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

---

1. Background Information

Since the start of the pandemic, the Public Relations (PR) Department at a local hospital has produced statistics on hospitalizations, ICU admissions, vaccinations, availability of doctors and nurses, and on other topics related to COVID-19.

The PR department has organized all the statistical data into several Excel tables, with each table saved in a text file using comma-separated values (CSV) file format, a common text file format used to store tabular data.

Each row in an Excel table is represented by a line in the CSV file with the row cells separated by a comma. Specifically, each line of a CSV file, called a **data row** or a **data record**, is composed of one or more **data fields**, separated by commas.

For example, the following line represent a data row with three data fields:

```
179, Montreal, Hello Word
```

If a data field in a data row contains commas, then that field in the CSV file is enclosed in double quotation marks; similarly, if a data field contains quotation marks as part of its data, then each internal quotation mark is doubled in the CSV file.

For example, the following line represents a data row with a data field that includes a comma

```
179, Montreal, "Hello, world"
```

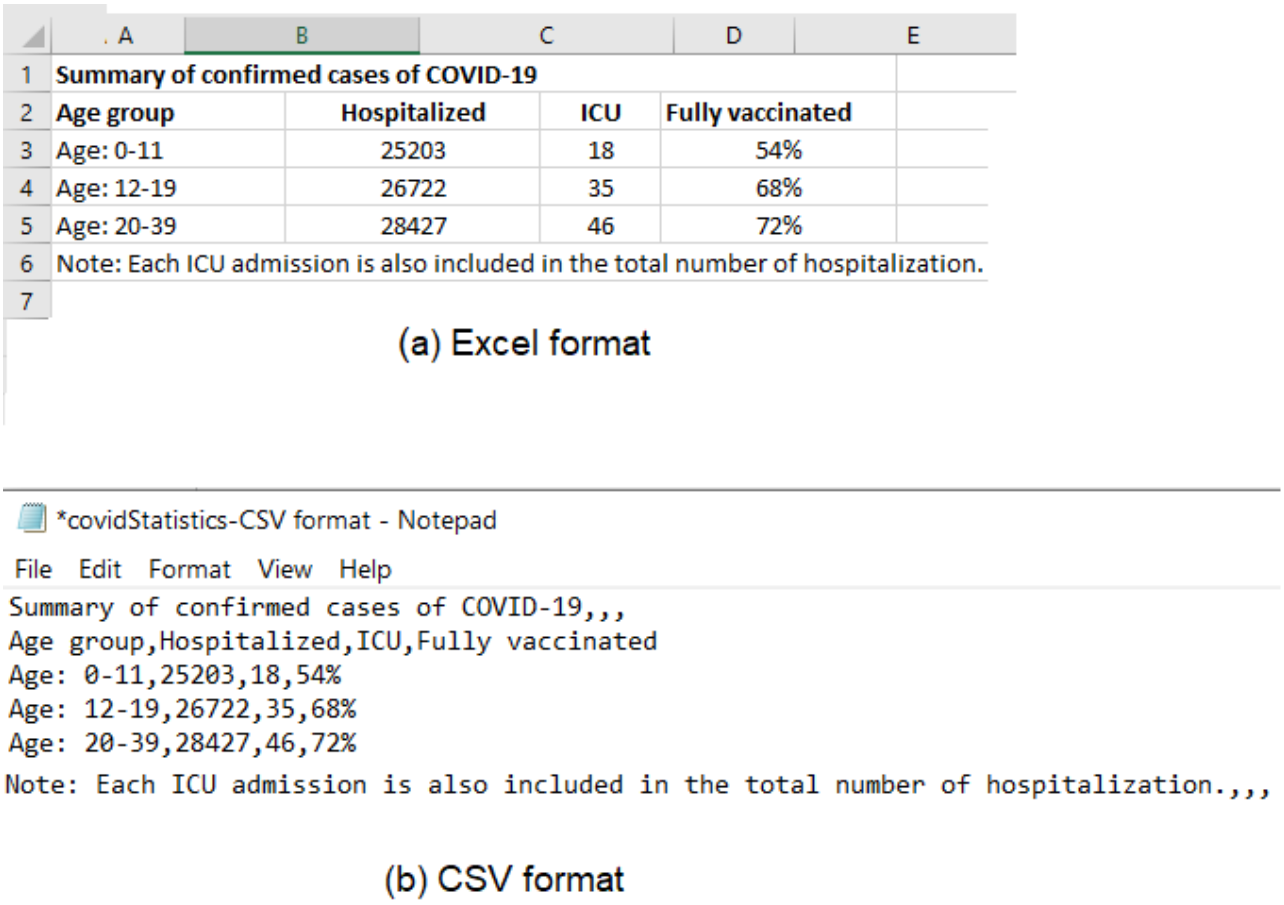
while the following line represents a data row with a data field that includes both a comma and quotation marks

```
179, Montreal, """Hello"", world"
```

where the data field `"Hello, world"` represents **Hello, world** and the data field `"""Hello"", world"` represents **"Hello", world**.

In this assignment you may assume that each data row consists of exactly four data fields, and that the data fields **DO NOT** include commas or quotation marks.

The table stored in a CSV file may have different number of data rows, with the first data row representing the **title** and the second data row representing the **attributes**. Figure 1 below shows one example of a table in both Excel and CSV formats.



	A	B	C	D	E
1	**Summary of confirmed cases of COVID-19**				
2	**Age group**	**Hospitalized**	**ICU**	**Fully vaccinated**	
3	Age: 0-11	25203	18	54%	
4	Age: 12-19	26722	35	68%	
5	Age: 20-39	28427	46	72%	
6	Note: Each ICU admission is also included in the total number of hospitalization.				
7					

(a) Excel format

*covidStatistics-CSV format - Notepad

File Edit Format View Help

```
Summary of confirmed cases of COVID-19,,,
Age group,Hospitalized,ICU,Fully vaccinated
Age: 0-11,25203,18,54%
Age: 12-19,26722,35,68%
Age: 20-39,28427,46,72%
Note: Each ICU admission is also included in the total number of hospitalization.,,,
```

(b) CSV format

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

2. Your Task

As a software designer, you have been tasked by the PR department to design and implement a program to transfer the contents of the Excel tables to the hospital's website in a best way possible. Based on your investigation, you have determined that the best option is to convert the Excel tables into HTML tables. You are therefore required to design and implement a Java tool called CSV2HTML to read and process CSV files and create the corresponding HTML tables. For information on HTML tables see: https://www.w3schools.com/html/html_tables.asp

3. Input Files

The input files consist of two or more text files in CSV format. The CSV files typically include attributes as described in Figure 1(a), but since they could be modified by the PR staff, your code must be able to accommodate any CSV file with any attributes.

In your design you may assume the following about the input CSV files:

a. There are at least three lines in a CSV file.
b. The first line represents the **Title** of the table.
c. The second line contains exactly four **attributes,** whose names may vary depending on the CSV file.
d. The third and possibly the following lines, if any, each represent the actual data records.
e. The last line may represent a **note** line if it begins with the text "**Note:**" in its first data field.

The title in a CSV file is stored in HTML `<caption>` tags, the attributes are stored in `<th>` tags, the data records in `<tr>` tags, the data fields in `<td>` tags, and finally the **note**, if any, is stored in `<span>` tags. See the correspondence between Figures 2 (a) and (b) .

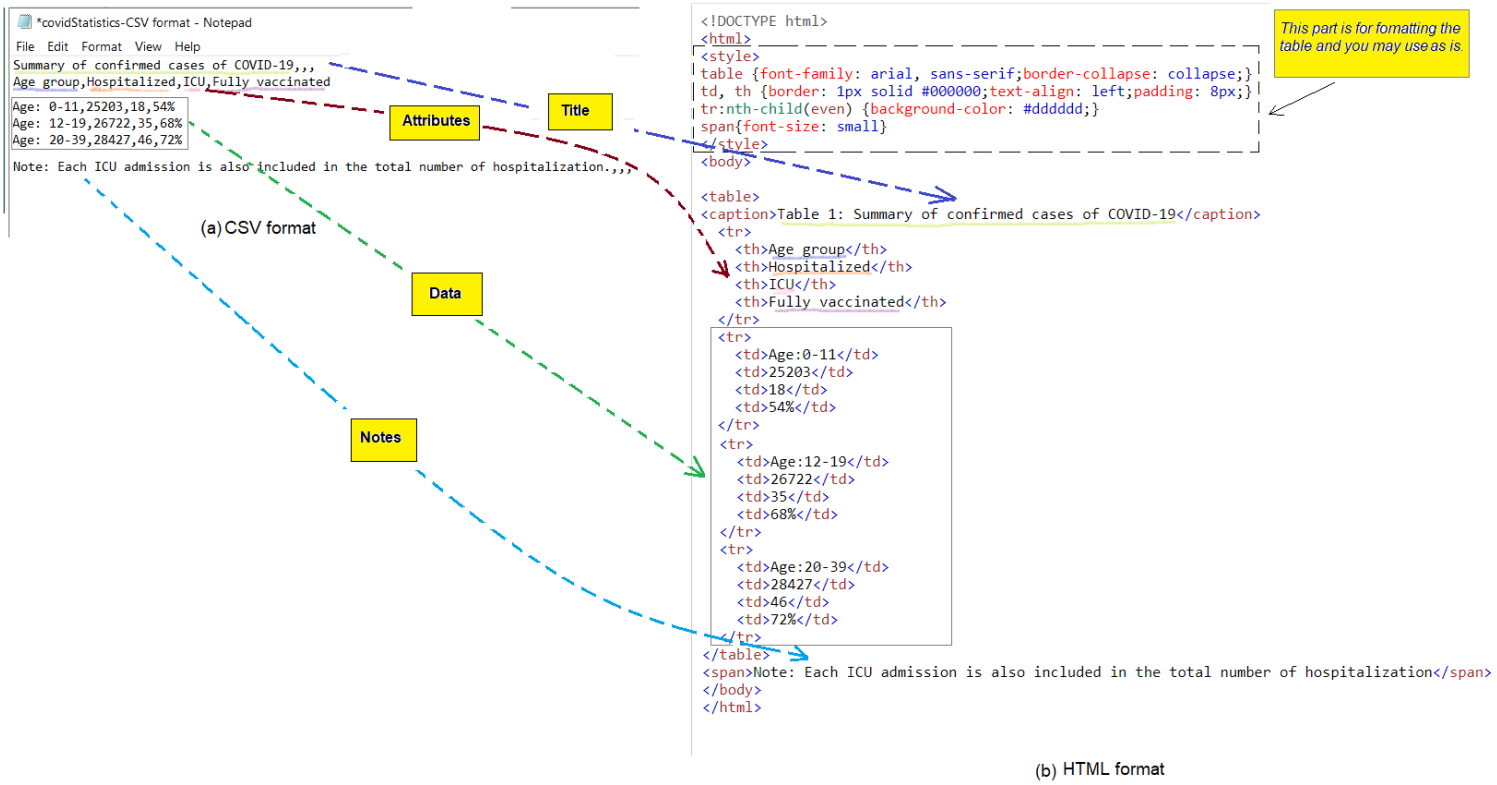


Figure 2: Sample file **(a)** in Comma Separated Values (CSV) format **(b)** HTML format

4. Output Files

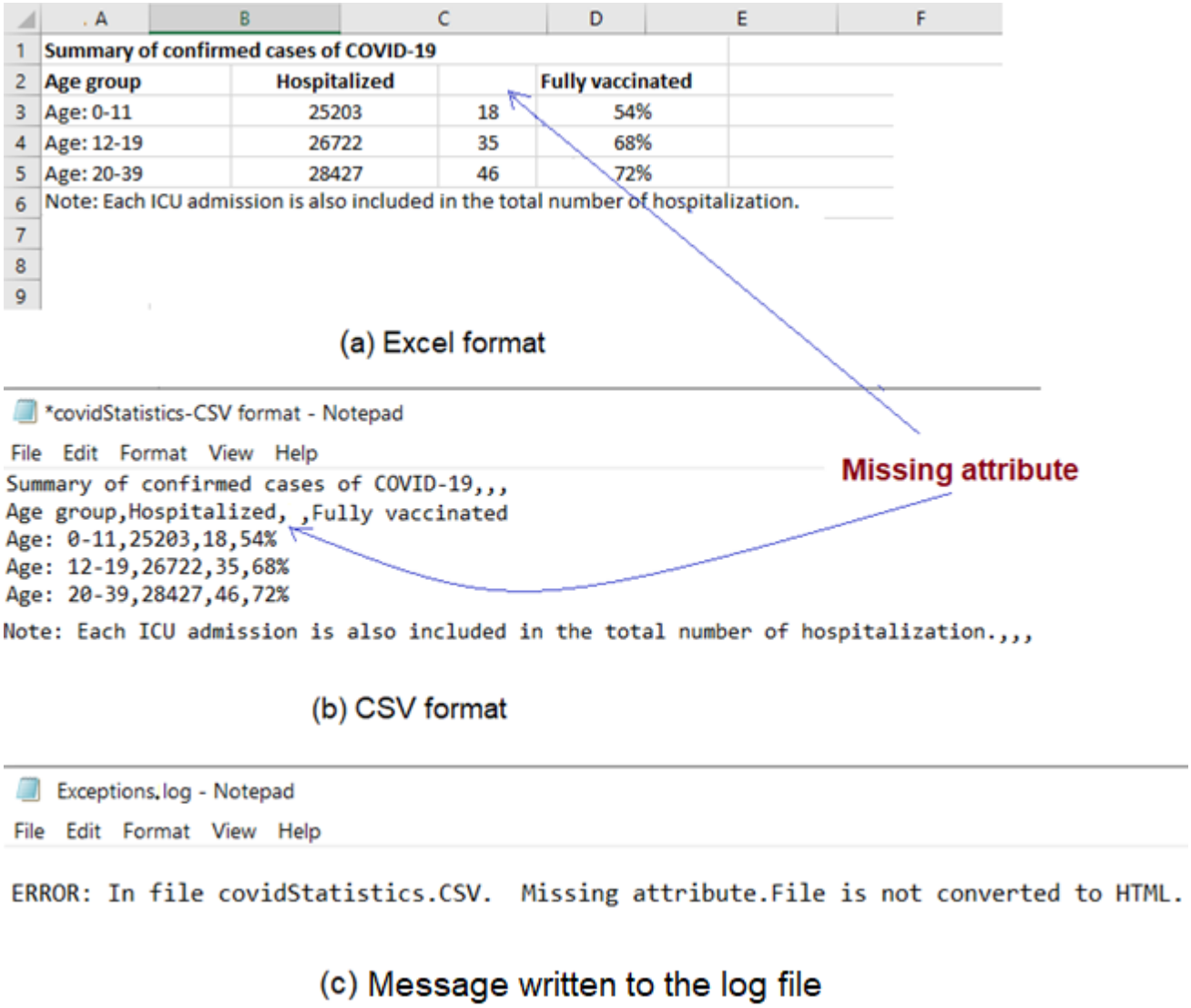
Given a number of CSV files, your program must generate the corresponding HTML files. The HTML files must have the same name as their corresponding CSV files but with the extension "**.html**". Figure 2 depicts a sample CSV file along with the corresponding HTML file.

Your program must also log particular runtime processing exceptions (more details below), if any, to a text file named "**`Exceptions.log`**". This log file must be opened for **appending** whether it exists or not, and regardless of whether your program encounters processing errors.

5. How to Process a CSV Files in Case of missing Attribute or Data?

a. In case of a **missing attribute**, the file should not be converted to HTML. Instead, a message must be displayed to the user and the name of the file must be written to the log file ("**`Exceptions.log`**"). See Figure 3 for an example of a file with missing attribute.

b. In case of **missing data**, the data record should not be included in the HTML table; however, a message should be both displayed to the user and written to the log file ("**`Exceptions.log`**"). See Figure 4 for an example of missing data and the specific error message associated with the missing data.



	A	B	C	D	E	F
1	**Summary of confirmed cases of COVID-19**					
2	**Age group**	**Hospitalized**		**Fully vaccinated**		
3	Age: 0-11	25203	18	54%		
4	Age: 12-19	26722	35	68%		
5	Age: 20-39	28427	46	72%		
6	Note: Each ICU admission is also included in the total number of hospitalization.					
7						
8						
9						

(a) Excel format

*covidStatistics-CSV format - Notepad

File Edit Format View Help

```
Summary of confirmed cases of COVID-19,,,
Age group,Hospitalized, ,Fully vaccinated
Age: 0-11,25203,18,54%
Age: 12-19,26722,35,68%
Age: 20-39,28427,46,72%
Note: Each ICU admission is also included in the total number of hospitalization.,,,
```

Missing attribute

(b) CSV format

Exceptions.log - Notepad

File Edit Format View Help

```
ERROR: In file covidStatistics.CSV.  Missing attribute.File is not converted to HTML.
```

(c) Message written to the log file

Figure 3: Sample file with a missing attribute. In this case the CSV file will not be transformed to HTML, but a message is displayed to the user and written to the log file.

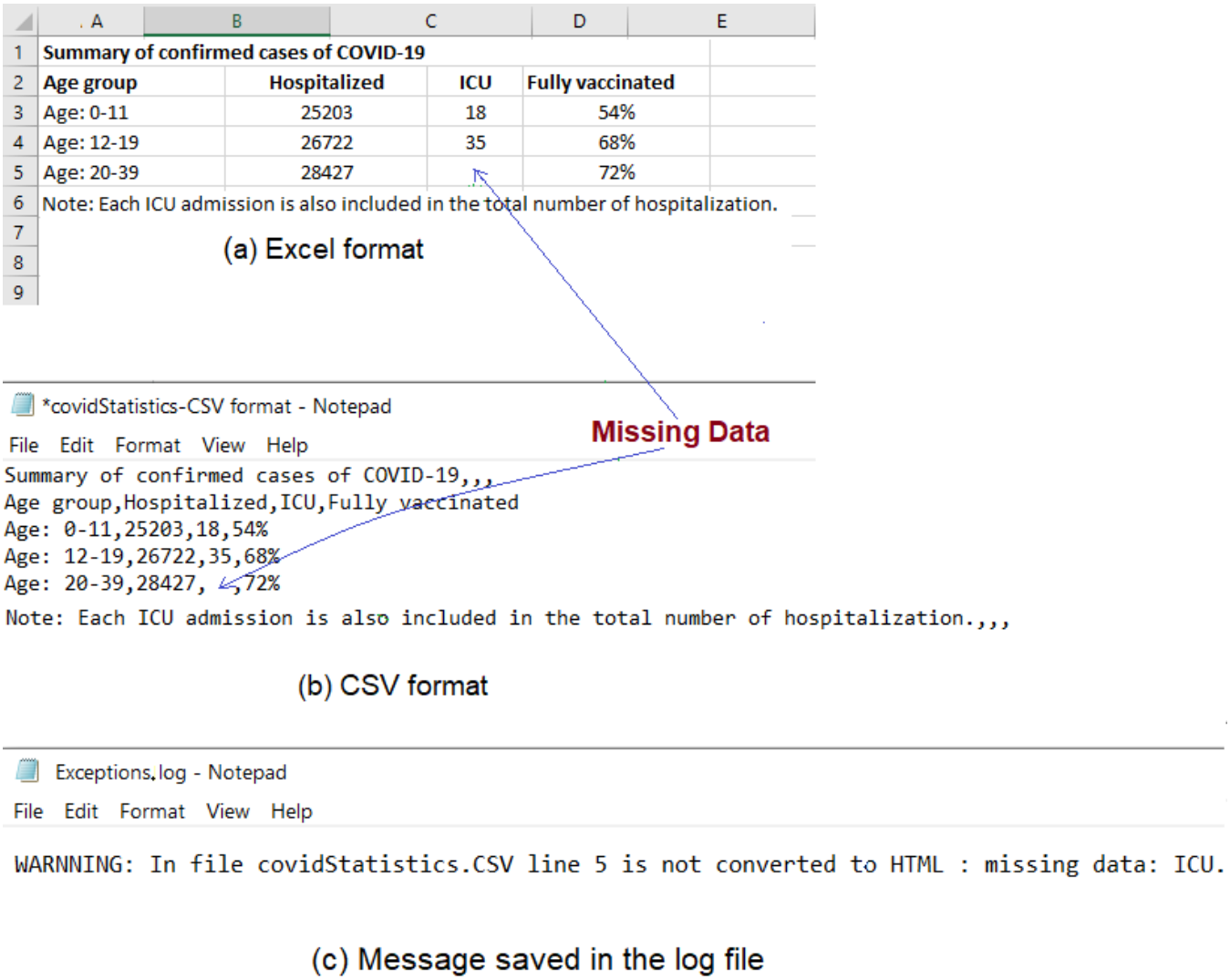


Figure 4: Sample file with missing value in a data field. In this case, the entire data record is ignored, and for each missing value, a message is written to the log file specifying the corresponding attribute name.

6. Requirements:

Requirement 1**: Input Files**

In this assignment you must demonstrate your program with a minimum of two CSV files. For demonstration purposes, you will find two sample files **`covidStatistics.`***CSV* and **`doctorList.`***CSV* included in the assignment zip file. Your code must work on the two sample files as well as on any number of CSV file with arbitrary number data records.

Requirement 2**: Programmer-Defined Exceptions**

Write two exception classes called **CSVAttributeMissing** and **CSVDataMissing**. These classes should have appropriate constructors to allow:

a. A default error message that reads "*Error: Input row cannot be parsed due to missing information*".
b. A programmer supplied message. This is actually the constructor that you will be using throughout this assignment. The behavior of the two exceptions is presented in Requirement 4 below.

Requirement 3**: Opening the input and output streams**

In the method main() of the application, use a **Scanner** object to open the input files (**`covidStatistics`**.*CSV* and **`doctorList`**.*CSV*) for reading. If either of the files does not exist or is not readable, then your program must display the following error message and then terminate:

Could not open input file xxxxx for reading.
Please check that the file exists and is readable. This program will terminate after closing any opened files.

However, you must close all the opened files before exiting the program. For example, if **`covidStatistics`**.*CSV* does not exist, then the following shows the behavior of the program:

Could not open file covidStatistics.CSV for reading
Please check that the file exists and is readable. This program will terminate after closing any opened files.

If the input files have successfully been opened, then your program will attempt to open/create all of the output files (**`covidStatistics`**.html, **`doctorList`**.html and **`Exceptions`**.log). You will 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. Clean the directory by deleting all of the other output files that have already been successfully created.

c. Close all the opened input files, and then exit the program.

Requirement 4: **Creating the HTML file**

Write a method called **ConvertCSVtoHTML** This method will represent the core engine for processing the input files and creating the output files. You can pass any needed parameters to this method, and the method may return any needed information. This method however must declare two exceptions **`CSVAttributeMissing`** and **`CSVDataMissing`**. In other words, any exceptions that may occur within this method must be handled by the calling method . Specifically:

a. The method should work on the files that are already open;

b. The method must process each of these files to find out whether it is valid or not. (Here valid means no missing attributes, remember the number of attributes must be four)

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

d. If a file is invalid, then the method must stop processing this file, and must throw a **`CSVAttributeMissing`** exception to display an exception message and save the exception information in the log file. For example in Fig. 3, the following message is displayed to the user and saved in the log file :

ERROR: In file covidStatistics.CSV. Missing attribute.File is not converted to HTML.

e. If a file is valid but one of the data rows has missing data then, the method throws a **`CSVDataMissing`** exception. This data row will not be converted to HTML and a message is both presented to the user and written to the log file. For the example in Figure 4, the following message is displayed to the user and saved in the log file:

WARNNING: In file covidStatistics.CSV line 5 is not converted to HTML: missing data: ICU.

The method will then continue with the processing of the remaining data rows, if any.

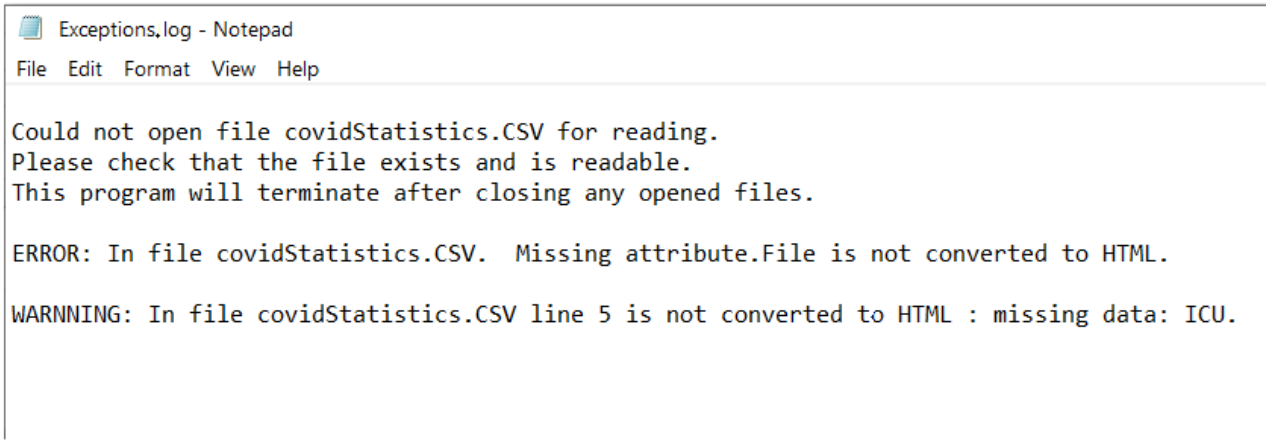


Figure 5: Sample log file

Requirement 5**: Displaying the HTMl files**

Finally, your program must display one of the output file on the screen. So your program will prompt the user to enter the name of one of the output files created above.

- If the user enters an invalid name, a **`FileNotFoundException`** should be thrown, but the user must be allowed a second and final chance to enter another file name. If this second attempt also fails, then the program must terminate.

- Otherwise, if the entered file name is valid, then your program must open this file for reading using the **BufferedReader** class, outputting its contents on the screen. Do not use the **Scanner** class to read the file for this task.

7. General information about designing the code

a. For processing of input files, you may want to use the **StringTokenizer** class or the **split** method. But you **CANNOT** use lists, Hash tables, hash maps, collections, or any other built-in data types.

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 required; that will directly result in a 0 mark!

d. Your program must work for any input files. The CSV files provided with this assignment are only one possible versions, and must not be considered as the general case when writing your code.

8. 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).
- 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 HTML output for CSV files	2 pt
Generating and Handling exception: **`CSVAttributeMissing`**	2 pts
Generating and handling exception: **`CSVDataMissing`**	2 pts
Generating and Handling exception: **`FileNotFoundException`**	2 pt
General Quality of the Assignment	1 pt