Software Development and Integration - Lab 1

The deadline for handing in this lab is **November 17, 2024 at 23:59 CET.**

Introduction

The objective of this lab is to evaluate your knowledge of git as a version control system and the use of GitHub as a repository for your code, as you work through some programming challenges in Java involving the different structured data formats seen in class.

- This lab will be performed in groups of 4 to 5 people.
- Results must be uploaded to a specific Github repository associated to your group, hosted in Github classroom (more details below).
- As mentioned, the final result is just as important as your management of git throughout the project.
- Results must also be uploaded as a *.zip* file (including the *.git* folder) onto the Canvas platform before the deadline. **Your results will not be evaluated if they are not uploaded to Canvas, regardless of them being present in the Github repository previously mentioned.**
- You should try to showcase as much as possible your knowledge of *git* throughout the project. This includes **branch management, merges (and conflict resolution), pull requests, tags, ability to restore lost files, git flow, etc.**
- Please read the entire document before starting to write code, so as to avoid any surprises.
- All authors of this project are required to be present in the commit history of the repository in equal weight (not only in the number of commits but in the contents of the commits). **The final grade of the entire team may be heavily influenced by this.**

Requirements

Given a CSV file with data related to the number of tourists at national level grouped by total, province of origin, communities and destination in monthly periods, you are asked to create a small Java console application that allows reading the content of the file and work with it using Java objects. You are required to handle the information and display it on the console, whether it is integrated into your development environment such as VSCode or IntelliJ IDEA, or the operating system's terminal.

Outline:

1. Clone the repository available at https://classroom.github.com/a/QsNp6epm. This repository contains a ZIP file that includes the CSV file that you will be working with. Make sure to add any relevant information to the *README.md* file included.

2. You should implement the Java classes that will allow you store the information stored in the CSV and JSON files that will be created. They should have all the required attributes and methods to interact with the imported data.

3. Iterate through the CSV file and create the corresponding objects from the Java classes mentioned above to hold the data in each row. These objects will need to be stored in a data structure such as a list or an array, allowing you to iterate through them. Use these objects along with the *gson* library to then create a JSON file that represents the same information as the CSV file.

4. In subsequent steps, you are then asked to read the JSON file you created back into memory and using the same Java classes and storing the objects in a new list or array.

5. Complete the rest of the requirements outlined below and create a console application.

6. **Note:** In the cloned repository you will find a file called *Utils.java*, which contains a method that will help you transform the Total column value into a valid range so you can properly parse it.

Creating a console application

Create a console application that shows a menu in which the user is prompted to choose from 5 options.

1. **Option 1:** *"Transform CSV file to JSON".* The CSV file in the repository will be read and converted to JSON format. **This JSON file is the one you will be working with in options 2 - 4.** Transforming the CSV file into JSON must be done programmatically using the Java programming language. You may use external libraries to assist you with this. The CSV file is called **TurismoComunidades.csv**. The corresponding JSON you will create will be named **TurismoComunidades.json.**

    - All JSON objects generated will need to have an additional *key* named **"_id"**. The *value* of that key will be a unique value. You must use the **UUID** java library to generate the unique id for each element.

2. **Option 2**: *"Group by 'comunidad de destino'".* When the user selects this option, the JSON file created in the previous step will be read and a new one will be generated in which the elements are grouped by the *Comunidad de destino* field. This means that the new JSON file will be an object in which the *keys* are the different *Comunidad de destino* fields and the corresponding *values* are an array of all objects in the file containing that value in the field *Comunidad de destino*. The file should be saved to disk under the name **ComunidadesAgrupadas.json**. (Read further below for an example of what this looks like). It is recommended you use the HashMap data structure.

3. **Option 3:** *“Results by range”.* Introducing a data in standard format (yyyy-mm-dd), get the period that date references and show all objects referencing that period, along with the total number of results shown.

4. **Option 4:** *“Results by origin”.* When this option is chosen, the user will be asked to provide a province as a parameter. You need to show all objects referencing that province as the origin where the total number of tourists is greater than 10,000. If there are no valid results, show a message indicating that.

5. **Option 5:** *“Exit”.*

Please do not forget organize the project in such a way that you all collaborate together, applying the different concepts we discussed in class, such as creating the appropriate branches and creating (and possibly reviewing) pull requests. This is just as important as the final solution.

Tips

You are encouraged to use Maven to manage the dependencies in your project.

The **gson** library required for the management of JSON objects can be found here (version 2.11.1): https://mvnrepository.com/artifact/com.google.code.gson/gson

This chapter may be helpful in reading the input from the user of your application: https://books.trinket.io/thinkjava2/chapter3.html

---

Data schema

Please find below a table with the different fields present in the CSV file provided:

Name	Data Type	Description
Provincia de origen	String	Constant value “Total Nacional”, which means the total number of tourists at a national level.
Provincia de origen	String / null	Province of origin from which the tourist began their journey. Can be null.
CCAA y provincia de destino	String	Constant value “Total Nacional”, which means the total number of tourists.
CCAA y provincia de destino	String / null	Destination autonomous community
CCAA y provincia de destino	String / null	Destination province

Name	Data Type	Description
Concepto turístico	String	Constant value “Total Nacional”, which indicates the tourist concept
Period	String	Period to which the total count of tourists refers to at a given month and year
Flag	Number	Total tourist count for the given period and origin and destination.

Empty values should be treated as *null*. Your application should treat them as empty values.

Here are some examples on how to interpret the data:

Total Nacional		Total Nacional			Turistas	2023M09	15,605,926

The row above indicates that the national total number of tourists in the month of September 2023 increased to 15,605,926. Since the columns *Provincia de origen, CCAA and Provincia de destino* are not present, it is referencing total values in the country for the given period.

Total Nacional		Total Nacional	Andalucía	Huelva	Turistas	2021M08	670,005

The row above indicates that the national total number of tourists in the month of August 2021 who visited the Huelva province in Andalucía was 670,005.

Total Nacional		Total Nacional	Aragón		Turistas	2020M10	491,880

The row above indicates that the national total number of tourists in the month of October 2020 who visited Aragón, without specifying any province or origin, was 491,880.

Total Nacional	Pontevedra	Total Nacional	Castilla y León	Segovia	Turistas	2021M11	760

The row above indicates that the total number of tourists who went from Pontevedra to Segovia, in Castilla y León in the month of November 2021 was 760.

Created JSON file

Please ensure that the JSON file contains at least the following fields (names can change). If you feel the need to add more fields, you are welcome to do so. Please justify in your report. Java variable names must adhere to standards.


```
{
    "_id": "5f7f1b3b-1b7b-4b3b-8b3b-1b7b4b3b8b3b",
    "from": {
        "community": "Total Nacional",
        "province": "Sevilla"
    },
    "to": {
        "community": "Madrid",
        "province": "Madrid"
    },
    "timeRange": {
        "from": "2022-06-01",
        "to": "2022-06-30",
        "period": "2022M06"
    },
    "total": 1000
}
```


Report

You must write a report in English in which you will outline all the steps followed to complete this project. **The report must be properly written and structured.** Maximum 8 pages of content (not a strict requirement). Do not include code in the report unless you find it necessary.

The report should elaborate on how you divided the project and collaborated together using git. Please include images regarding the management of the repository (for example, pull requests and reviews), or any relevant / interesting git commands. It should also explain how you arrived to your solution.

Deliverables

Your deliverables should be contained in a *.zip* file and uploaded onto the Canvas platform:

- Project folder, **including the *.git* folder.**
- Java application with all your code.
- JSON files generated by running the application.
- Report in ***pdf*** format.
- Please do not spend time trying to game the evaluation criteria below. You should know what is truly important based on what has been discussed in class. Focus on ensuring you properly learn git and are able to write good code.

The filename should be *group_[X]_Lab1.*

The deadline for handing in this lab is **November 17, 2024 at 23:59 CET**.

Evaluation criteria

Git: 45%. Code: 35%. Report 20%.

Git

	0 points			**10 points**
Correct use of git	No version control	Bare minimum of commits and other git commands	Proper commit history and showcase of git use through branches.	All git commands seen in class properly used. Pull requests and branches used extensively.
Repository structure (README.md, .gitignore, pom.xml)	Default repository configuration	Correct repository structure.		Complete repository structure.
Problem solving	No version control	Struggles with solving challenges presented using git	Confidently knows how to navigate through day-to-day challenges present in writing code.	Exhibits complete mastery of git.

Java

	0 points			**10 points**
Follows instructions	Files are not read or written. Console application does not show menu. Java objects are not created. Data structures are not used.	Able to read files but not write them. Java classes are deficient. Significant errors in managing or displaying data.	All the requirements of the project are met, but not using the specified libraries. Code shows structural deficiencies.	All requirements are met and instructions followed. The team clearly read the document and wrote code accordingly. The code is nicely structured and easy to follow.
Shows proper use of tools taught in class	Does not set up a project following standard conventions.	Acceptable project structure. Does not use maven but is able to properly navigate through the requirements. May be using libraries that were not mentioned.		Uses all libraries and methods explained in class to handle data.

Report

	0 points			10 points
Proper use of English	Writes the report in Spanish.	Difficult to understand expression and reasoning contained in the report.	Nicely written report. Small grammatical mistakes.	Mastery of English.
Content	Lack of structure and does not include the minimum required content.	Writes a basic report that meets the requirements. May have deficiencies in explaining the reasoning behind the approaches taken.		Perfectly structured report that is a pleasure to read and to the point. Includes images.

Appendix 1 - JSON format (EXAMPLE / DUMMY DATA)

Original


```
[
  {
    "_id": "ef3d9436-0501-49ff-9d7c-7c55801712e3",
    "from": {
      "community": "Total Nacional",
      "province": ""
    },
    "to": {
      "community": "",
      "province": ""
    },
    "timeRange": {
      "from": "2024-08-01",
      "to": "2024-08-31",
      "period": "2024M08"
    },
    "total": 19906000
  },
  {
    "_id": "54b7bef9-57ae-4a72-b696-a7847a772e02",
    "from": {
      "community": "Total Nacional",
      "province": ""
    },
    "to": {
      "community": "",
      "province": ""
    },
    "timeRange": {
      "from": "2024-07-01",
      "to": "2024-07-31",
      "period": "2024M07"
    },
    "total": 18338774
  },
  {
    "_id": "432ded49-e629-4309-8976-5f2c8f85bc8c",
    "from": {
      "community": "Total Nacional",
      "province": ""
    },
    "to": {
      "community": "",
      "province": ""
    },
    "timeRange": {
      "from": "2024-06-01",
      "to": "2024-06-30",
      "period": "2024M06"
    },
    "total": 15437035
  }
]
```


Grouped


```
{
  "Andalucía": [
    {
      "_id": "84a3a8d0-fd7a-47d7-9aed-a19019f92ce8",
      "from": {
        "comunidad": "Total Nacional",
        "provincia": ""
      },
      "to": {
        "comunidad": "Andalucía",
        "provincia": ""
      },
      "timeRange": {
        "from": "2024-08-01",
        "to": "2024-08-31",
        "period": "2024M08"
      },
      "total": 4075981
    },
    {
      "_id": "313b038c-199f-4b33-99ed-07dc2e4efa6a",
      "from": {
        "comunidad": "Total Nacional",
        "provincia": ""
      },
      "to": {
        "comunidad": "Andalucía",
        "provincia": ""
      },
      "timeRange": {
        "from": "2024-06-01",
        "to": "2024-06-30",
        "period": "2024M06"
      },
      "total": 2930509
    }
  ]
  // more below...
}
```