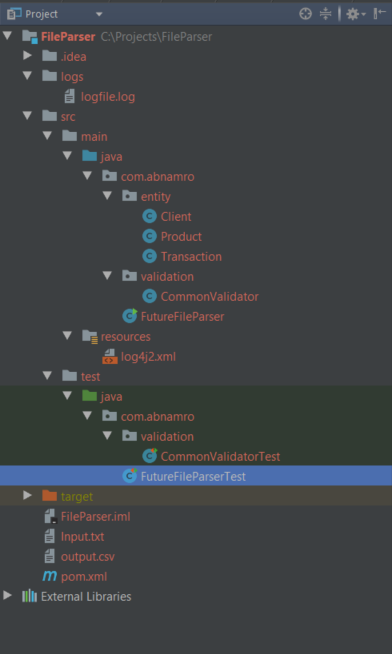
# Instruction

## 1.1 Run Environment

JDK: 1.7

Maven: 3.5.2

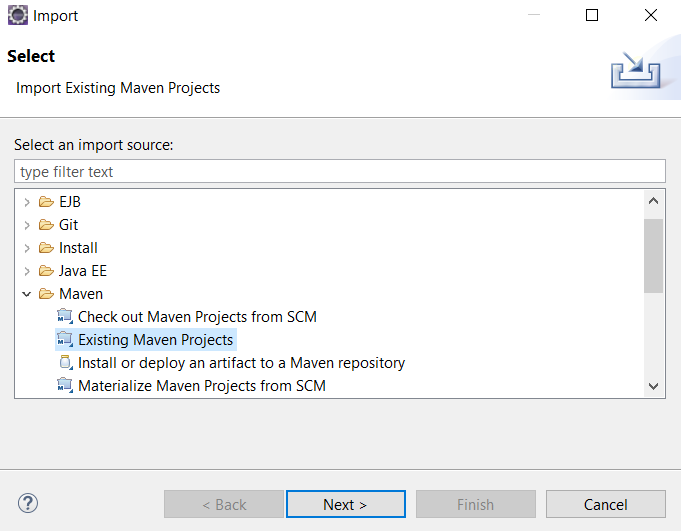
## 1.2 Project Structure



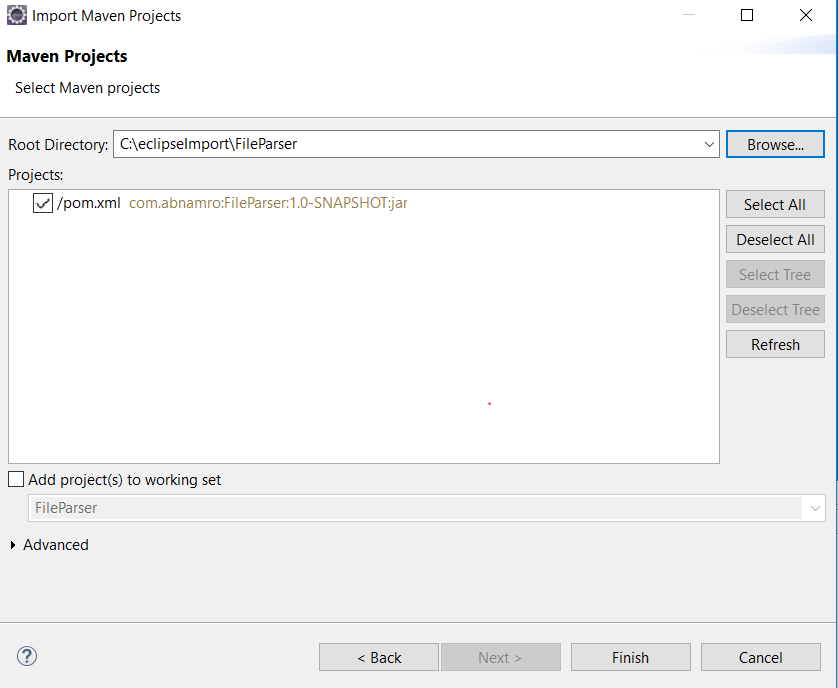
1. Logfile.log is the log for the project which in the /{project}/logs
2. All source codes in /{project}/src/main/java
3. All test code in /{project}/src/test/java
4. Log4j2.xml which in the /{project}/src/main/resources is the configuration file for the log. It can change log level, pattern and change log file name and location
5. Input.txt under the folder /{project} is default file, programmer will use that as default file if not provide file name

## 1.3 Import project into Eclipse

1. In Eclipse, File 🡪Import and select “Existing Maven Projects” as below



1. Select the location of the project and click “Finish”



## 1.4 How to Run

### 1.4.1 Run without IDE

1. Go to the location of project and run “mvn clean install”, it will clean, run test case and build the jar file
2. There is jar named “**FileParser-1.0-SNAPSHOT-spring-boot.jar**” would be generate under {project}/target
3. Run “java -jar **FileParser-1.0-SNAPSHOT-spring-boot.jar** {full file path and file name}” it would generate output.csv

### 1.4.2 Run in IDE

Right click on FutureFileParser.java and click “Run As”🡪”JAVA Application” (on Eclipse) or “Run ‘FutureFileParser.main()’”(on IntelliJ). It would pick the file Input.txt under /{project} if no parameter provide, developer can use that way to debug and do testing quickly.

## 1.5 Implementation

1. According to the requirement ” Each Record in the input file represents ONE Transaction from the client for a particular product”, I create Transaction.java to represent each record. Moreover, each Transaction object contains one Client object and one Production object. Transaction.java, Client.java and Production.java are locate in {project}/src/main./java/com/abnamro/entity
2. For Client.java, override hashCode and equals methods. So if two objects have the same client number, client type, account number and sub account number, I think they are the same client
3. For Product.java, override hashCode and equals methods, so if two objects have the product group code, exchange code, expiration date and symbol, I think they are the same product.
4. For Transaction.java, override hashCode and equals methods. So I can group transactions based on client and product combination.
5. Read the file, iterate the line by line, do the validation(Class: commonValidate) and parse each line to one Transaction object.(Class:FutureFileParse, method: parseSingleLine )
6. Group Transaction objects based on client and product combination, calculate the total amount and store in the Map(Class:FutureFileParse, method: parseFile)
7. Iterate the map and generate the CVS file (Class:FutureFileParse, method: generateCSVFile)