# **Java Developer Code Challenge V1.0 – Trade Reporting Engine**

By: Boris Sirota

Date: 30th March 2021

# Design:

## Tools & Technologies:

Eclipse – Java IDE

Spring-Boot – rapid Java application development with dependency injection

Maven – build tool and dependency management

## Solution:

**Mission Statement:**

Create loosely coupled classes that can perform specific tasks to achieve a complete solution.

The classes are configured via Spring and must be easily extendable.

The classes methods need to be simple and easily testable.

The application needs to be run on the command line and take arguments.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Type** | **Responsibilities** |
|  | **TradeReportingEngine** | class | Main Java class that instantiates Spring Context;  Takes in command line arguments that are used in processing;  Orchestrates invocation of other classes to perform required processing; |
|  | **FileParser** | interface | Provides file parsing API |
|  | **XMLFileParser** | class | Implements FileParser for XML files |
|  | **OutputCreator** | interface | Provides API for output creation |
|  | **CSVOutputCreator** | class | Implements CSV output creation |
|  | **filters** | properties | Contains XPath to CSV headers mappings |
|  | **TradeReportingEngineTests** | class | JUnit enabled class that contains tests for all the classes methods. |

# Assumptions and Trade-offs:

1. The code could be made more generic with any custom number of columns read in from the configuration file.
2. Checking for specific seller parties and currencies required those to be hardcoded. However, it is possible to pass those as parameters. It all depends on the complexity of business rules that might be very hard or impossible to implement via simple if/else conditions.
3. More unit tests can be written. Also, other unit test frameworks can be used to mock objects for example.
4. *application.properties* file can potentially be used to externalise some of the application specifics to suit various environments like dev, test or prod.

# Solution Execution:

* A minimum of Java 8 is required

Please set JAVA\_HOME as per below:

**set JAVA\_HOME=*<local java installation directory>***

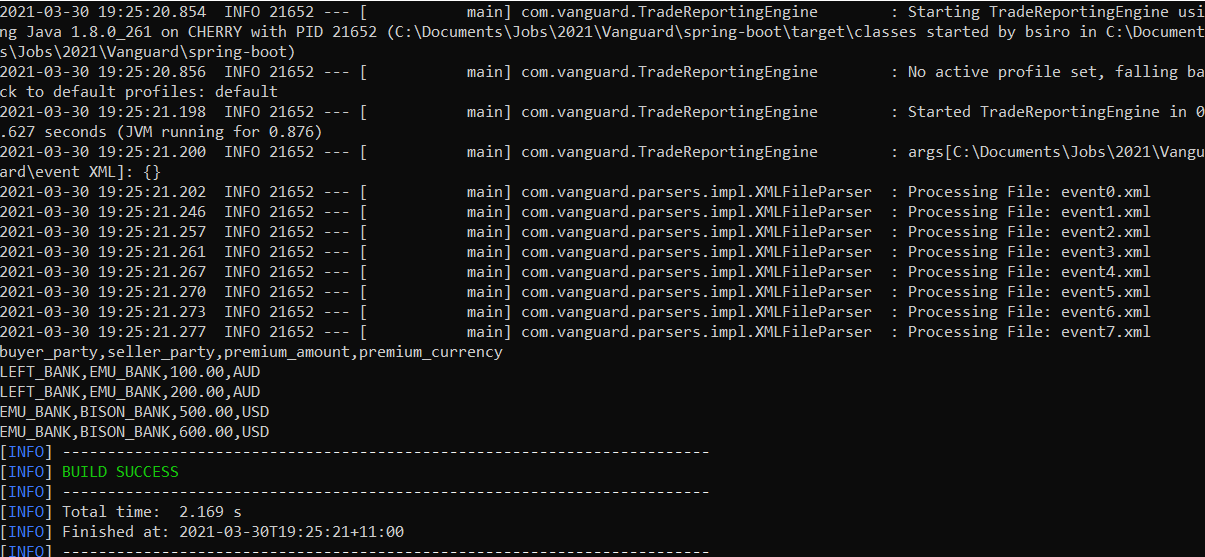
Please note: **pom.xml** containsthe default location of the data files:

|  |
| --- |
| <properties>  …  <data.path>"..\event XML"</data.path>  </properties>  …  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <configuration>  <arguments>  <argument>${data.path}</argument>  </arguments>  </configuration>  </plugin> |

Change directory to the root of the project tree:

Run: ***.\mvnw spring-boot:run***

# Example Output:



# Executing Unit Tests:

Change directory to the root of the project tree:

Run: **.\mvnw -Dtest=TradeReportingEngineTests test**

# Example Output:

