Concordia University
Comp 249 – Winter 2021
Object-Oriented Programming II
Assignment 3
Due 11:59 PM - Wednesday March 31, 2021

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 car rental company uses excel spreadsheets to store and access data related to the daily operations of the company. The company management realized that using Excel alone brings many challenges such as: manual entry of data, difficulty in performing in depth analysis of data, cell and formula errors, limited security, and inefficient sharing of data between employees in different branches. To answer the challenges, the company decided to store all the data using a cloud server with a NoSQL database as shown in Figure 1.

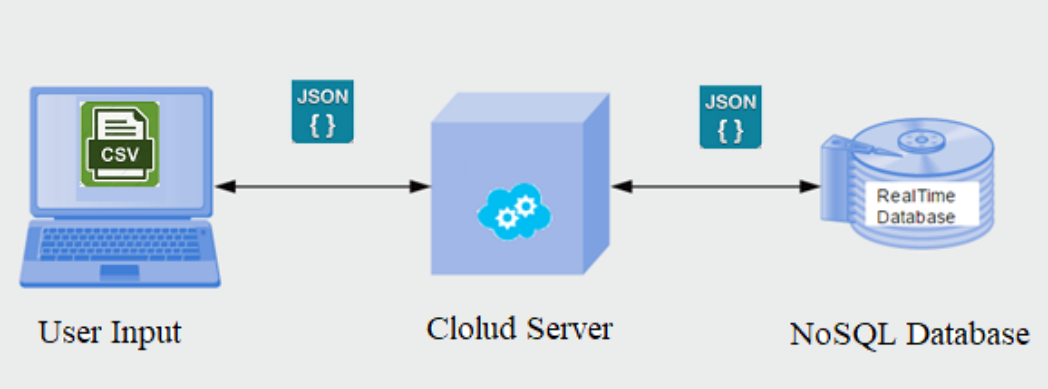


Figure 1: Server architecture with a NoSQL Database that uses JSON format to store data.

The NoSQL database uses **JSON** (**J**ava**S**cript **O**bject **N**otation) format to store the data**. JSON** is a well-known, lightweight data-interchange format, which is easily readable/writable by humans. It is also easy for machines to parse and generate. JSON is based on data objects consisting of **attribute**–**value** pairs and array data types. It is widely used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.

As a software designer, you are hired to design and implement a Java code to read the data from the excel sheets to JSON format. The excel sheet is saved in comma-separated values (CSV) format. The CSV format 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 CSV2JSON to help the car rental company read and process CSV files and create the corresponding JSON files. The company has two main files, the first file is "`Car Rental Records.CSV`" that keeps the history of renting the cars. The file contains the follwoing attributes:

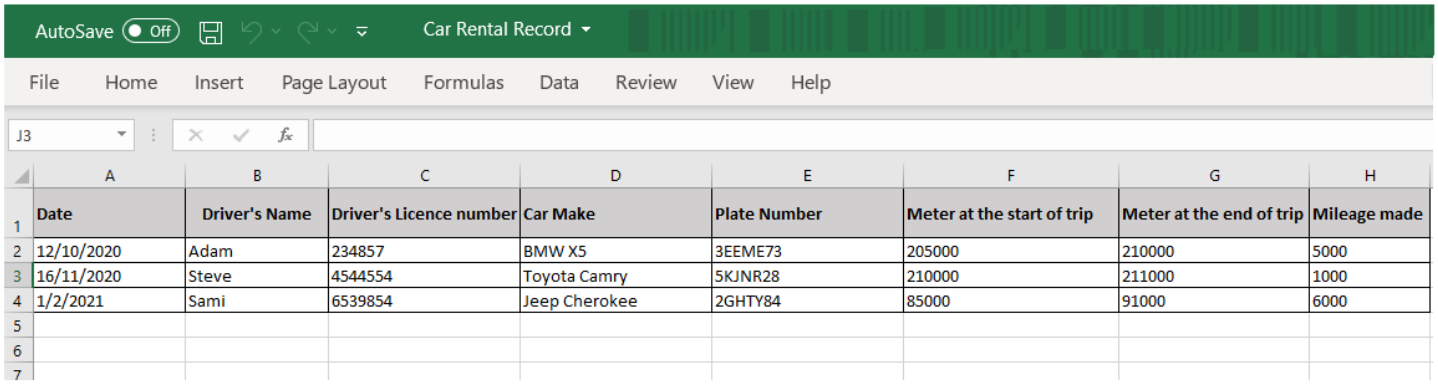
> Date, Driver's Name, Driver's Licence number, Car Make, Plate Number, Meter at the start of trip, Meter at the end of trip and Mileage made.

Figure 2 shows an overview of the file in both Excel and CSV formats.

The second file is "`Car Maintenance Records.CSV`" which contains every car's maintenance history. The file contains the following attributes:

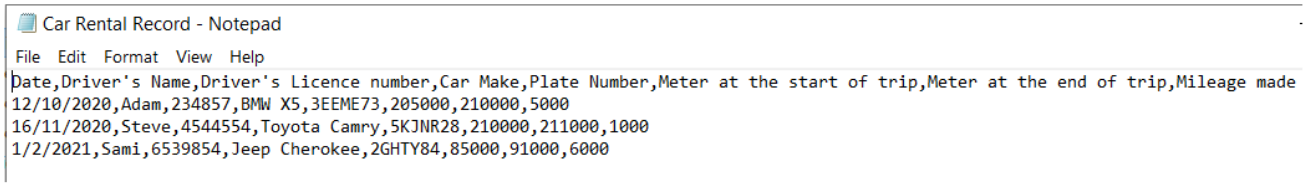
> Plate Number, Maintenance summary, Workshop, Phone number, Address, Labor cost, Material cost and Total Cost

Figure 3 shows an overview of the file in both Excel and CSV formats.



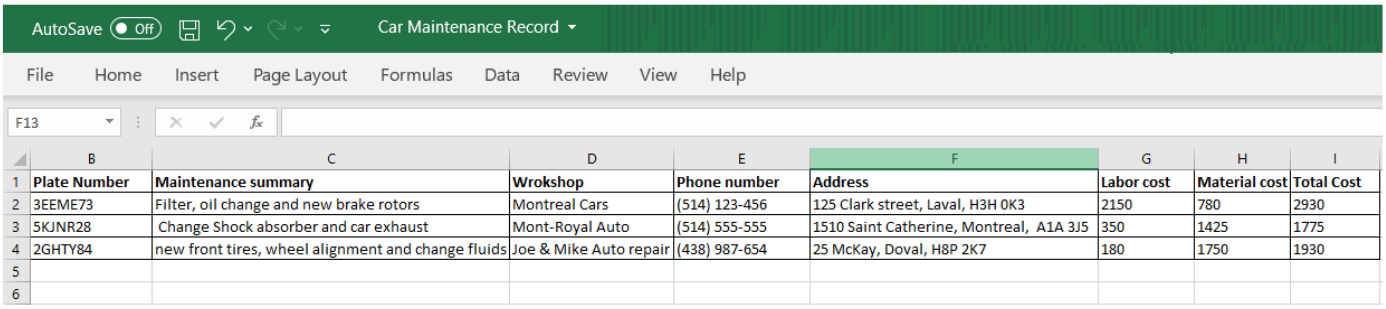
	A	B	C	D	E	F	G	H
1	Date	Driver's Name	Driver's Licence number	Car Make	Plate Number	Meter at the start of trip	Meter at the end of trip	Mileage made
2	12/10/2020	Adam	234857	BMW X5	3EEME73	205000	210000	5000
3	16/11/2020	Steve	4544554	Toyota Camry	5KJNR28	210000	211000	1000
4	1/2/2021	Sami	6539854	Jeep Cherokee	2GHTY84	85000	91000	6000
5								
6								
7								

(a) Excel file



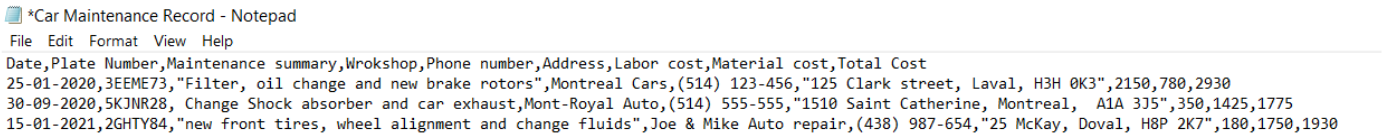
(b) Comma Seperated Values

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



	B	C	D	E	F	G	H	I
1	**Plate Number**	**Maintenance summary**	**Wrokshop**	**Phone number**	**Address**	**Labor cost**	**Material cost**	**Total Cost**
2	3EEME73	Filter, oil change and new brake rotors	Montreal Cars	(514) 123-456	125 Clark street, Laval, H3H 0K3	2150	780	2930
3	5KJNR28	Change Shock absorber and car exhaust	Mont-Royal Auto	(514) 555-555	1510 Saint Catherine, Montreal, A1A 3J5	350	1425	1775
4	2GHTY84	new front tires, wheel alignment and change fluids	Joe & Mike Auto repair	(438) 987-654	25 McKay, Doval, H8P 2K7	180	1750	1930
5								
6								

(a) Excel format



(b) Comma Sperated Values format

Figure 3: Car Maintenance sample file **(a)** in Excel format **(b)** in Comma Separated Values (CSV) format

The input and output of your code are as follows:

Input: two or more file in CSV format. The files typically include attributes as described in Figure 2(b) and Figure 3 (b), however the attributes could be modified by the user and your code must be able to accommodate any CSV file with any attributes. However, you may assume that the first line of the CSV file is composed of the attributes and the rest of the lines are the data.

Output: Your code must generate the JSON file for each CSV file, the JSON files must be called same name as the corresponding CSV files but with extension ".json". The corresponding ".json" files for the CSV files in Figure 2(b) and Figure 3 (b) are presented in Figure 4 (a) and (b) respectively. Note the ".json" file is composed of an array of JSON objects, each line of data in the CSV file is represented as a JSON object in the array.

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 the attributes and the rest of the lines contain the data (records).
c. In case of a missing attribute, the file should not be converted to JSON. A message must be displayed to the user and the name of the file must be saved in the log file. See Figure 5 for an example of a file with missing attribute.

d. Each line of data (record) must be converted into a JSON object and saved in a JSON array. In case of missing data, the record should not be transferred to JSON object, however a message should be displayed to the user and the record nust be saved in a log file. See Figure 6 for an exmaple of missing data.

CarRentel File - Notepad

File Edit Format View Help


```
[
  {
    "Date": "12/10/2020",
    "Driver's Name": "Adam",
    "Driver's Licence number": 234857,
    "Car Make": "BMW X5",
    "Plate Number": "3EEME73",
    "Meter at the start of trip": 205000,
    "Meter at the end of trip": 210000,
    "Mileage made": 5000
  },
  {
    "Date": "16/11/2020",
    "Driver's Name": "Steve",
    "Driver's Licence number": 4544554,
    "Car Make": "Toyota Camry",
    "Plate Number": "5KJNR28",
    "Meter at the start of trip": 210000,
    "Meter at the end of trip": 211000,
    "Mileage made": 1000
  },
  {
    "Date": "1/2/2021",
    "Driver's Name": "Sami",
    "Driver's Licence number": 6539854,
    "Car Make": "Jeep Cherokee",
    "Plate Number": "2GHTY84",
    "Meter at the start of trip": 85000,
    "Meter at the end of trip": 91000,
    "Mileage made": 6000
  }
]
```


(a)

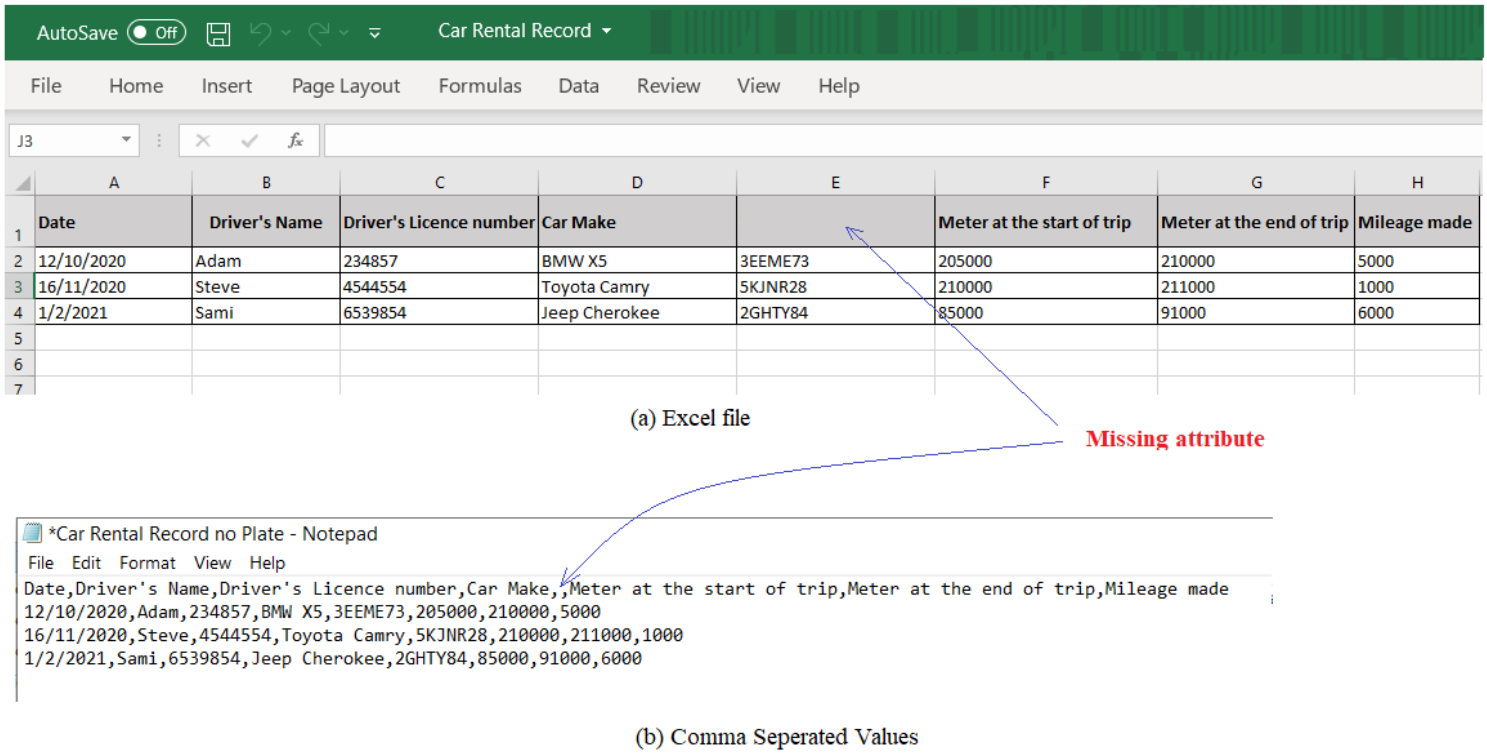
*Car Maintenance Record.json - Notepad

File Edit Format View Help


```
[
  {
    "Date": "25-01-2020",
    "Plate Number": "3EEME73",
    "Maintenance summary": "Filter, oil change and new brake rotors",
    "Wrokshop": "Montreal Cars",
    "Phone number": "(514) 123-456",
    "Address": "125 Clark street, Laval, H3H 0K3",
    "Labor cost": 2150,
    "Material cost": 780,
    "Total Cost": 2930
  },
  {
    "Date": "30-09-2020",
    "Plate Number": "5KJNR28",
    "Maintenance summary": "Change Shock absorber and car exhaust",
    "Wrokshop": "Mont-Royal Auto",
    "Phone number": "(514) 555-555",
    "Address": "1510 Saint Catherine, Montreal,  A1A 3J5",
    "Labor cost": 350,
    "Material cost": 1425,
    "Total Cost": 1775
  },
  {
    "Date": "15-01-2021",
    "Plate Number": "2GHTY84",
    "Maintenance summary": "new front tires, wheel alignment and change fluids",
    "Wrokshop": "Joe & Mike Auto repair",
    "Phone number": "(438) 987-654",
    "Address": "25 McKay, Doval, H8P 2K7",
    "Labor cost": 180,
    "Material cost": 1750,
    "Total Cost": 1930
  }
]
```


(b)

Figure 4: JSON files **(a)** of Car Rental Record **(b)** of Car Maintenance Record

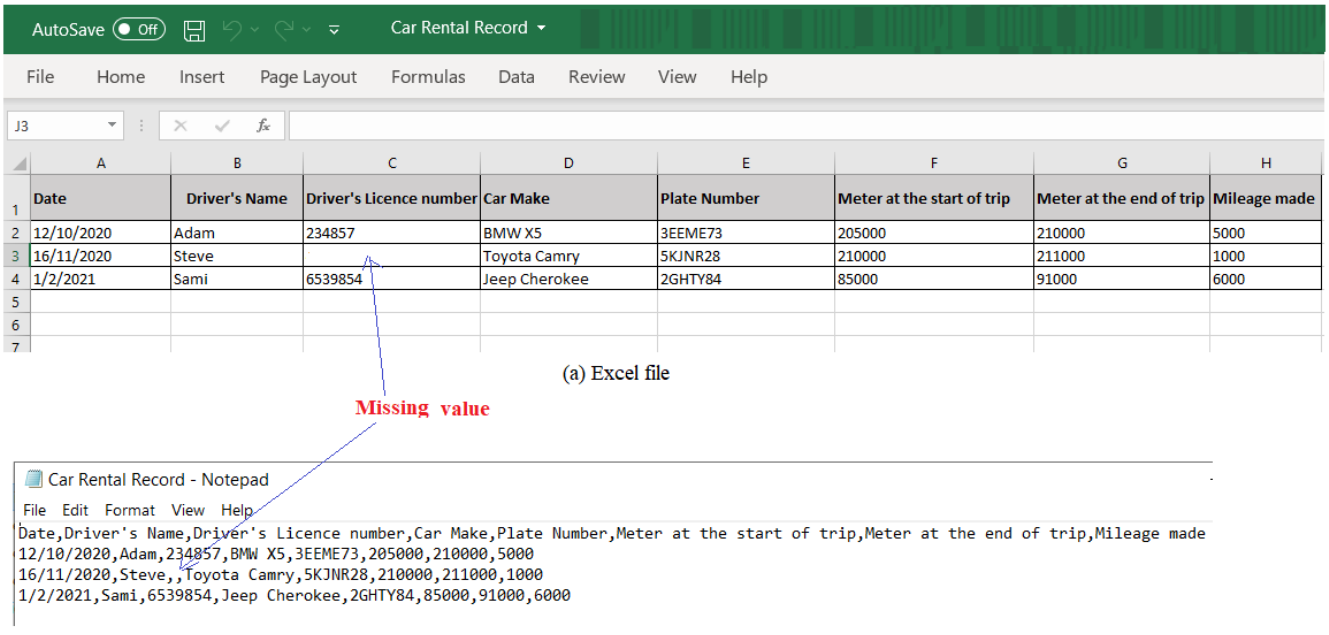


	A	B	C	D	E	F	G	H
1	Date	Driver's Name	Driver's Licence number	Car Make		Meter at the start of trip	Meter at the end of trip	Mileage made
2	12/10/2020	Adam	234857	BMW X5	3EEME73	205000	210000	5000
3	16/11/2020	Steve	4544554	Toyota Camry	5KJNR28	210000	211000	1000
4	1/2/2021	Sami	6539854	Jeep Cherokee	2GHTY84	85000	91000	6000
5								
6								
7								

(a) Excel file

(b) Comma Seperated Values

Figure 5: A sample file with a missing attribute. In this case the file will not be transformed to JSON.



	A	B	C	D	E	F	G	H
1	Date	Driver's Name	Driver's Licence number	Car Make	Plate Number	Meter at the start of trip	Meter at the end of trip	Mileage made
2	12/10/2020	Adam	234857	BMW X5	3EEME73	205000	210000	5000
3	16/11/2020	Steve		Toyota Camry	5KJNR28	210000	211000	1000
4	1/2/2021	Sami	6539854	Jeep Cherokee	2GHTY84	85000	91000	6000
5								
6								
7								

(a) Excel file

(b) Comma Seperated Values

Figure 6: Sample file with missing data value. In this case the rest of the records are transformed to JSON and the entry with missing data is reported to the log file

In this assignment you must 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 the main() method of the application, attempt to open the input files (*CarRentalRecord.CSV* and *CarMaintenancRecord.CSV*) for reading. You need to 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 *CarRentalRecord.CSV* does not exist, then the following image shows the behavior of the program.

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

If the input files can successfully be opened, the program will attempt to open/create the output files (*CarRentalRecord*.json and *CarMaintenanceRecord*.json). 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 the entire 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 the 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 JSON files.
d. If a file is invalid, then the method must stop processing of this file, then throws **CSVFileInvalidException** to display the exception error message and save information about this exception in the log file.

For the example in Figure 5, the user message is:

File CarRentalRecors.CSV is invlaid: field is missing.
File is not converted to JSON.

The log file will include the following:

File CarRentalRecors.CSV is invalid.
Missing field: 7 detected, 1 missing
Date, Driver's Name, Driver's Licence number, Car Make,***, Meter at the start of the trip, Meter at the end of the trip, Mileage made

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

For the example in Figure 6, the user message is:

In file CarRentalRecords.CSV line 2 not converted to JSON : missing data.

And the log file will include:

In file CarRentalRecords.CSV line 2
16/11/2020 Steve *** Toyota Camry 5KJNR28 210000 211000 1000
Missing: Driver's Licence number

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

Finally, at this point, the program needs to ask the user to enter the name of one of the created output files to display. If the user enters an invalid name, a **FileNotFoundException** should be thrown; however, the 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 the processing of input files, you may want to use the **StringTokenizer** class;
b. <u>You should minimize opening and closing the files as much as possible; a better mark will be given for that</u>;
c. Do not use any external libraries or existing software to produce what is needed; that will directly result in a 0 mark!
d. Again, your program must work for any input files. The CSV files provided with this assignment are only one possible version, and must not be considered as the general case when writing your code.

<u>General Guidelines When Writing Programs</u>

- Include the following comments at the top of each class you are writing.

```
// -------------------------------------------------
// Part: (include Part Number)
```

```
// Written by: (include your name(s) and student ID(s))
// -------------------------------------------------
```

- Use **JavaDoc** to create the documentations of your program. Use appropriate comments when needed.
- Display clear prompts for the user whenever you are expecting the user to enter data from the keyboard.
- All outputs 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.

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). Groups of more than 2 students = zero mark for all members! 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.
- Students will have to submit their assignments (one copy per group) using Moodle. Assignments must be submitted in the right DropBox/folder of the assignments. **Assignments uploaded to an incorrect DropBox/folder will not be marked and result in a zero mark. No resubmissions will be allowed.**
- 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 the number of the assignment and *StudentName/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.*

- 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 (Please read very carefully): Additionally, which is very important, a demo will take place with the markers afterwards. Markers will inform you about the details of demo time and how to book a time slot for your demo. If working in a group, both members must be present during demo time. Different marks may be assigned to teammates based on this demo.

- **If you fail to demo, a zero mark is assigned regardless of your submission.**
- **If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.**
- **Failing to demo at the second appointment will result in zero marks and no more chances will be given under any conditions.**

Evaluation Criteria for Assignment 3 (10 points)

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