Mariner Innovations - Data Merge Documentation

Date: 8/27/2020 - Neervan Anooroop

➢ Problem Description

3 input files have been provided: reports.csv, reports.json, & reports.xml.

Write a JAVA application to read the data of these files and merge them into one single csv file.

The output file should follow the following guidelines:

- The same column order and formatting as reports.csv
- All report records with packets-serviced equal to zero should be excluded
- records should be sorted by request-time in ascending order

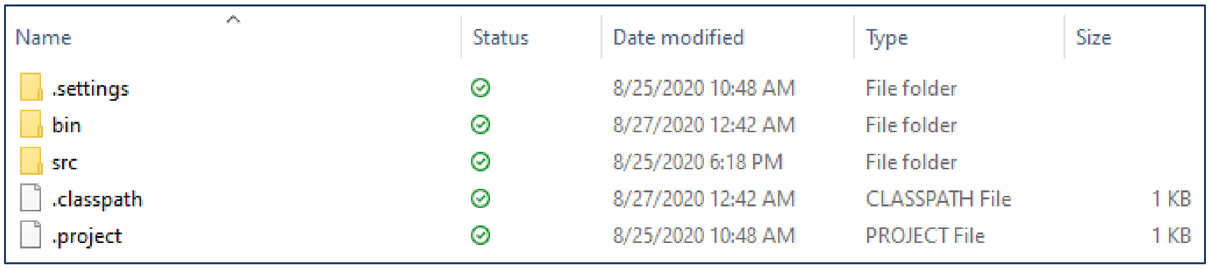
Additionally, the application should print a summary showing the number of records in the output file associated with each service-guid.

➢ Solution Description (Logic behind the code)

1. Create a class called 'Report' with the following attributes:
    a. client-address
    b. client-guid
    c. request-time
    d. service-guid
    e. retries-request
    f. packets-requested
    g. packets-serviced
    h. max-hole-size
2. Create another class called 'DataMerge' which contains the main method.
3. Read data from the three report files and create an object of class 'Report' for each record.
4. Create an Array List of type 'Report' that will hold the objects created in step 3.
5. Filter the Array List by excluding the objects that has 'packets-serviced' = 0.
6. Sort the Array List in ascending order using the 'required-time' attribute.
7. Write the content of the Array List in a new file called 'combinedReport.csv'.
8. Create a Map Interface that has 'service-guid' as the key and number of records related with 'service-guid' as the value. Iterate through the Map Interface and print the key alongside its value to get the summary.

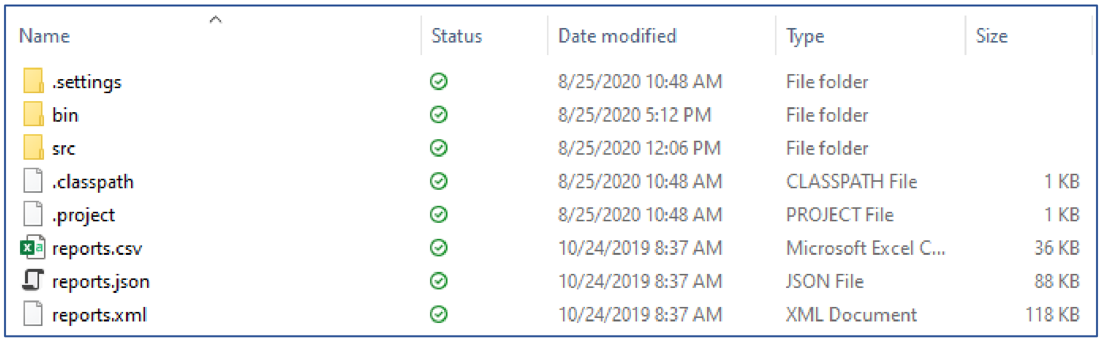
➢ Implementation & Testing

1. Create a JAVA Project



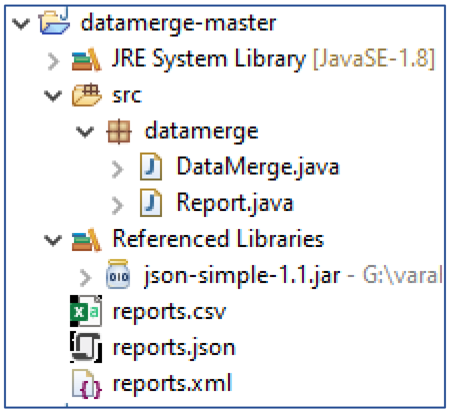
Name	Status	Date modified	Type	Size
.settings	✓	8/25/2020 10:48 AM	File folder	
bin	✓	8/27/2020 12:42 AM	File folder	
src	✓	8/25/2020 6:18 PM	File folder	
.classpath	✓	8/27/2020 12:42 AM	CLASSPATH File	1 KB
.project	✓	8/25/2020 10:48 AM	PROJECT File	1 KB

2. Move the report files to the project folder:



Name	Status	Date modified	Type	Size
.settings	✓	8/25/2020 10:48 AM	File folder	
bin	✓	8/25/2020 5:12 PM	File folder	
src	✓	8/25/2020 12:06 PM	File folder	
.classpath	✓	8/25/2020 10:48 AM	CLASSPATH File	1 KB
.project	✓	8/25/2020 10:48 AM	PROJECT File	1 KB
reports.csv	✓	10/24/2019 8:37 AM	Microsoft Excel C...	36 KB
reports.json	✓	10/24/2019 8:37 AM	JSON File	88 KB
reports.xml	✓	10/24/2019 8:37 AM	XML Document	118 KB

3. Create the following folder structure:

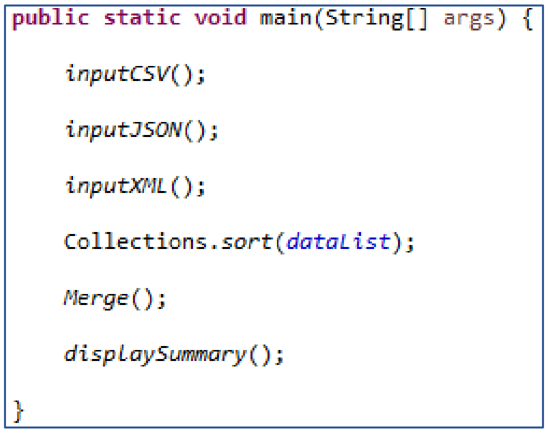


```
datamerge-master
  JRE System Library [JavaSE-1.8]
  src
    datamerge
      DataMerge.java
      Report.java
  Referenced Libraries
    json-simple-1.1.jar - G:\varal
  reports.csv
  reports.json
  reports.xml
```

a. Create a package named ‘datamerge’
b. Create 2 java classes inside the packege:
   i. DataMerge.java
   ii. Report.java
c. Add the External Jar ‘json-simple-1.1.jar’ (Click to download)

[Right Click on Project → Build Path → Configure Build Path.. → Add External JARS… → Select jar file]

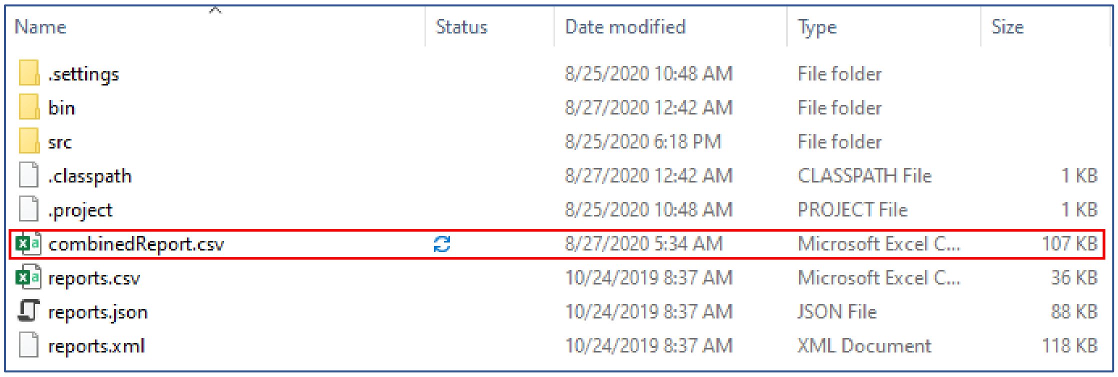
4. **Run the main method of 'DataMerge.java':**



```
public static void main(String[] args) {

    inputCSV();

    inputJSON();

    inputXML();

    Collections.sort(dataList);

    Merge();

    displaySummary();

}
```

5. **After the program is run, a new file namely 'combinedReport.csv' is created in the projects folder.**

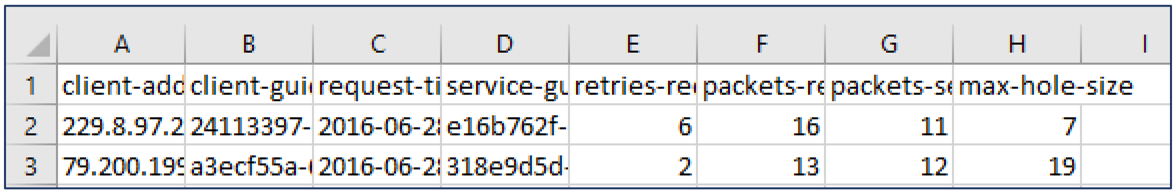
    a. This is the combined CSV file of reports.csv, reports.json, & reports.xml



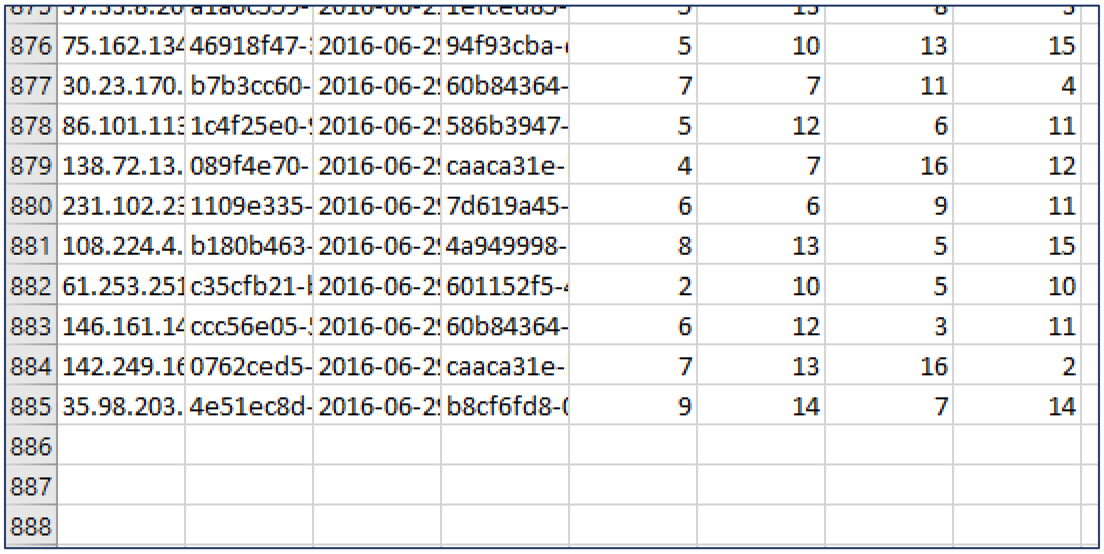
Name	Status	Date modified	Type	Size
.settings		8/25/2020 10:48 AM	File folder	
bin		8/27/2020 12:42 AM	File folder	
src		8/25/2020 6:18 PM	File folder	
.classpath		8/27/2020 12:42 AM	CLASSPATH File	1 KB
.project		8/25/2020 10:48 AM	PROJECT File	1 KB
combinedReport.csv		8/27/2020 5:34 AM	Microsoft Excel C...	107 KB
reports.csv		10/24/2019 8:37 AM	Microsoft Excel C...	36 KB
reports.json		10/24/2019 8:37 AM	JSON File	88 KB
reports.xml		10/24/2019 8:37 AM	XML Document	118 KB

6. **If we open the 'combinedReport.csv' file, we may see that we have <u>885 records including the headers</u> and:**

    a. the column order and formatting are similar to 'report.csv'
    b. there are no 'packets-serviced' that is equal to zero
    c. the records are sorted by request-time in ascending order

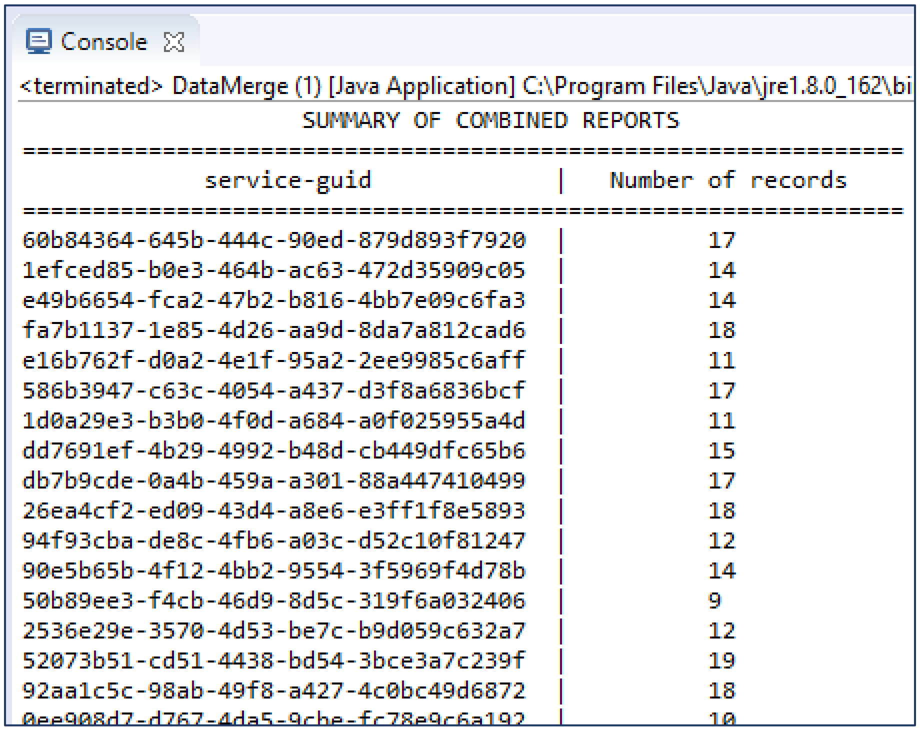


	A	B	C	D	E	F	G	H	I
1	client-add	client-gui	request-ti	service-gu	retries-re	packets-re	packets-se	max-hole-size	
2	229.8.97.2	24113397-	2016-06-2	e16b762f-	6	16	11	7	
3	79.200.19	a3ecf55a-	2016-06-2	318e9d5d-	2	13	12	19	



876	75.162.134	46918f47-	2016-06-2!	94f93cba-	5	10	13	15
877	30.23.170.	b7b3cc60-	2016-06-2!	60b84364-	7	7	11	4
878	86.101.11	1c4f25e0-	2016-06-2!	586b3947-	5	12	6	11
879	138.72.13.	089f4e70-	2016-06-2!	caaca31e-	4	7	16	12
880	231.102.2	1109e335-	2016-06-2!	7d619a45-	6	6	9	11
881	108.224.4.	b180b463-	2016-06-2!	4a949998-	8	13	5	15
882	61.253.251	c35cfb21-	2016-06-2!	601152f5-	2	10	5	10
883	146.161.14	ccc56e05-	2016-06-2!	60b84364-	6	12	3	11
884	142.249.16	0762ced5-	2016-06-2!	caaca31e-	7	13	16	2
885	35.98.203.	4e51ec8d-	2016-06-2!	b8cf6fd8-	9	14	7	14
886								
887								
888								

7. On the console, we may see an output message as follows:

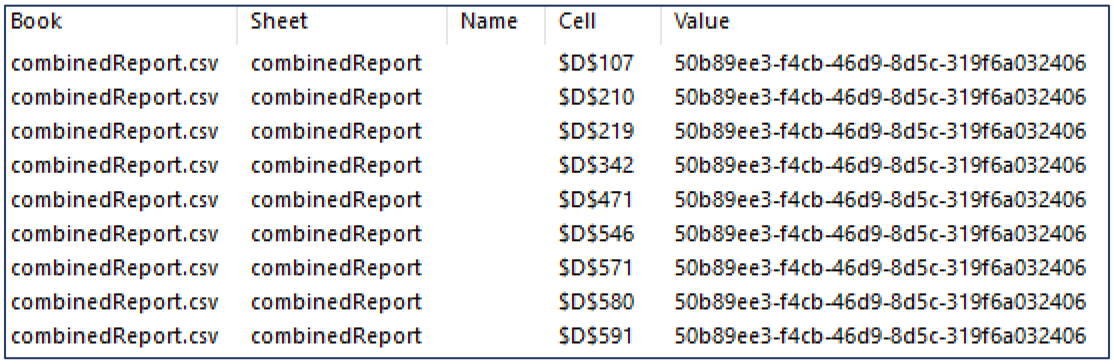


```
<terminated> DataMerge (1) [Java Application] C:\Program Files\Java\jre1.8.0_162\bi
                        SUMMARY OF COMBINED REPORTS
===============================================================
             service-guid                |   Number of records
===============================================================
60b84364-645b-444c-90ed-879d893f7920     |          17
1efced85-b0e3-464b-ac63-472d35909c05     |          14
e49b6654-fca2-47b2-b816-4bb7e09c6fa3     |          14
fa7b1137-1e85-4d26-aa9d-8da7a812cad6     |          18
e16b762f-d0a2-4e1f-95a2-2ee9985c6aff     |          11
586b3947-c63c-4054-a437-d3f8a6836bcf     |          17
1d0a29e3-b3b0-4f0d-a684-a0f025955a4d     |          11
dd7691ef-4b29-4992-b48d-cb449dfc65b6     |          15
db7b9cde-0a4b-459a-a301-88a447410499     |          17
26ea4cf2-ed09-43d4-a8e6-e3ff1f8e5893     |          18
94f93cba-de8c-4fb6-a03c-d52c10f81247     |          12
90e5b65b-4f12-4bb2-9554-3f5969f4d78b     |          14
50b89ee3-f4cb-46d9-8d5c-319f6a032406     |          9
2536e29e-3570-4d53-be7c-b9d059c632a7     |          12
52073b51-cd51-4438-bd54-3bce3a7c239f     |          19
92aa1c5c-98ab-49f8-a427-4c0bc49d6872     |          18
```

a. This is a summary which shows the number of records associated with each 'service-guid'.

b. To verify these numbers, we may copy the 'service-guid' value and do a 'CTRL + F' search for all occurrences of that 'service-guid'.

As we can see in the snapshop below, we have nine occurrences for 'service-guid' = '50b89ee3-f4cb-46d9-8d5c-319f6a032406', just like the program printed.



Book	Sheet	Name	Cell	Value
combinedReport.csv	combinedReport		D107	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D210	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D219	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D342	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D471	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D546	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D571	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D580	50b89ee3-f4cb-46d9-8d5c-319f6a032406
combinedReport.csv	combinedReport		D591	50b89ee3-f4cb-46d9-8d5c-319f6a032406

➢ Libraries

❖ JRE System Library [JavaSE-1.8] – rt.jar

rt.jar stands for runtimes and contains compiled class files for the core Java Runtime environment. rt.jar is where all the Java packages reside and it needs to be included else, we don't have access to core classes.

Below are the rt.jar packages used in this project:

1. java.io

- Read the 'reports.xml' file
- Read the 'reports.json' file
- Write to the new 'combinedReport.csv' file
- Handle Exceptions (FileNotFoundExceptions & IOException)

2. java.nio

- Get the path the 'reports.csv' file

3. java.text

- Handle Exceptions (ParseException)
- Convert String variable to Date format

4. **java.time**

   - Convert the ‘request-time’ in ‘reports.json’ from milliseconds to Date format

5. **java.util**

   - Create the List, Array List, Map, Hash Map & Collections
   - Create iterator for Array List
   - Create Date formatted variables
   - Read ‘reports.csv’ path
   - Get Time Zone for ‘request-time’ format

6. **javax.xml.parsers**

   - Parse the ‘reports.xml’ file

7. **org.w3c.dom**

   - Parse constructor of File class
   - Cast Element object
   - Get xml sub elements

❖ **External Library: json-simple-1.1.jar**

JSON.simple is a simple Java based toolkit for JSON. We can use JSON.simple to encode or decode JSON data. Since we need to decode a JSON file namely reports.json, JSON.simple proved to be a very efficient tool in doing so.

Below are the JSON.simple packages used in this project:

8. **org.json.simple**

   - Parse ‘reports.json’ into instance of JSONArray
   - Cast JSONObject object

9. **org.json.simple.parser**

   - Provide forward, read-only access to JSON data

NOTE: To have a closer look at the functionalities of the program, I recommend checking out the JAVA source files. A more in-depth documentation can be generated using Javadoc.