

To begin with let's detail the CSV file we are going to be using. The first file is GPS.CSV file, this file contains information about cars and their location at a certain time, the second file is CC_DATA.CSV file that holds information about credit card transaction, the names of the people, the name of the shop, timestamp and price, finally the third file is the CAR-ASSIGNMENT.CSV file, it's linking people to their car id and people's job.

Our objective is to get an approximate location of the shops using this data. To do so, we will be programming JAR file that will process the data in order for us to get results, to execute the JAR file in an optimise way we will be using the framework.

The different JAR file that will be used :

- **group.jar** : This jar file will group the lines according to their common argument in a given column and write it in a new file.
@param 0 Input file location.
@param 1 Output file location.
@param 2 Index of the Id column in the input file.
- **sortGroupChronologically.jar** : This jar file will sort a each group from a grouped file chronologically.
@param 0 Input file location.
@param 1 Output file location.
@param 2 Index of the Id column in the input file.
@param 3 Index of the time column in the input file.
@param 4 Format of the time.
- **getJourney.jar** : This jar file will find every journey for each individual in the file, a journey is defined by a starting point and time, ending point and time and every journey in separated by a defined amount of time given by the user.
@param 0 Input file location.
@param 1 Output file location.
@param 2 Amount of time between 2 journey.
@all subsequent param the program will only get the journey from those id.
- **getDistinctValue.jar** : This jar file will find every distinct value of a given column and write them into a new file
@param 0 Input file location.
@param 1 Output file location.
@param 2 Index of the column.
- **deducePlacesLocation.jar** : This jar file will deduces the approximative location of places with the given data.
@param 0 Input file location. Credit card info grouped and sorted chronologically.
@param 1 Input file location. Journey information grouped and sorted chronologically.
@param 2 Index of the Id column in the credit card file

@param 3 Index of the Id column in the journey file
 @param 4 Index of the time column in the credit card file
 @param 5 Index of the time journey start column in the journey file
 @param 6 Index of the time journey end column in the journey file
 @param 7 Index of the latitude end of the journey column in the journey file
 @param 8 Index of the longitude end of the journey end column in the journey file
 @param 9 Format of the time in the credit card file
 @param 10 Format of the time in the journey file
 @param 11 Output file location.

- **regroupDeduction.jar** : This jar file will regroup the deduction find with the deducePlacesLocation.jar file and write them into a new file. To do so it'll read the input file, and get the average of all the location find for a place. Note : the deduction file need to be grouped by place before executing this file.

@param 0 Input file location.
 @param 1 Output file location.

- **replaceNameById.jar** : This jar file will replace all the names of the people in the credit card transaction file by their id, it will write this in a new file.

@param 0 Input file location. Credit card transaction.
 @param 1 Input file location. Assignment of the car to the people.
 @param 2 Output file location.

The reverseJAR file that will be used :

- **reverse1to1.jar** : This jar file tell the user how a line resulted in an other file after any 1 to 1 computation.

@param 0 Input file location.
 @param 1 Output file location.

- **reverseWithIndication.jar** : This jar file tell the user how a line resulted in an other file after any computation if an indication on the line is given.

@param 0 Input file location.
 @param 1 Output file location.

Using those JAR file in the right order will allow us to get a file holding an approximate location of all the shops.

deductionGroupedRegrouped						
	A	B	C	D	E	F
1	place,latitude,longitude					
2	Brew've Been Served,36.05407865,24.90118424					
3	Kalami Kafenia,36.06584491,24.85238533					
4	Hallowed Grounds,36.063708,24.8859479					
5	Octavio's Office Supplies,36.05852747,24.85788532					
6	Guy's Gyros,36.05573985,24.90257457					
7	Roberts and Sons,36.06326609,24.85228891					
8	Katerina's Café,36.05446454,24.89987552					
9	Abila Zacharo,36.06342408,24.85099602					
10	Hippokampos,36.07667875,24.85764175					
11	Bean There Done That,36.04802319,24.87957127					
12	Jack's Magical Beans,36.04803009,24.87957617					
13	Gelatogalore,36.05980651,24.85799138					
14	Ouzeri Elian,36.05197704,24.8708008					
15	Shoppers' Delight,36.05282842,24.86799073					
16	Brewed Awakenings,36.04802733,24.8795655					
17	Frydos Autosupply n' More,36.05492902,24.90184636					
18	Albert's Fine Clothing,36.07528939,24.85630786					
19	U-Pump,36.07665029,24.85761023					
20	Ahaggo Museum,36.07713388,24.87619413					
21	Kronos Mart,36.05850979,24.90105793					
22	Desafio Golf Course,36.08955831,24.86076496					
23	Coffee Cameleon,36.05468599,24.88982134					
24	General Grocer,36.06039628,24.85648014					
25	Chostus Hotel,36.07074329,24.89517911					
26	Daily Dealz,36.06646999,24.88262915					
27	Frank's Fuel,36.07212924,24.84130258					
28						
29						

Let's breakdown every necessary steps to get this result :

- Grouping the GPS data by car id :

The screenshot shows the 'Execute' window with the following configuration:

- Current input Files selected :** gps.csv
- Current output Files selected :** gpsGrouped.csv
- Write down any additional args :** 1
- Current jar file selected :** Group.jar
- Current reverse jar file selected :** reverse1to1.jar

Buttons: Add, Remove, Jar file, Reverse Jar file, Give information.

- Sort chronologically the gpsGrouped.csv file :

The screenshot shows the 'Execute' window with the following configuration:

- Current input Files selected :** gpsGrouped.csv
- Current output Files selected :** gpsGroupedSorted.csv
- Write down any additional args :** 1 0 M/dd/yyyy HH:mm
- Current jar file selected :** sortGroupChronologically.jar
- Current reverse jar file selected :** reverse1to1.jar

Buttons: Add, Remove, Jar file, Reverse Jar file, Give information.

- Get the 10 minutes journey out of the gpsGroupedSorted.csv :

The screenshot shows the 'Execute' window with the following configuration:

- Current input Files selected :** gpsGroupedSorted.csv
- Current output Files selected :** journey10minutes.csv
- Write down any additional args :** 600000
- Current jar file selected :** getJourney.jar
- Current reverse jar file selected :** reverseWithIndication.jar

Buttons: Add, Remove, Jar file, Reverse Jar file, Give information.

- We now need to process the cc_data.csv file, but first we have to change the name of the people by their corresponding car id :

- We need to group and sort the resulting file just like we did for the gps.csv file.
- We can now deduces the places location with the journey file and the credit card data grouped and sorted :

arguments : 3 0 0 1 2 5 6 M/d/yyyy HH:mm MM/dd/yyyy HH:mm:ss
C:\\Users\\Rifhice\\Documents\\Rapport de stage\\VAST 2014\\CSV
files\\Output\\Output2\\deduction.csv

- We then need to group the deduction.

- Finally we need to regroup all the deduction :

The screenshot shows a software interface with several sections:

- Current input Files selected :** A text box containing "deductionGrouped.csv". Above it are "Add" and "Remove" buttons.
- Current output Files selected :** A text box containing "deductionGroupedRegrouped.csv". Above it are "Add" and "Remove" buttons.
- Write down any additional args :** An empty text box.
- Give information** button.
- Jar file** section: A button labeled "Jar file" above a text box containing "regroupDeduction.jar".
- Reverse Jar file** section: A button labeled "Reverse Jar file" above a text box containing "reverseWithIndication.jar".

On an other hand, we can use the reverse jar file we specified during the execution to try to traceback the information. To do so simply click on the “reverse specific execution” button, and choose the execution you want to reverse. For the exemple I reverse the last execution giving me this file :

The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1	Initial lines, resulting lines				
2	0/145,1				
3	145/172,2				
4	172/212,3				
5	212/250,4				
6	250/270,5				
7	270/327,6				
8	327/371,7				
9	371/402,8				
10	402/540,9				
11	540/599,10				
12	599/625,11				
13	625/643,12				
14	643/651,13				
15	651/664,14				
16	664/799,15				
17	799/826,16				
18	826/833,17				
19	833/870,18				
20	870/872,19				
21	872/878,20				
22	878/879,21				
23	879/887,22				
24	887/892,23				
25	892/900,24				
26	900/903,25				
27	903/904,26				

With this I easily know from witch line the deduction came back and even go further back.