes 3 inputs from the user, the ID, index number of what to change and the changes themselves. Similar to delete, BufferReader and Writer are used to read leads.csv and write tempLead.csv, respectively. When going through each line, split the lines by the comma into different tokens, the id will be at the first token or token[0], if it is equal to the input ID run a for loop through all of the tokens, then search for the input index, if the index match, change the content accordingly, then write it along with the rest of the line to the new file. After the line containing the ID input has been edited and written, write the rest of the lines until the last line is reached. Then both files are closed, leads.csv is replaced with tempLead.csv.(Figure: 2)

Reporting and Statistics

Number of leads by age

To report the number of leads by age, there needs to be a way to get the age from date of birth data. So another method was created inside the leads class to help with that, using Period's getYears. After that, to create the report, the method reads leads.csv, get the age from the dates then count accordingly. The flowchart below will give a more detailed description on how it works.(Figure: 3)

Number of interactions by potentials

This works similarly with reporting by age, but this time there is added a limiter dictated by the user, ie. start and end dates. LocalDate provides the ability to check if the date is after, before or is the date through the methods isAfter, isBefore and isEqual. The flowchart belows will give the specific details of where this was utilized in the method and also how this method works in detail.(Figure: 4)

Number of interactions by month

This report also works very similarly to the previous 2 reports, but this time added a new problem of counting by month and omitting the months which counts are at 0. To solve this, the months are put into an array of String and the counts for each are stored in a separated array called interCount. After checking the date and line, a for loop will run though the month array, if the month's index + 1 is equal to the month in the interaction's line, then the corresponding index in the interCount will be incremented, the loop will be broken as soon as that happens. The below flow chart will give a more specific description of how the method works.(Figure: 5)

Challenges and Progresses

While working on the project, we came against several problems that we managed to overcome. First was our difficulties in understanding the full requirements of the application, what is specifically required of each function, for example: the date format, the specific rules in validations, etc. This problem was quite easily solved by taking notes on what was confusing and asking our lecturer about them.
Another problem we faced was the lack of code organization. The code is later organized, but it took us quite a bit of time to do so, but it would have been a lot of time saved if we had a plan in the beginning.
After resolving all the problems, our progress was significantly improved. We did not have much problems meeting the agreed deadlines, so we managed to finish everything pretty early on, after which it was mostly just polishing and bug fixing.

Work Division and Responsibilities

Bao: menu design, display, add functions for leads and interaction, all validations
Phuc: delete, edit functions for leads and interactions, all report and statistics functionalities

List of Figures and Charts

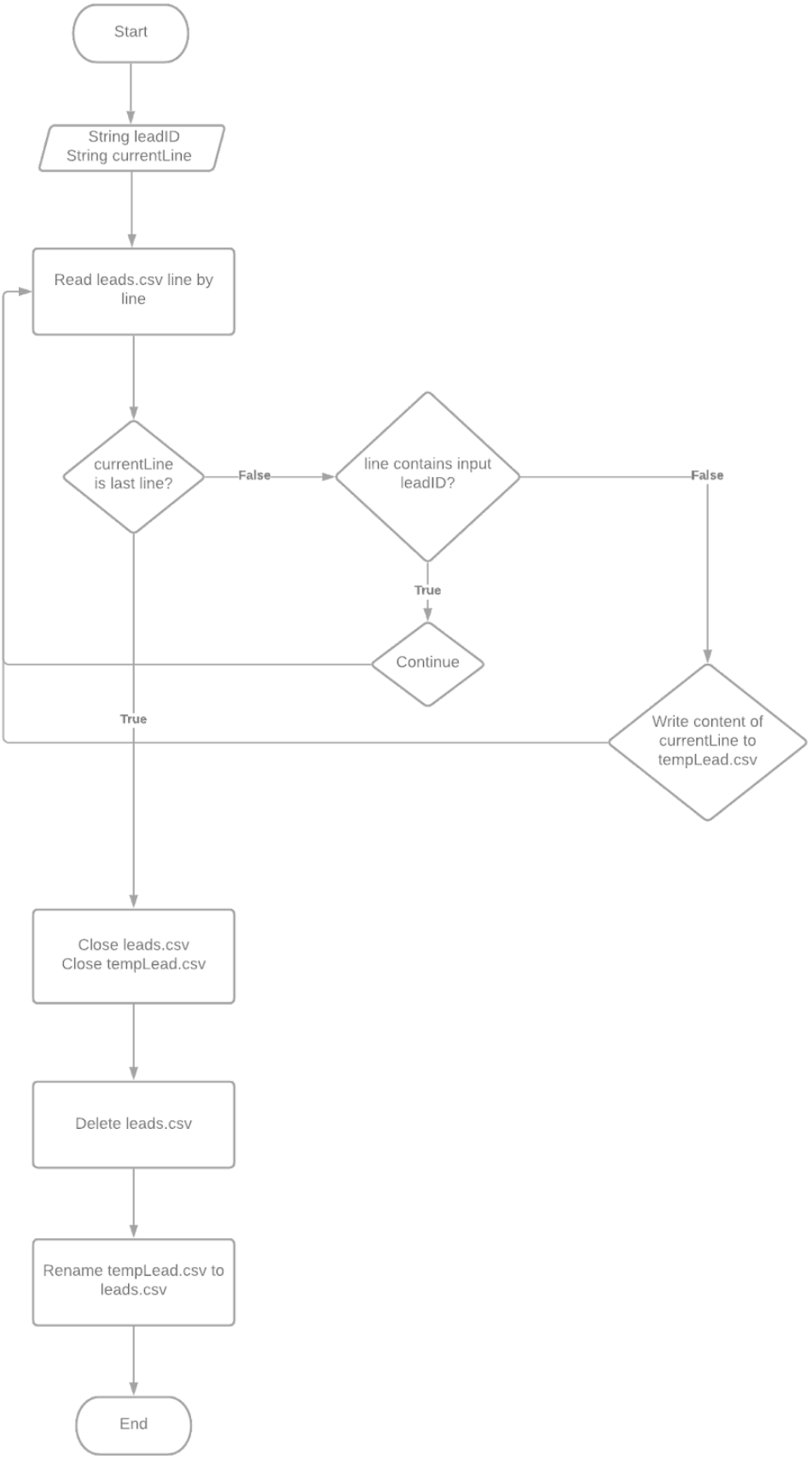
Start
String leadID
String currentLine
Read leads.csv line by line
currentLine is last line?
False
line contains input leadID?
False
True
Continue
True
Write content of currentLine to tempLead.csv
Close leads.csv
Close tempLead.csv
Delete leads.csv
Rename tempLead.csv to leads.csv
End

Figure: 1

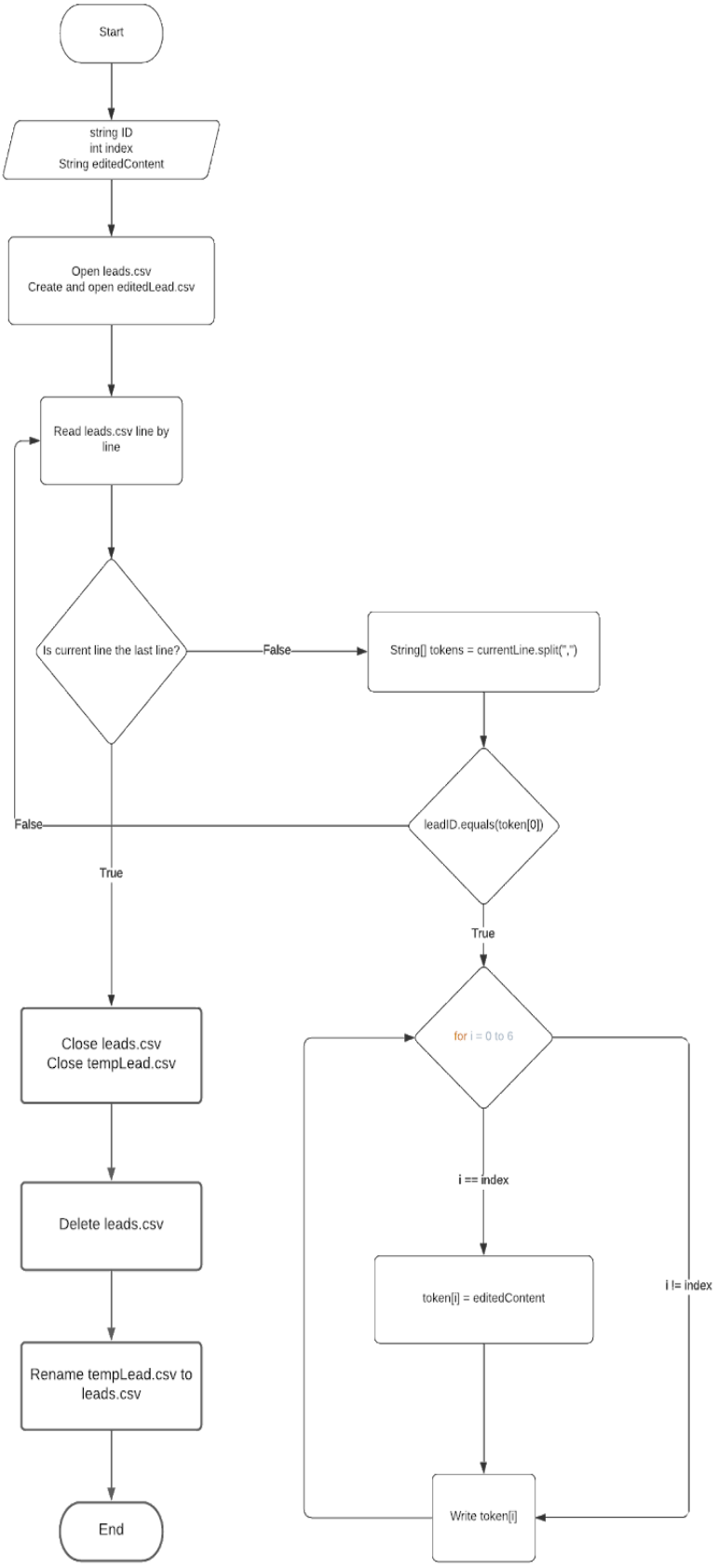
Start
string ID
int index
String editedContent
Open leads.csv
Create and open editedLead.csv
Read leads.csv line by line
Is current line the last line?
False
String[] tokens = currentLine.split(",")
leadID.equals(token[0])
False
True
True
Close leads.csv
Close tempLead.csv
Delete leads.csv
Rename tempLead.csv to leads.csv
End
for i = 0 to 6
i == index
token[i] = editedContent
i != index
Write token[i]

Figure: 2

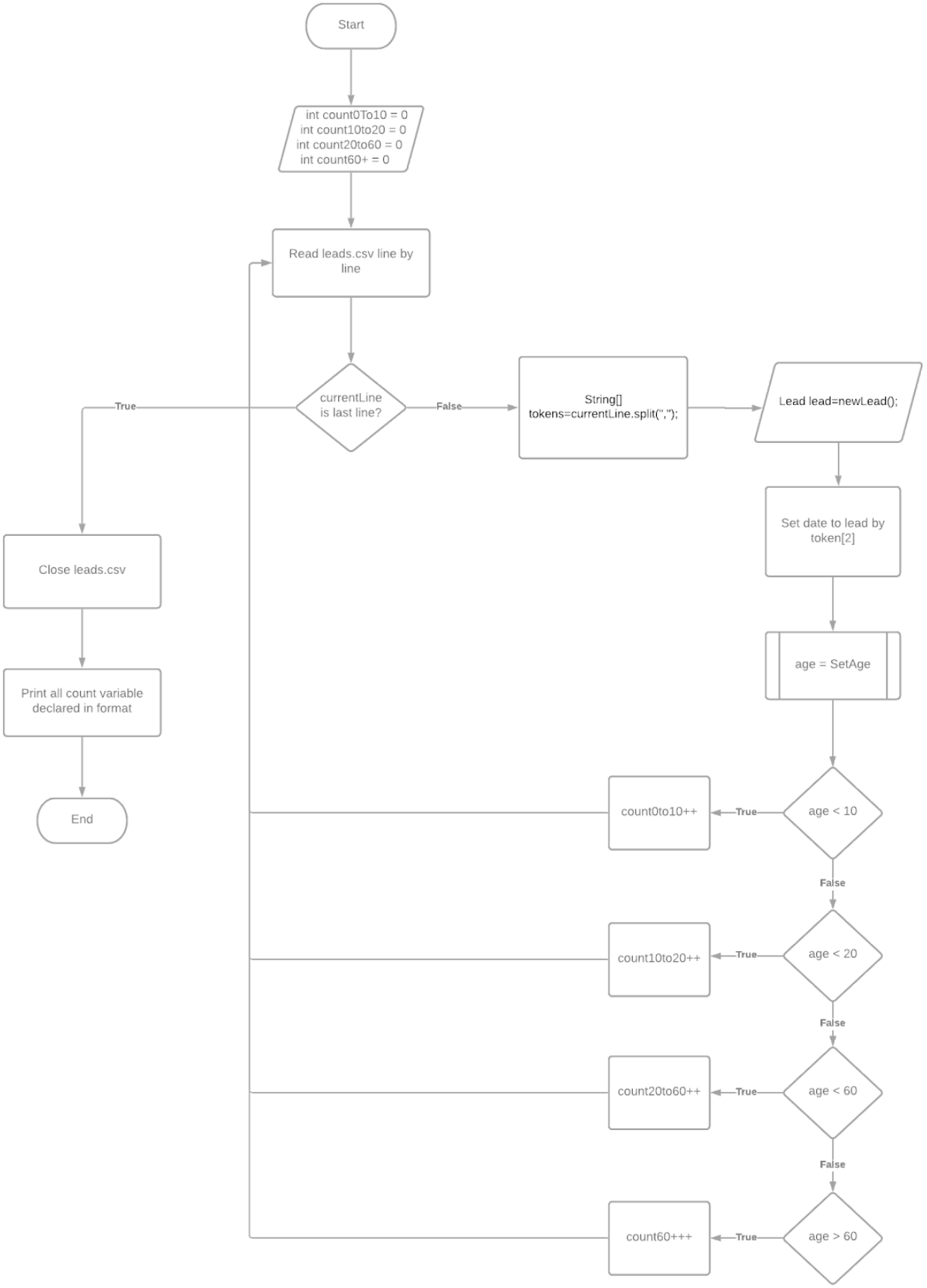
Start
int count0To10 = 0
int count10to20 = 0
int count20to60 = 0
int count60+ = 0
Read leads.csv line by line
currentLine is last line?
True
False
String[] tokens=currentLine.split(",");
Lead lead=newLead();
Set date to lead by token[2]
age = SetAge
Close leads.csv
Print all count variable declared in format
End
age < 10
True
count0to10++
False
age < 20
True
count10to20++
False
age < 60
True
count20to60++
False
age > 60
True
count60+++

Figure: 3

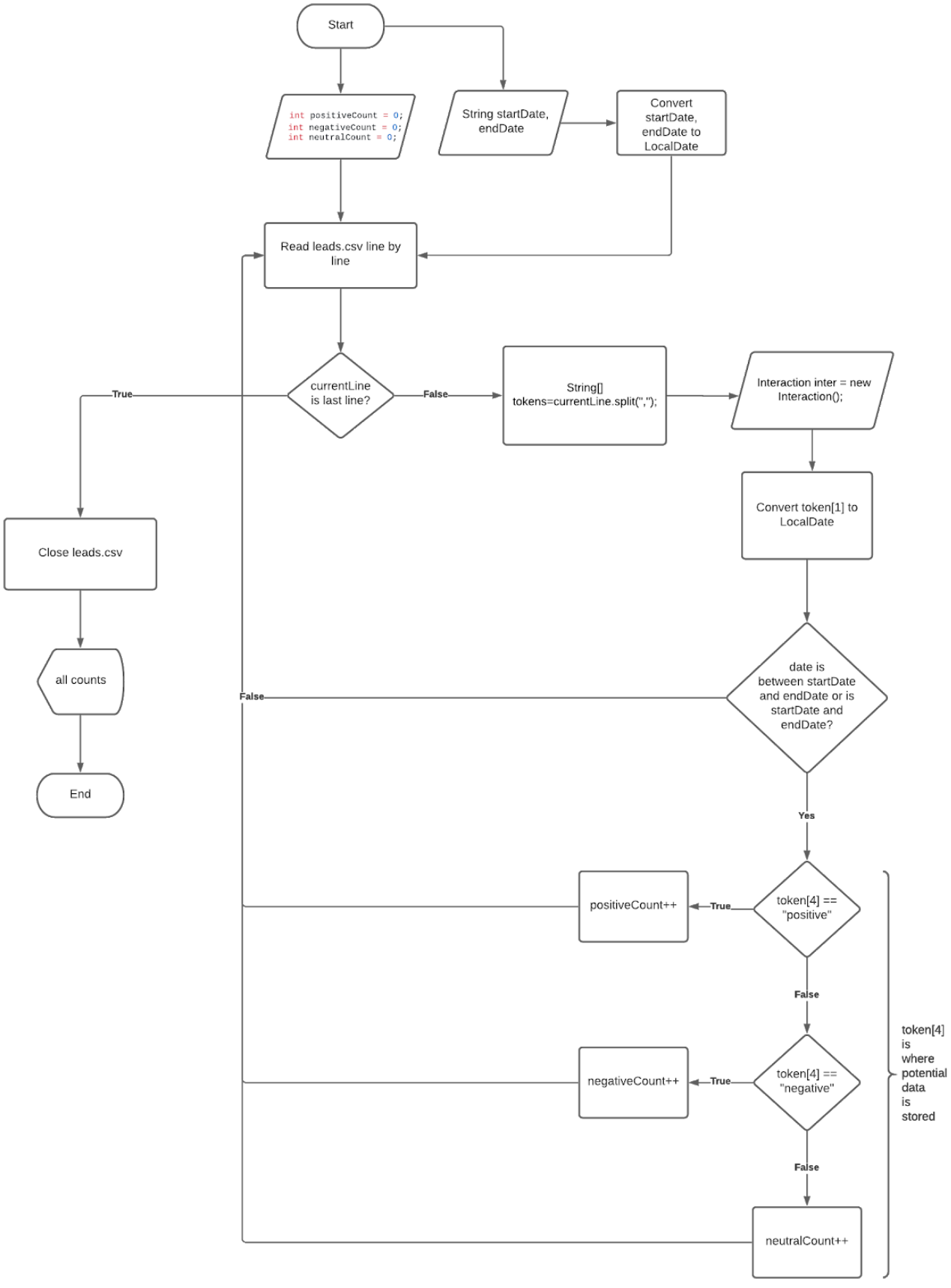
Start
int positiveCount = 0;
int negativeCount = 0;
int neutralCount = 0;
String startDate, endDate
Convert startDate, endDate to LocalDate
Read leads.csv line by line
currentLine is last line?
True
False
String[] tokens=currentLine.split(",");
Interaction inter = new Interaction();
Convert token[1] to LocalDate
Close leads.csv
all counts
End
date is between startDate and endDate or is startDate and endDate?
False
Yes
positiveCount++
True
token[4] == "positive"
False
negativeCount++
True
token[4] == "negative"
False
neutralCount++
token[4] is where potential data is stored

Figure: 4

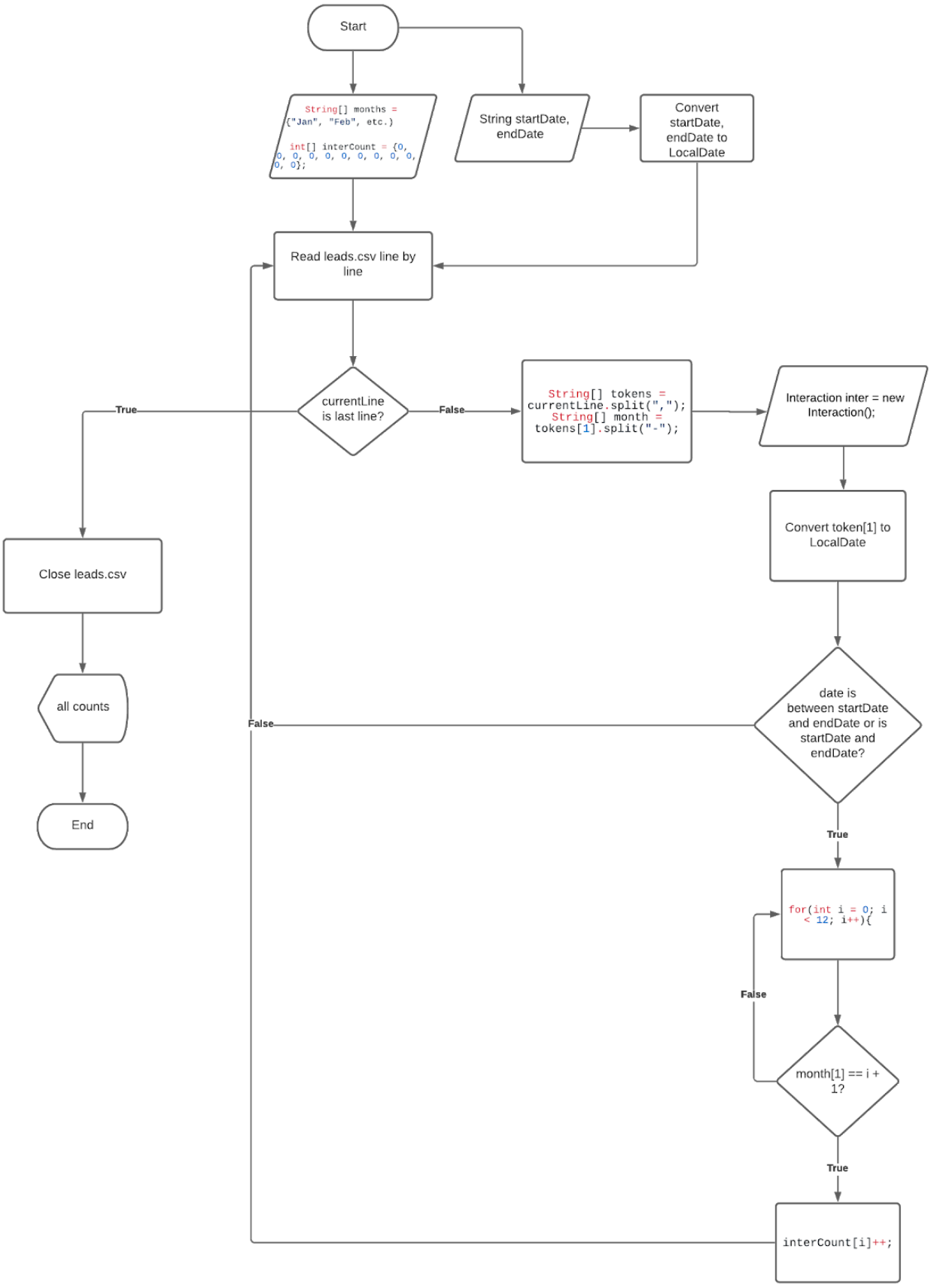
Start
String[] months = {"Jan", "Feb", etc.)
int[] interCount = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
String startDate, endDate
Convert startDate, endDate to LocalDate
Read leads.csv line by line
currentLine is last line?
True
False
String[] tokens = currentLine.split(",");
String[] month = tokens[1].split("-");
Interaction inter = new Interaction();
Convert token[1] to LocalDate
Close leads.csv
all counts
End
date is between startDate and endDate or is startDate and endDate?
False
True
for(int i = 0; i < 12; i++){
False
month[1] == i + 1?
True
interCount[i]++;

Figure: 5