**Purpose:** The purpose of this assignment is to allow you to practice Exception Handling, and File I/O, as well as other previous object-oriented concepts.

---

A group of social scientists has conducted studies on the community and data has been recorded using excel sheets. The scientists want to write a book about the findings of the study. The book is written using LaTeX. LaTeX is a typesetting system that is widely used among scientific community to produce technical report, scientific research papers and books.

The data gathered throughout the study will be published in the book in the form of LaTeX tables. As a software designer, you are hired to design and implement a Java code to convert the data into LaTeX table format. To learn about LaTeX table and view different formats visit: https://www.learnlatex.org/en/lesson-08.

Excel sheet is saved in comma-separated values (CSV) format which is commonly used for tabular data. Each table row is a line, with columns separated by commas. Items may be enclosed in quotation marks, and they must be if they contain commas or quotation marks. Here is a line with four fields:

179, Montreal, “Hello, world”

In this assignment, you are required to design and implement a Java tool called CSV2LATEX to help the scientists read and process CSV files and create the corresponding LaTeX tables.

The scientist collected the data on many excel files and each file could have different attributes and different number of entries. Figure 1 shows one example of the files in both Excel and CSV formats.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Employer Interview for Future Hiring** | | | | | |
| 2 | **Interview Date** | **CEO Name** | **Num of Employees** | **Industry** | **City/Province** | **Hiring** |
| 3 | 11/5/2020 | Adam | 326 | Software Development | Laval/ Que | Yes |
| 4 | 14/05/2020 | Steve | 1512 | Retail Store | Ottawa/ON | NO |
| 5 | 3/4/2020 | Sami | 720 | Car dealer | Toronto/ON | NO |
| 6 | 3/2/2020 | Jim | 142 | Traver Agency | Montreal/Que | Yes |
| 7 | | | | | | |

(a) Excel Format

*EmployerInterview - Notepad

File Edit Format View Help

```
Employer Interview for Future Hiring,,,,,
Interview Date,CEO Name,Num of Employees,Industry,City/Province,Hiring
11/5/2020,Adam,326,Software Development,Laval/ Que,Yes
14/05/2020,Steve,1512,Retail Store,Ottawa/ON,NO
3/4/2020,Sami,720,Car dealer,Toronto/ON,NO
3/2/2020,Jim,142,Traver Agency,Montreal/Que,Yes
```

(b) CSV format

Figure 1: Sample file (a) in Excel format (b) in Comma Separated Values (CSV) format

Input and output of your code are as follows:

**Input**: Input fiels in CSV format. Files typically include attributes as described in Figure 1(a), however they could be modified by the user and your code must be able to accommodate any CSV file with any attributes. You may assume that first line of CSV file contains title of the table, second line contains the attributes, and the rest of the lines contain the data. Title in CSV files is stored in LaTeX \caption{}command, file attributes are LaTeX table headings, CSV data are the LaTeX table entries, and name of the CSV file is stored into the LaTeX \label{} command, see the coorespondence between Figure 2 (a) and (b) .

**Output**: Your code must generate corresponding LATEX file for a CSV file. LATEX files must have the same name as corresponding CSV files but with extension ".tex". Figure 2 depicts a sample CSV file along with the corresponding LATEX file.

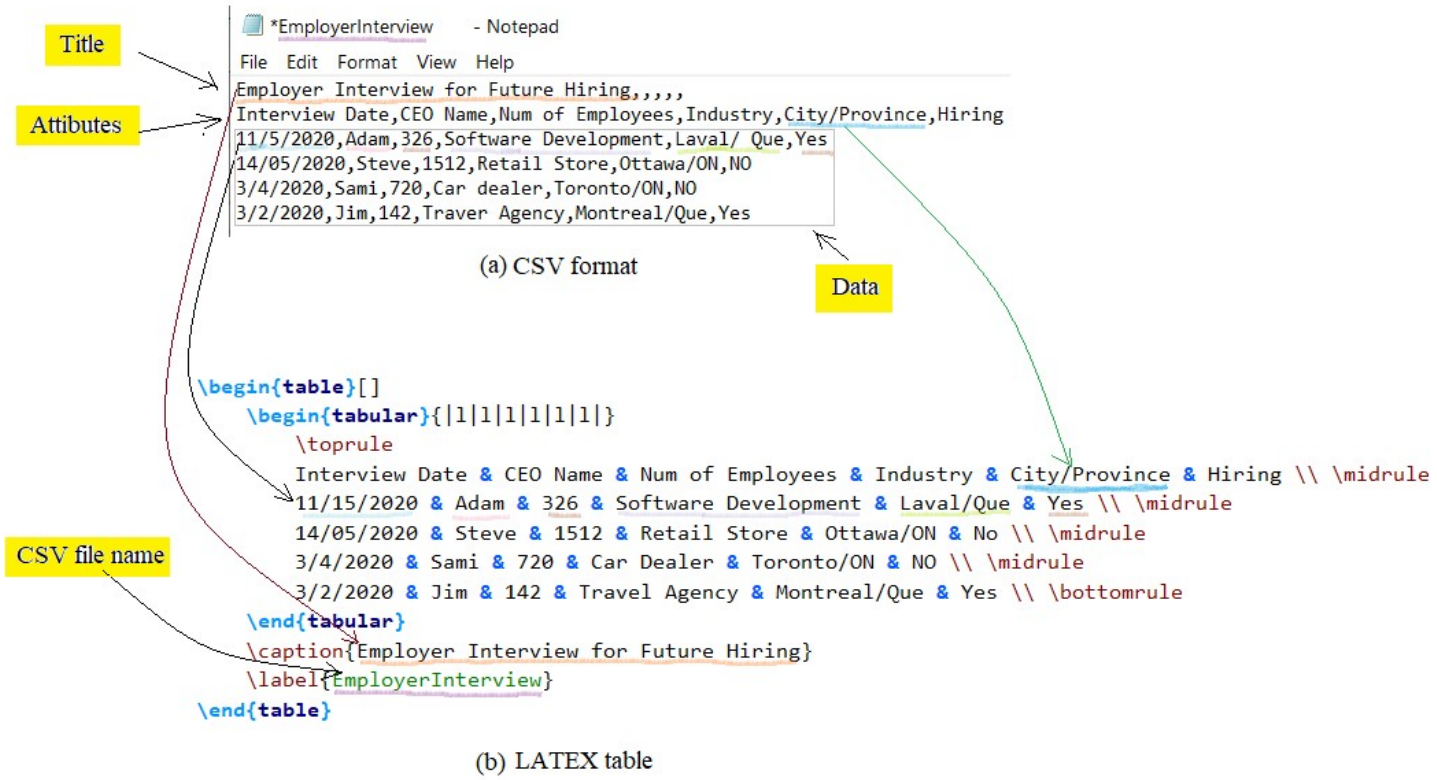



**Figure 2:** Sample file **(a)** in Excel format **(b)** in Comma Separated Values (CSV) format

The details of what you need to do and how your application should work are given below.

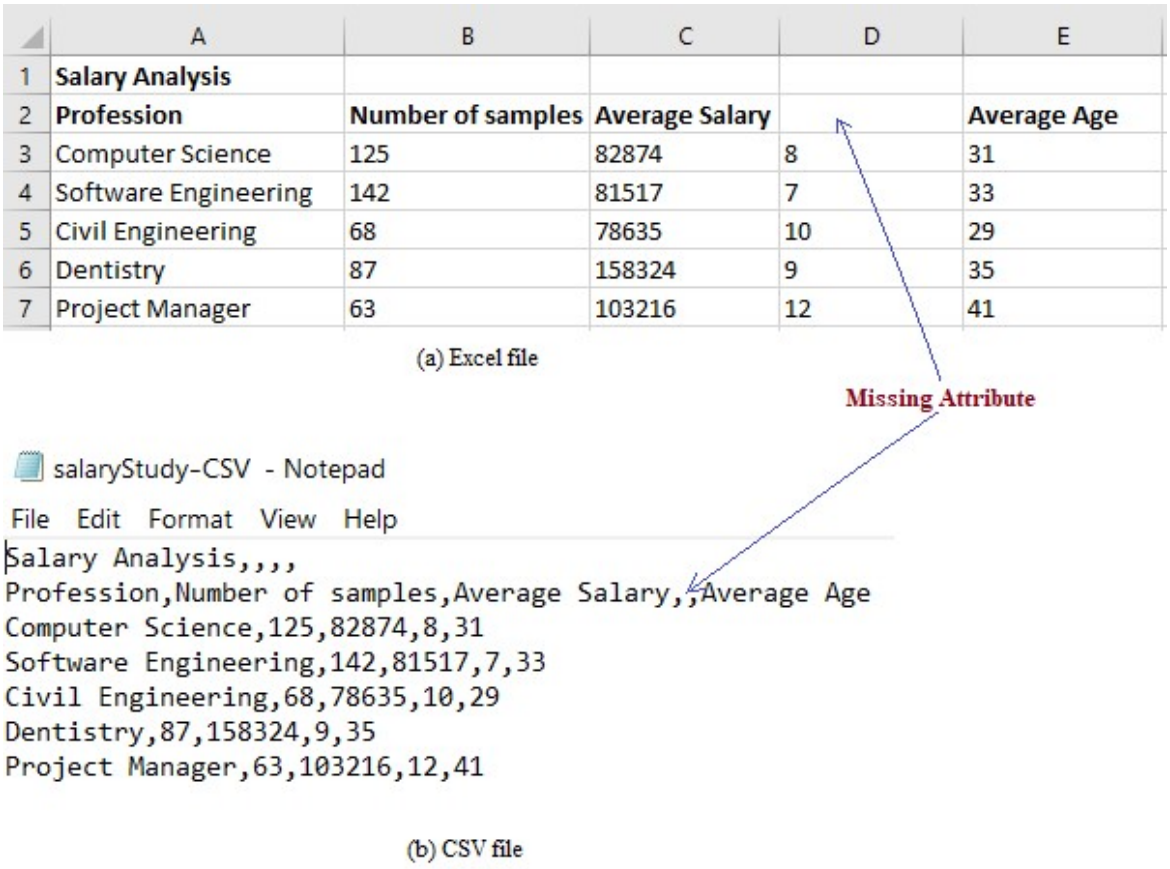
a. Each file may have one or more entries, the number is unknown before processing.

b. The first line of the CSV file contains title of the table, second line contains the attributes and rest of the lines contain the data (records).

c. In case of a missing attribute, the file should not be converted to LATEX. A message must be displayed to the user and name of the file must be saved in the log file. See Figure 3 for an example of a file with missing attribute.

d. In case of no missing attributes, data of the CSV file is transferred to a LATEX table. In case of missing data, the record should not be included in the LATEX table, however a message should be displayed to the user and the record must be saved in a log file. See Figure 4 for an exmaple of missing data.

Note: In this assignment you must dmonstrate your code with a minimum of two CSV files. For demonstration purposes, you will find two sample files *EmployerInterview.CSV* and *SalaryStudy.CSV* within inside the assignment zip. Your code must work on the two sample files as well as on any CSV file with arbitrary number of attributes and data records.



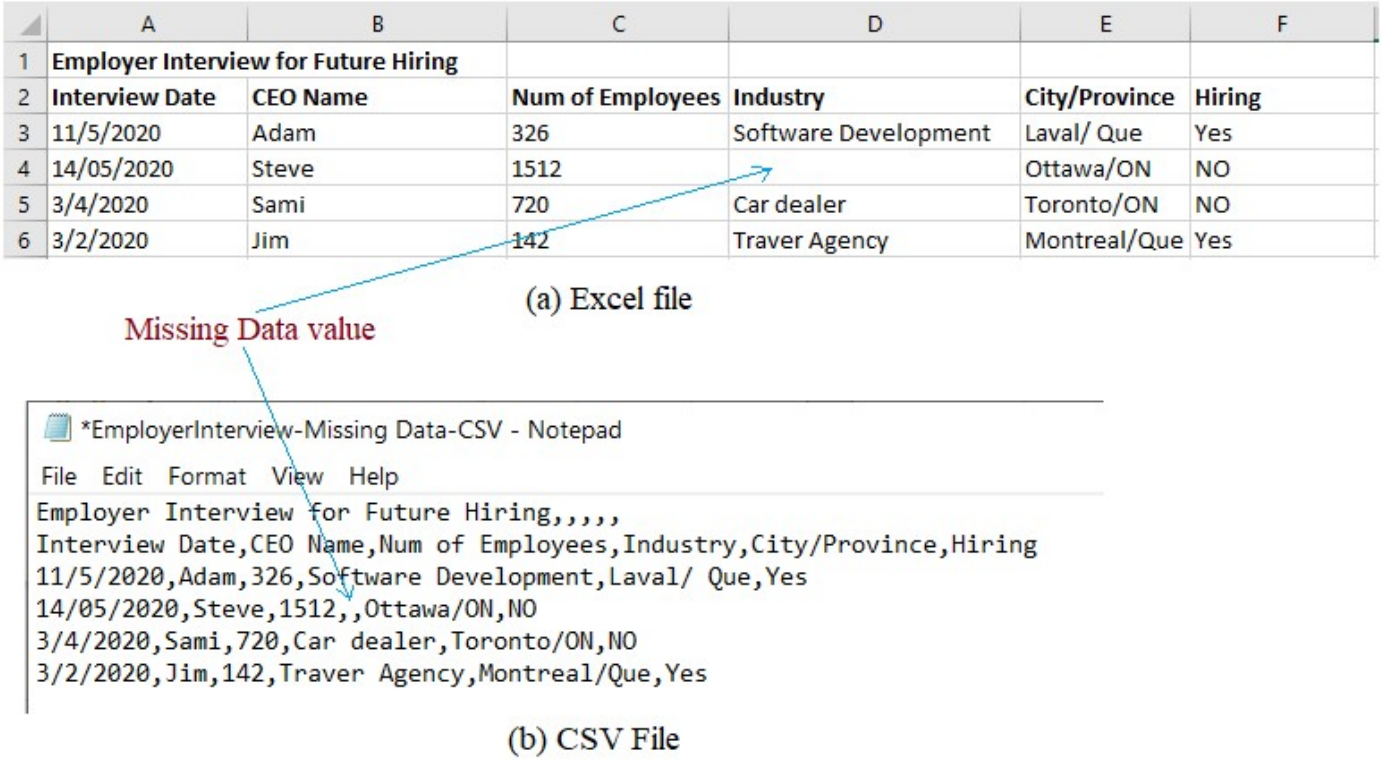
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Salary Analysis** | | | | |
| 2 | **Profession** | **Number of samples** | **Average Salary** | | **Average Age** |
| 3 | Computer Science | 125 | 82874 | 8 | 31 |
| 4 | Software Engineering | 142 | 81517 | 7 | 33 |
| 5 | Civil Engineering | 68 | 78635 | 10 | 29 |
| 6 | Dentistry | 87 | 158324 | 9 | 35 |
| 7 | Project Manager | 63 | 103216 | 12 | 41 |

(a) Excel file



(b) CSV file

**Figure 3:** Another sample file with a missing attribute. In this case the file will not be transformed to LaTeX.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Employer Interview for Future Hiring** | | | | | |
| 2 | **Interview Date** | **CEO Name** | **Num of Employees** | **Industry** | **City/Province** | **Hiring** |
| 3 | 11/5/2020 | Adam | 326 | Software Development | Laval/ Que | Yes |
| 4 | 14/05/2020 | Steve | 1512 | | Ottawa/ON | NO |
| 5 | 3/4/2020 | Sami | 720 | Car dealer | Toronto/ON | NO |
| 6 | 3/2/2020 | Jim | 142 | Traver Agency | Montreal/Que | Yes |

(a) Excel file



(b) CSV File

**Figure 4:** Sample file with missing data value. In this case the rest of the records are transformed to LATEX and the entry with missing data is reported to the log file

In your code write an exception class called **InvalidException**. The class should have sufficient constructors allow:

a. A default error message "*Error: Input row cannot be parsed due to missing information".* The message is to be stored in the thrown object; and

b. The passing of any different error message if desired. This is actually the constructor that you will be using throughout the assignment.

In main() method of the application, try to open input files (*EmployerInterview.CSV* and *SalaryStudy.CSV*) for reading. You must use the **Scanner** class for reading these files. If one of the files does not exist, the program must display an error message indicating "*Could not open input file xxxxx for reading. Please check if file exists! Program will terminate after closing any opened files.*", and then exits. You MUST however, close all opened files before exiting the program. For example, if *EmployerInterview.CSV* does not exist, then the following shows the behavior of the program:

```
Could not open file EmployerInterview.CSV for reading
Please check if file exists! Program will terminate after closing all files.
```

If the input files can successfully be opened, program will attempt to open/create the output files (*EmployerInterview*.tex and *SalaryStudy*.tex). You need to use **PrintWriter** to open these output files. If "any" of these output files cannot be created, then you must:

a. Display a message to the user indicating which file could not be opened/created;

b. Delete the created output file (if any) related to the file that cannot open. That is, if you cannot create all of these output files, then you must clean the directory by deleting all other created files;

c. Close all opened input files; then exit the program.

Write a method (you should take advantage of static throughout assignment!) called **processFilesForValidation(…)**. This method will represent the core engine for processing the input files and creating the output ones. You can pass any needed parameters to this method, and the method may return any needed information. This method however must NOT declare any exceptions. In other words, all needed handling of any exceptions that may occur within this method, must be handled by the method. In specific:

a. The method should work on already opened files;

b. A method must process each of these files to find out whether it is valid or not;

c. If a file is valid, then the method must create the corresponding LATEX files.

d. If a file is invalid, then the method must stop processing of this file, then must throw **CSVFileInvalidException** to display exception message and save exception information in log file. For example in Figure 3, user message is:

```
File SalaryStudy.CSV is invlaid: attribute is missing.
Profession,Number of samples,Average Salary,***,Average Age
File is not converted to LATEX.
```

e. If a file is valid but one of the records has missing data then the method throws **CSVDataMissing** exception. This record will not be converted to LATEX and a message is presented to the user and is saved in the log file. For the example in Figure 4, the user message is:

```
In file EmployerInterview.CSV line 4 not converted to LATEX : missing data.  And the log file will include:
File EmployerInterview.CSV line 4.
14/05/2020, Steve,1512,***,Ottawa/ON,NO
Missing: Industry
```

f. The method will then continue with the processing of the following records.

Finally, at this point, the program needs to ask user to enter name of one of the created output files to display. If the user enters an invalid name, a **FileNotFoundException** should be thrown; however, user is allowed a second and final chance to enter another name. If this second attempt also fails, then the program exits. You must however apply the following:

- If the entered file is valid, then your program must open this file for reading using the **BufferedReader** class. Do not use the Scanner class to read the file for that task.

General information in designing the code:

a. For processing of input files, you may want to use the **StringTokenizer** class;
b. You should minimize opening and closing the files as much as possible; a better mark will be given for that;
c. Don't use any external libraries or existing software to produce what is needed; that will directly result in a 0 mark!
d. Your program must work for any input files. CSV files provided with this assignment are only one possible version, and must not be considered as the general case when writing your code.

**General Guidelines When Writing Programs:**

- Include the following comments at the top of your source codes.

```
// -------------------------------------------------------
// Assignment (include number)
// Question: (include question/part number, if applicable)
// Written by: (include your name and student ID)
// -------------------------------------------------------
```

- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy-to-read format.
- End your program with a closing message so that the user knows that the program has terminated.

**JavaDoc Documentation:**

Documentation for your program must be written in **javaDoc**.
In addition, the following information must appear at the top of each file:

| | |
|---|---|
| Name(s) and ID(s) | (include full names and IDs) |
| COMP249 | |
| Assignment # | (include the assignment number) |
| Due Date | (include the due date for this assignment) |

## Submitting Assignment 3

- For this assignment, you are allowed to work individually, or in a group of a maximum of 2 students (i.e., you and one other student). You and your teammate must however be in the same section of the course. Groups of more than 2 students = zero mark for all members!
- Only electronic submissions will be accepted. Zip together the source codes.
- Naming convention for zip file: Create one zip file, containing all source files and produced documentations for your assignment using the following naming convention:
  The zip file should be called *a#_StudentName_StudentID*, where # is assignment number and *StudentName and StudentID* is your name and ID number respectively. Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. For example, for the first assignment, student 12345678 would submit a zip file named like: *a1_Mike-Simon_123456.zip.* if working in a group, the name should look like: *a1_Mike-Simon_12345678-AND-Linda-Jackson_98765432.zip.*
- Submit only ONE version of an assignment. If more than one version is submitted the first one will be graded, and all others will be disregarded.
- If working in a team, only one of the members can upload the assignment. Do **NOT** upload the file for each of the members!

⇨ **Important:** Following your submission, a demo is required (please refer to the courser outline for full details). The marker will inform you about the demo times. Please notice that failing to demo your assignment will result in zero mark regardless of your submission.

## **Evaluation Criteria for Assignment 3** (10 points)

| **Total** | **10 pts** |
|---|---|
| **JavaDoc** documentations | 1 pt |
| Producing proper LATEX output for CSV files | 2 pt |
| Generating and Handling exception: **LATEXFileInvalidException** | 2 pts |
| Generating and handling exception: **CSVDataMissing** | 2 pts |
| Generating and Handling exception: **FileNotFoundException** | 2 pt |
| General Quality of the Assignment | 1 pt |