Report a+

Deliverable 02 – Domain Model

María Jiménez Vega

Antonio Nolé Anguita

Álvaro Calle González

Julia García Gallego

Fernando Manuel Ruiz Pliego

Contenido

[INTRODUCTION 2](#_Toc528622149)

[GSON INSTALLATION IN THE PROJECT 3](#_Toc528622150)

[CODE 4](#_Toc528622151)

[BIBLIOGRAPHY 6](#_Toc528622152)

# INTRODUCTION

In the first place, we investigate about the two options that advise us in the project statement. After researching we decided on JSON because is faster than YAML and more interoperable whit more system and it could be more useful in the future.

Secondly, we investigate the different libraries that can be used to implement what is requested. We saw that there are several, including Jackson [1], GSON [2] among others. [3]

Finally, we opted for GSON because although Jackson provided it to us spring [4], GSON seemed easier to implement. In addition, facing a different library because of Jackson already knew her.

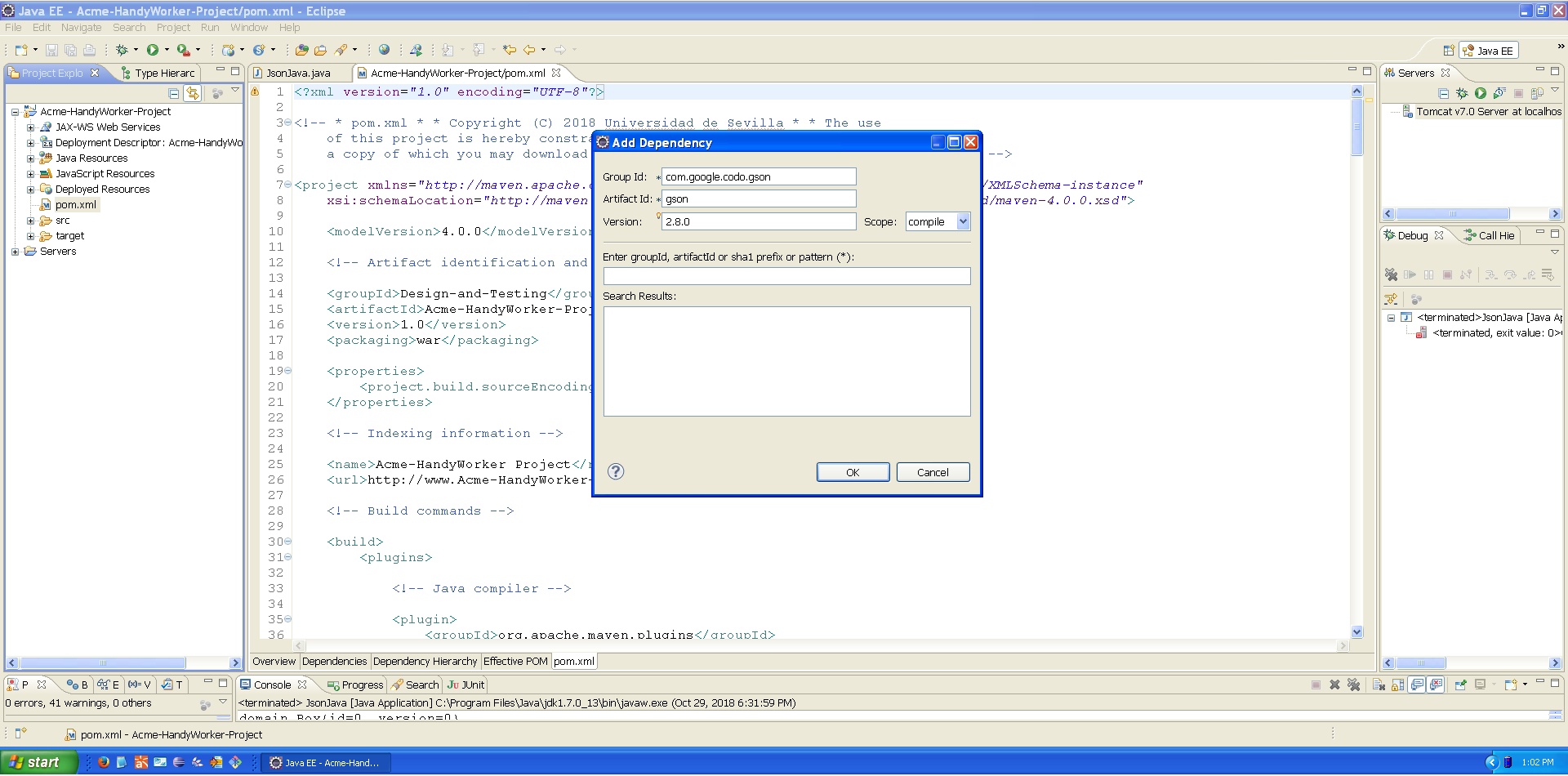
# GSON INSTALLATION IN THE PROJECT

To install GSON in the project we must open the pom.xml file [5]. We right click on the file and fill in the fields as it appears in the following picture.

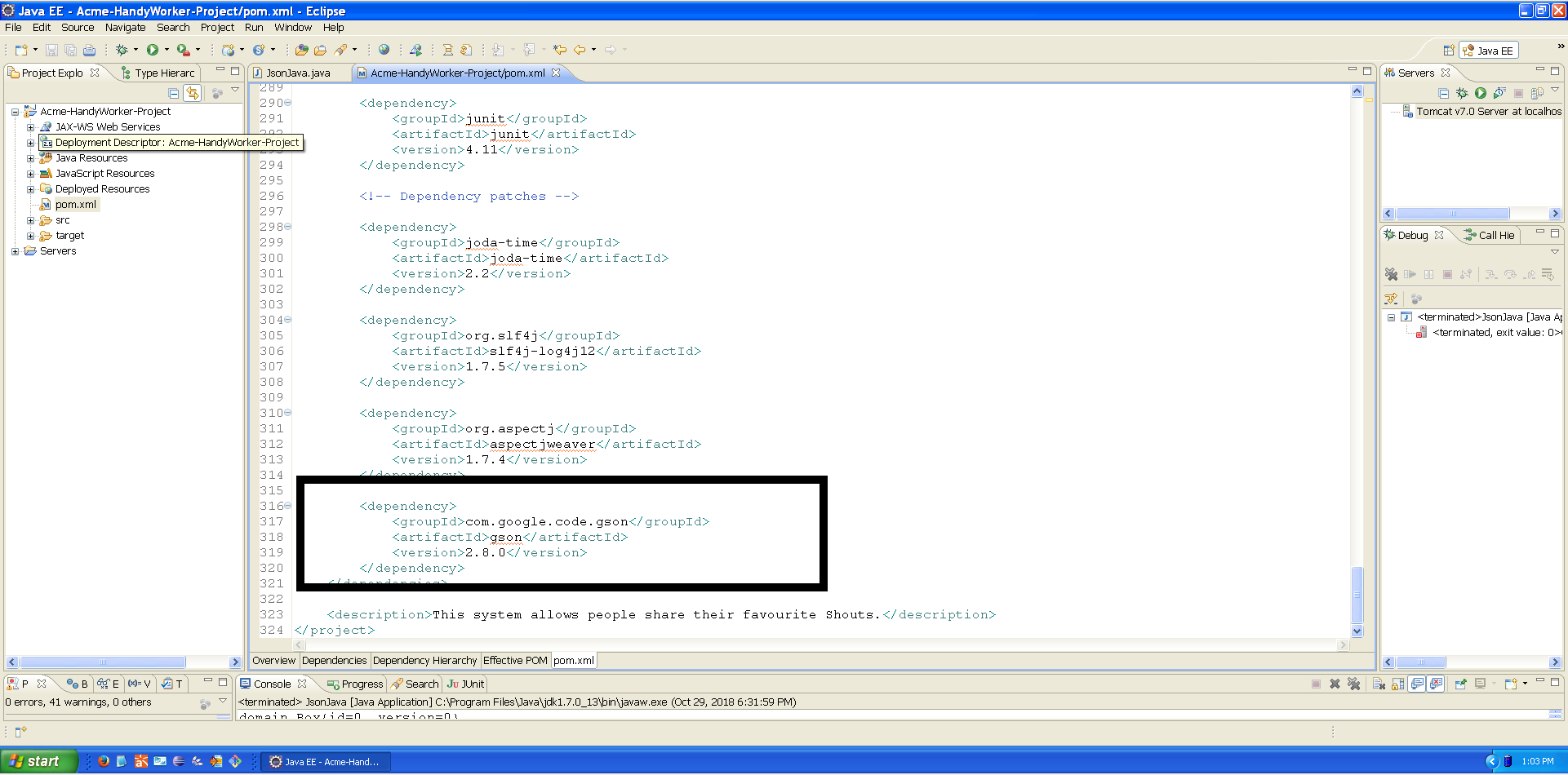
Group ID = com.google.code.gson

Artifact ID = gson

Version = 2.8.0



We click ok and the following code appears automatically:



To make sure everything is correct, we right click ok the proyect mave>update maven project and select check “focer update of snapshots/releases”.

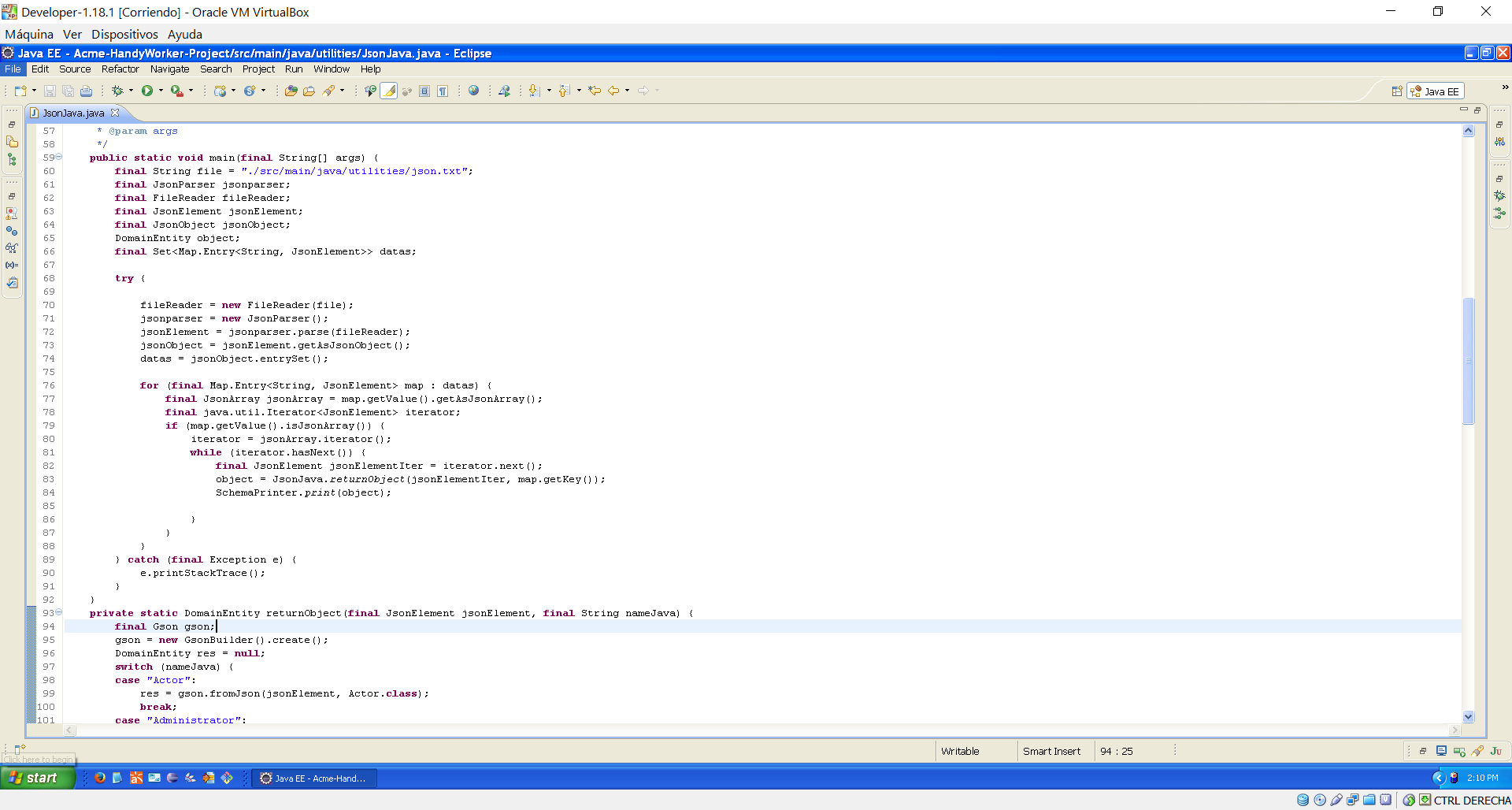
# CODE

To implement the functionality, you must create a java class and a file transformed in an object. Both are in src>main>java>utilities.

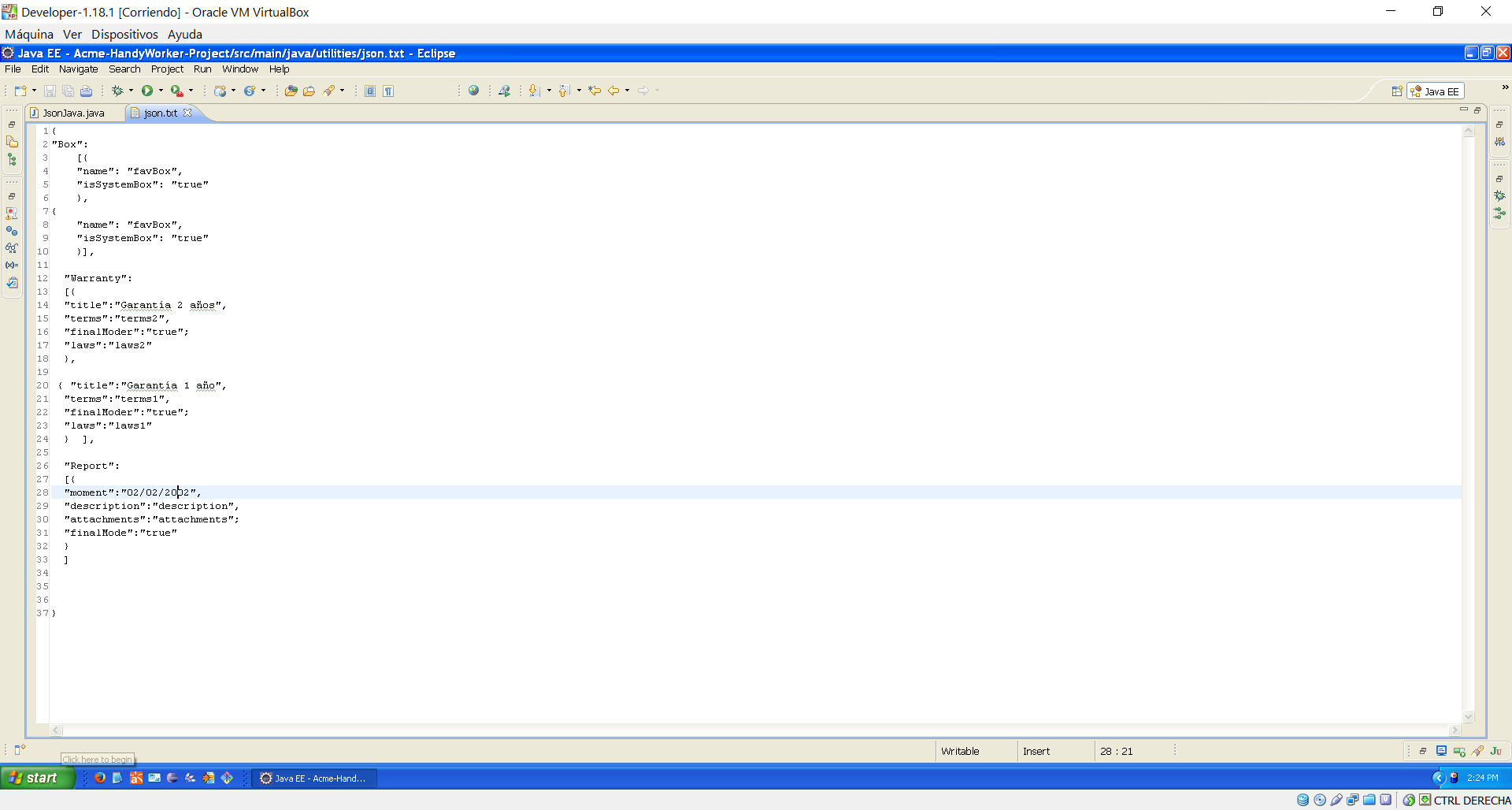
We have called the java class which implements the code that will be executed. This class is composed of a main method in which the following steps are performed within a try/catch:

1. Read json.txt file.
2. The file is passed by jsonParser so that they become data.
3. We transform it into an object.
4. The object through the method entrySet() we have a set of data.
5. Through a method we run the data which after making certain checks and making use of “Iterator” to go through the data that is within the map.
6. Once we have the piece of the JSON file that only contains what corresponded to a java entity, it passes through a private method that returns this data converted into a java object. This is done through the method offered by the GSON library “fromJson”, to which an element JSON is passed and the class to which it corresponds. To know which java class, it belongs to we do it with a switch method, by passing the name of the class in string we obtain its java class.

While we were doing tests, we verified that it does not detect well the format of the dates. To solve this problem, we have used setDateFormat to format the dates [6].

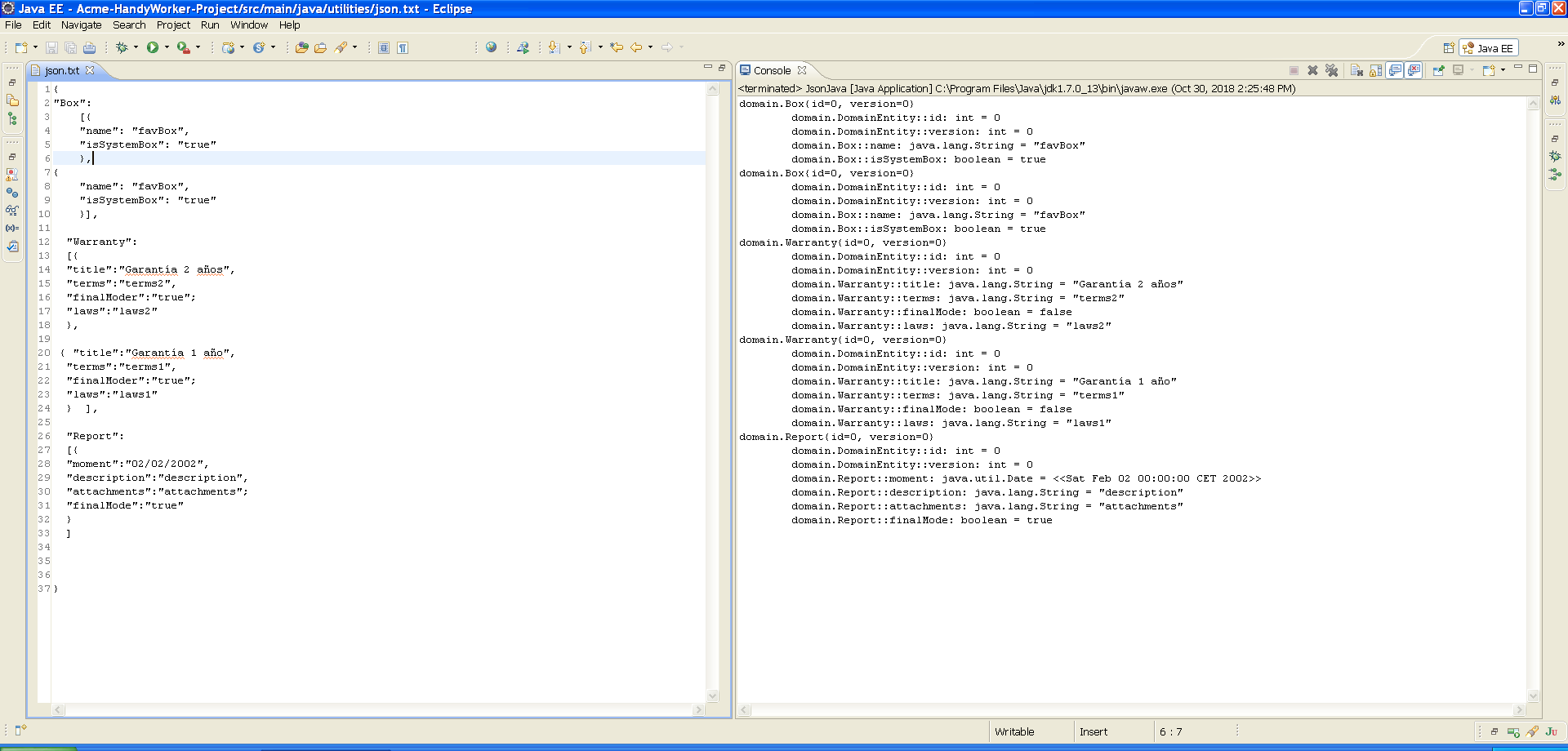


Finally, we created a file in which we created JSON objects and that we later used to test the implemented.



# TEST

Continued you can see the tests that we have done.



# BIBLIOGRAPHY

1. <https://www.adictosaltrabajo.com/2011/02/09/jackson-deserialize-with-constructor/>
2. <https://www.adictosaltrabajo.com/2012/09/17/gson-java-json/>
3. <https://www.baeldung.com/jackson-vs-gson>
4. <https://springframework.guru/processing-json-jackson/>
5. <https://stackoverflow.com/questions/26701452/gson-dependency-which-repository>
6. <https://stackoverflow.com/questions/7910734/gsonbuilder-setdateformat-for-2011-10-26t202959-0700>