Tesla Energy Feed Interpreter

Overview

- This is a Maven based Java project. The artifact jar for testing in included in the java project under TeslaInterpreter-> classes -> artifacts-> TeslaInterpreter_jar-> TeslaInterpreter.jar.
- The jar is bundled with dependencies included.

Features

- Processes a file containing feed and interprets them and based on the validation criteria validates them, if valid line feed, then writes it into a corresponding paritition's output csv file in the order the input was received.
- As a performance optimization, includes a feature of batched output directory write.
- Timestamp, Partition and Input Output Files validation.
- An interface and framework to include more interpreters and parsers as a part of design.
- **Assumptions**: Spark Processing, Spring Frameworks could have been used for larger file for broader scope however, considering requirements on the assignment and ordering requirements, it was not used. It is assumed that 1 input sample file will be passed for testing at a time to the program jar. However, code can be easily update to support multiple file for enhancement.

Prerequisites

- Please Install Java version 1.8 and jdk, jre to support java 8 version on the system if not present already.
- Setup Maven for Java dependencies if required.
- IDE like Jetbrains IDEA Intellij for running the java code

Contents

Run Instructions

Extract the zip folder. It contains the source and test code folder and target folder. Target folder contains output file generated on sample, compiled files. Under TeslaInterpreter project -> classes -> artifact -> TeslaInterpreter_jar contains the jar that can be used for testing standalone.
TeslaEnergyInterpreter.class :
The main program takes 2 input arguments.

```
~/TeslaInterpreter/classes/artifacts/TeslaInterpreter_jar/TeslaInterpreter.jar
```

InputFilePath: 1st argument is input file path that contains sample lines.

Ex:

```
~/Tesla/TeslaInterpreter/target/classes/<INPUT_FILE_NAME.txt>
```

OutputFilePathDirectory: 2nd argument is output directory full path including succeeding **/**. This is where the output files will be written based on the partition.

Ex:

```
~/Tesla/TeslaInterpreter/target/classes/
```

-COMMAND

```
java -jar <ENTER_PATH_TO_ARTIFACT_JAR> <ENTER_INPUT_FILE_FULL_PATH> <ENTER_OUTPUT_DIRECTORY_FULL_PATH>
```

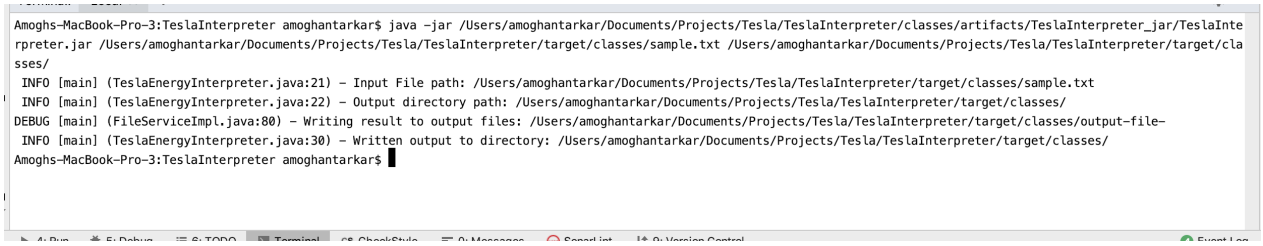
Tested Example Full Command:

```
Amoghs-MacBook-Pro-3:TeslaInterpreter amoghantarkar$ java -jar ~/TeslaInterpreter/classes/artifacts/TeslaInt
```

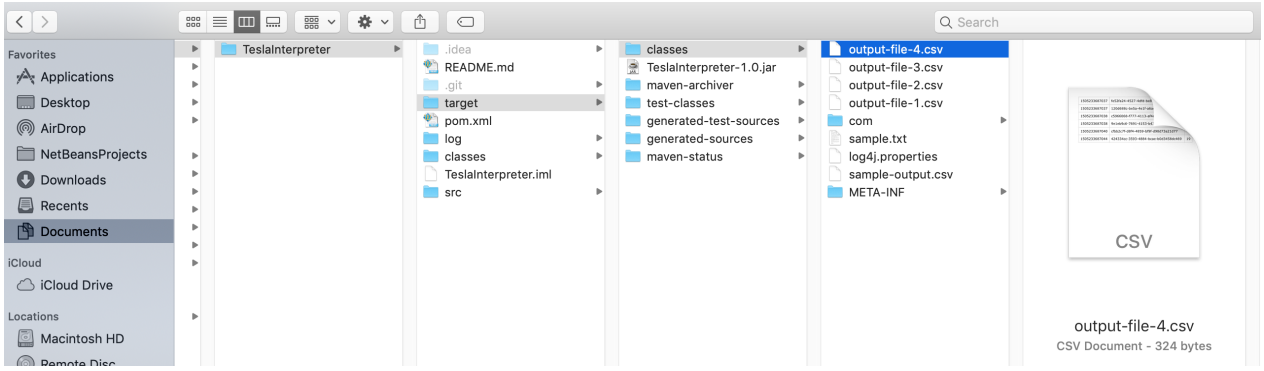
Output:

The result will be generated in the OutputFilePathDirectory directory you specified in the input argument.

The partitioned output will be generated in the *OutputFilePathDirectory with prefix* `output-file-<PARTITION>.csv`



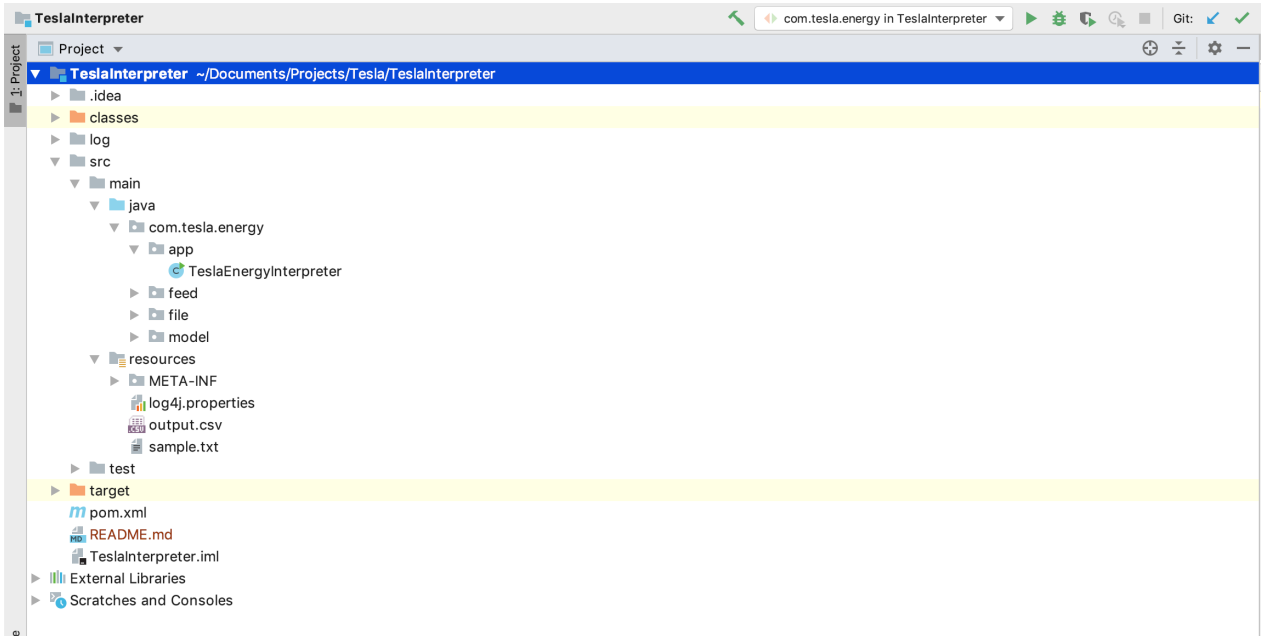
** Note on Dev Testing: **
The correct output was generated successfully in the `target` folder under `classes`.



The target folder contains: sample aggregated input and output tested and partitioned output.

Please check the `classes` was generated successfully with sum aggregation for the sample input file provided.

Project Structure and Source Code



- Includes java src, test and resources folder.
- Src: TeslaEnergyFeedInterpreter.java is the main class.
- Resources: Includes log4j.properties for configuration. Sample input output files for testing.
- Test: References resources folder for test input files. Note the output will be generated in the resources file for few tests.
- Readme - Details about
- Target folder -> classes folder includes output during testing.
- **Executable Artifact** - Includes executable jar: TeslaInterpreter.jar to be used to standalone run and testing.
  Location: `Tesla/TeslaInterpreter/classes/artifacts/TeslaInterpreter_jar/TeslaInterpreter.jar`

Only if you wish to modify the code and rebuild it follow:

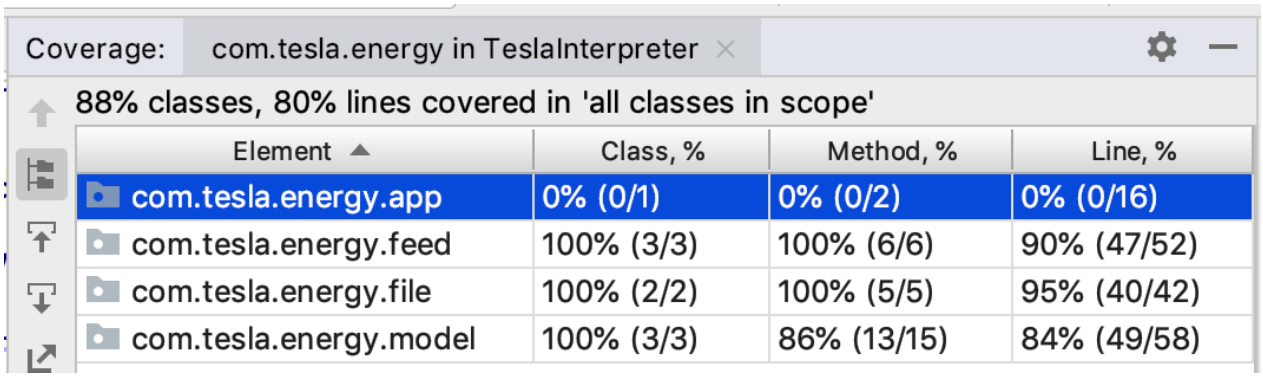
```
Amoghs-MacBook-Pro-3:TeslaInterpreter amoghantarkar$ mvn clean install

Amoghs-MacBook-Pro-3:TeslaInterpreter amoghantarkar$ mvn package

For building executable jar with dependencies: TeslaInterpreter.jar:
In Intellij, File-> ProjectStructure-> Artifacts-> Add -> Jar -> From Modules With dependencies.
In Build-> Build Artifacts -> Build.
The artifact will be generated in the classes/artifacts folder
```

Testing

- The project includes test coverage of over 80%
  Libraries used: Jupiter, JUnit etc.



Logging

- Used SLF4j logging library.
- Logs will be generated in the TeslaInterpreter-> log folder -> **log/tesla_interpreter.log**
- Default logging level is shipped with DEBUG level and not INFO due to code review and purposes.

Coding Standards

- **Google Coding Standards** with CheckStyle
- **SonarLint** for Code Analysis

Libraries

Java8, Google Guava, Jupiter, SLF4j, Log4j

References

StackOverflow, Java Docs