Documentation Assignment 2

1 Introduction

This is the documentation of my implementation of the Assignment 2 for Mariner Innovations. I explain which libraries I have used when implementing it, why I choose those libraries, and how I have developed the requested features.

2 Libraries

I have imported and managed the libraries of the project using **Maven**. When selecting libraries for implementing requested features, I had in mind that those libraries should be open source and free so it can be imported easily though Maven. A secondary criterion I have applied is it should have a strong community managing or developing the library, or that it should be easy to use and any problem that can arise should be easy to solve with a quick internet search.

2.1 Library for parsing CSV files:

For parsing CSV files, I have used the library **Apache Commons CSV**. I choose this library since it is an open source and free library, with a great community developing new features and solving problems that people using it may have.

2.2 Library for parsing JSON files:

For parsing JSON files, I have used the library **json-simple**. I choose this library since it is an open source and free library, and it is easy to use, any problem I had using it was easily solver with a quick internet search.

2.3 Library for parsing XML files:

For parsing XML files, I have used the API **DOM**, included with java. I choose this API since it is already included with java, and it use is recommended by the World Wide Web Consortium (W3C), which has a great community behind. The fact that it does not have to be imported simplifies its use.

3 Features

3.1 Merging CSV, JSON and XML files into a single CSV file

When merging the three files into a single CSV file, I have parsed each document separately and appended each record to a new CSV file. First, I have parsed the CSV file and inserted each record into a new CSV file, then I parsed the JSON file and appended each record into the CSV file previously created, and finally, I parsed the XML file and appended each record as well.

3.2 PRESERVING ORDER OF COLUMNS IN THE ORIGINAL CSV FILE

For preserving the order of the records of the columns that was in the original CSV file, I have stored the **header** (first row containing the names of each column) and, when parsing the XML and JSON files, I accessed the data in the order indicated by the columns in the header, then I created a new record using the data in that order, and I have appended each record into the CSV file.

3.3 DELETE RECORDS WITH "PACKETS-SERVICED" VALUE EQUAL TO 0

When processing each record, I checked if the "packets-serviced" value was equal to zero. In that case, those records were **ignored** and not appended to the new CSV file.

3.4 DATE ISSUE

When parsing the JSON file, I noticed that the date was expressed in milliseconds and not in "yyyy-MM-dd hh:MM:ss ADT" format, so I transformed those milliseconds into the requested **format** in the requested **time zone**.

3.5 RECORDS SORTED BY "REQUEST-TIME" IN ASCENDING ORDER

Since the date format was "yyyy-MM-dd hh:MM:ss ADT", I simply parsed all records already combined and filtered in the output CSV, I sorted it by "request-time" in **alphabetical order** (this date format has the characteristic that if your sort it in alphabetical order, your are sorting it by time), and I reinserted every record into the CSV, now sorted.

3.6 SUMMARY WITH NUMBER OF RECORDS PER "SERVICE-GUID"

For this task, I simply parsed the CSV inserting all "serviced-guids" into a TreeMap, using the "service-guid" as key of the **TreeMap**, and the number of records as value. If a "service-guid" has not been inserted in the map, it inserts it into the map as key, and put the value "1" as value. If a "serviced-guid" is already in the map, it updates the value adding "1".

4 RUNNING THE PROGRAM

The program has been packaged using Maven into a **"jar"** file that includes the dependencies. The input files ("csv", "json" and "xml") must be in the same location of the "jar" file, and the "jar" file must be executed (double click on it if you are using Windows). Then the program will be executed, and an output directory named "output" will be created containing two "csv" files.

The file **"merged-and-ordered.csv**" will contain the merged records from the "csv", "json" and "xml" files, sorted by "request-time" and with the records which contained a 0 value in the column "packets-serviced" deleted.

The file **"service-guids.csv"** will contain two columns, the first one containing each different "service-guid", and the second one containing the number of occurrences of each "service-guid" in the first column.