Technical University of Cluj-Napoca

Faculty of Automation and Computer Science

Computer Science Department

Year 2015-2016 Semester II

Programming Techniques

*5th Project*

Student: Dan Toderici

Group: 30422

Contents

[Project specification 3](#_Toc451683199)

[Problem Analysis 3](#_Toc451683200)

[Modeling 4](#_Toc451683201)

[Adapter Pattern 4](#_Toc451683202)

[Data Access Object Pattern 5](#_Toc451683203)

[Null Object Pattern 6](#_Toc451683204)

[Design 7](#_Toc451683205)

[Relational Diagram 7](#_Toc451683206)

[Conclusion and further developments 9](#_Toc451683207)

[References 9](#_Toc451683208)

# **Project specification**

Propose is to design and implement a system that simulates a dictionary management application for processing new words or searches.

The application, that resulted from design such system, can perform 7 main operations:

1) Insertion of a new word

2) Adding a new definition for the word

3) Deletion of definitions

4) Deletion of words

5) Creating a json file for saving changes

6) Searching words

7) Display words

However, all these operations must be executed based on user input. Therefore, a graphical user interface is needed. The UI must provide the user methods to insert orders and to choose which operations he or she desires to be execute. Adaptive view was not required, and only admin will be allowed to view the dictionary info.

# Problem Analysis

Splitting the problem into smaller parts is the first main problem. Due to the need of persistence and further requirements, a package which operates on a json file is needed. Not to forget about designing a proper structure, this will help us much more than an xml or a basic file and it saves data easier! Moreover, there should be extra logic for inserting or not into the json file or not and also to check if the input is okay or not.

Operations on the database, as expected must be in conformity with the Data Access Object pattern (DAO); this will be explained later when reaching the implementation and considerations chapters.

Views, due to my previous experience regarding Layered Architecture and Model View Controller designs, will follow the same basic rule. Views are the dumbest, but the most used. They do one thing only but perfectly, which is to show the user information and provide resources for him to pass his or her desired command.

Model, this time is better structure and is not only a recipient for the data in the file and does more complex stuff such as calculating frequency of search and accuracy.

I might consider Controllers my bread and butter, therefore this project cannot have weak linking and classes have high cohesion. Data must come from the json file and be placed in the model classes, from there the Controllers provide them to views and last, but not least views make connection with the user.

# Modeling

Design patterns are the most important part of this project and I will insist on those. Finding patterns to fit your project is not an easy job and taking into account that this project is also difficult to design in an OOP manner, due to the lack of complexity. However after hours of study I came up with 3 possible patterns which I consider to be a good match.

## Adapter Pattern

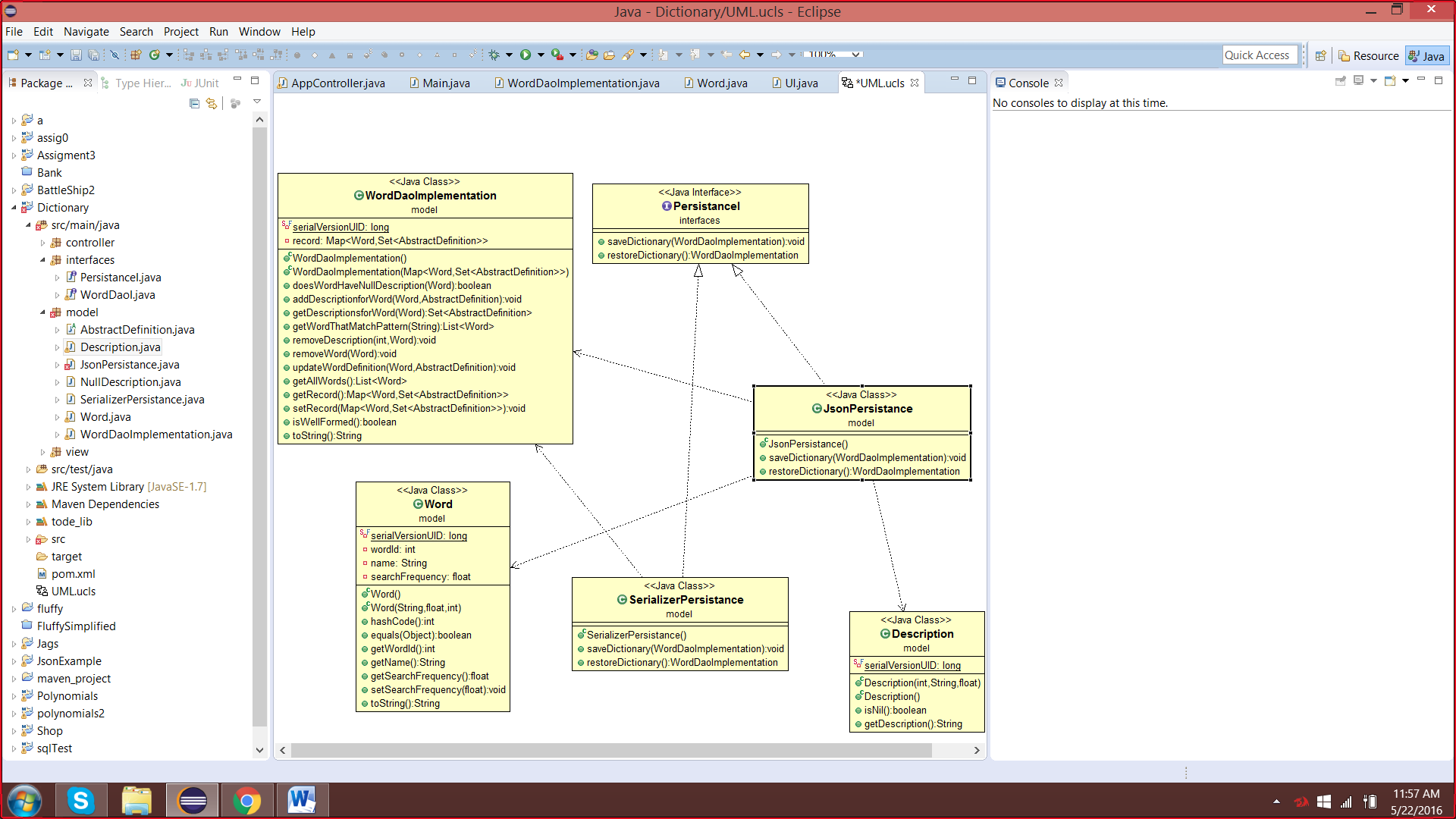
Adapter pattern works as a bridge between two incompatible interfaces. This type of design pattern comes under structural pattern as this pattern combines the capability of two independent interfaces.

This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces. A real life example could be a case of card reader which acts as an adapter between memory card and a laptop. You plug in the memory card into card reader and card reader into the laptop so that memory card can be read via laptop.

We are going to implement this pattern in our project in which persistence can be done through a json or a .ser file.

We have a PersistanceI interface and 2 implementations JsonPersistance and SerialiazerPersistance.

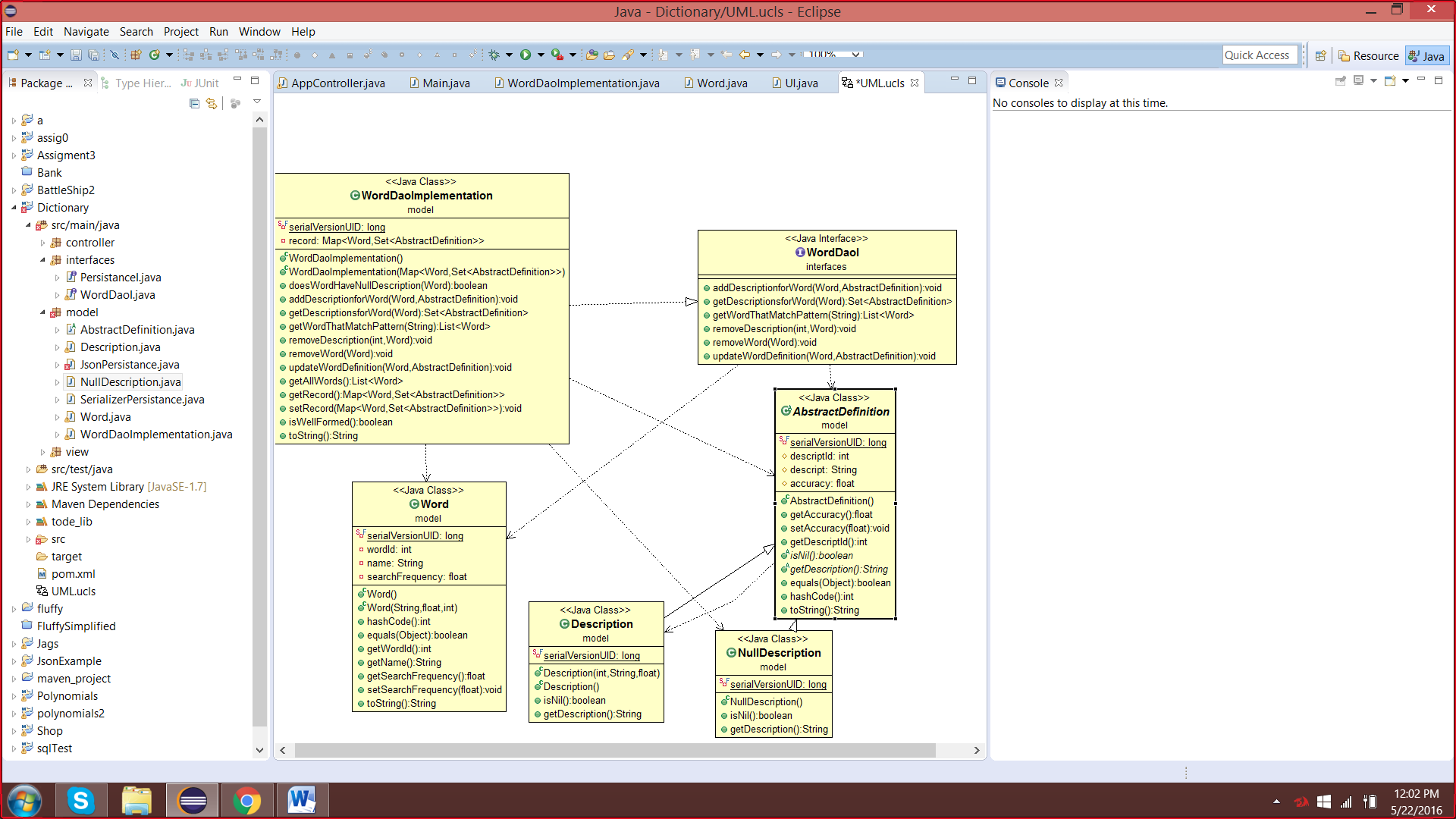
The methods of PersistanceI are save and restore, which are implemented differently by the 2 concrete classes, due to the fact that one is based on a Json and the other on a binary file.



## Data Access Object Pattern

Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services. Following are the participants in Data Access Object Pattern.

* **Data Access Object Interface** - This interface defines the standard operations to be performed on a model object(s).
* **Data Access Object concrete class** - This class implements above interface. This class is responsible to get data from a data source which can be database / json or any other storage mechanism.
* **Model Object or Value Object** - This object is simple POJO containing get/set methods to store data retrieved using DAO class.

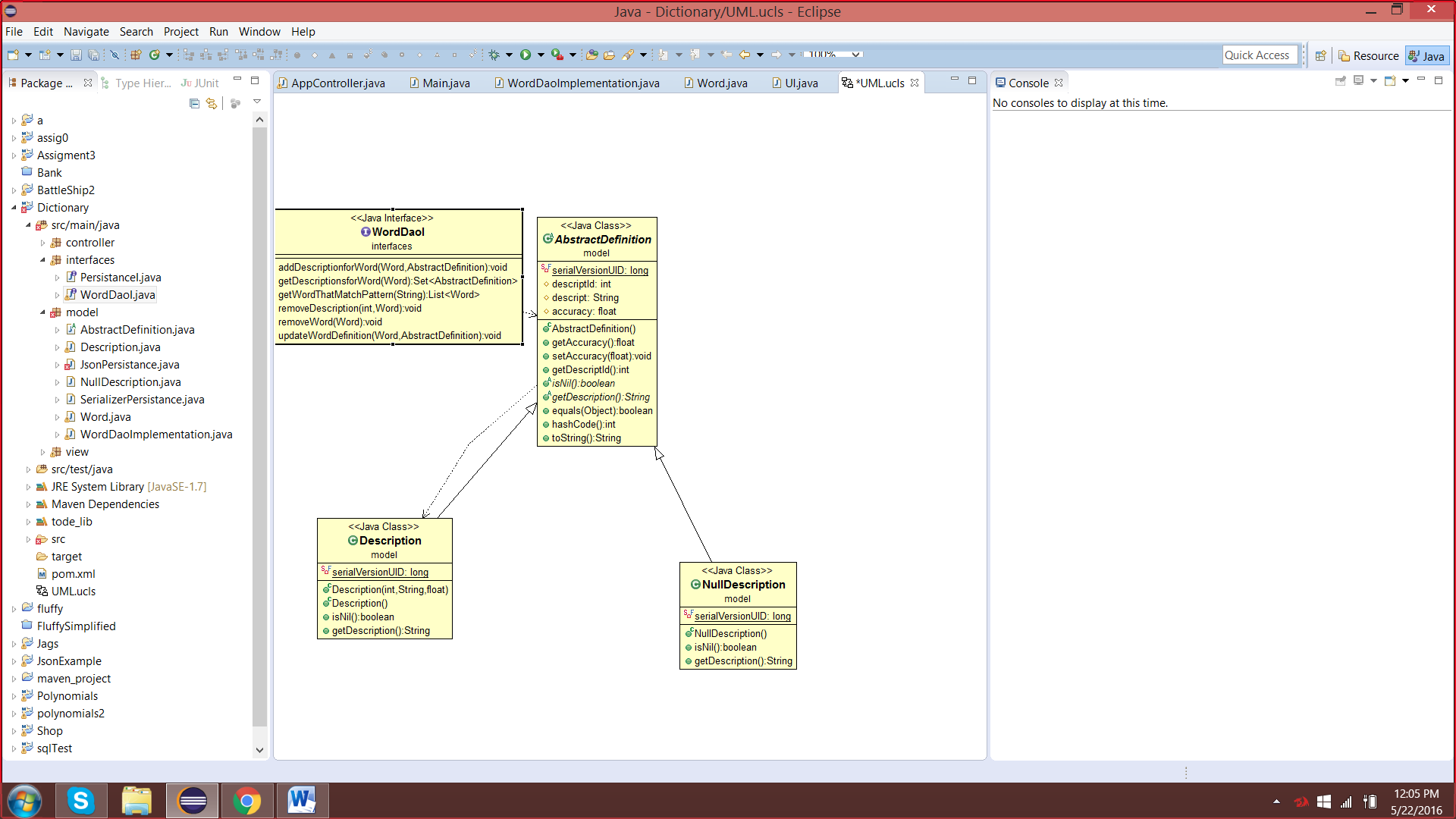


## Null Object Pattern

In Null Object pattern, a null object replaces check of NULL object instance. Instead of putting if check for a null value, Null Object reflects a do nothing relationship. Such Null object can also be used to provide default behavior in case data is not available.

In Null Object pattern, we create an abstract class specifying various operations to be done, concrete classes extending this class and a null object class providing do nothing implementation of this class and will be used seamlessly where we need to check null value.

We are going to create a *AbstractDefinition* abstract class defining operations. Here the name of the definition and concrete classes extending the *AbstractDefinition* class. A DAO class *WordDaoImplementation* is created to contain either *Description* or *NullDefintion* objects based on the name of customer passed to it.



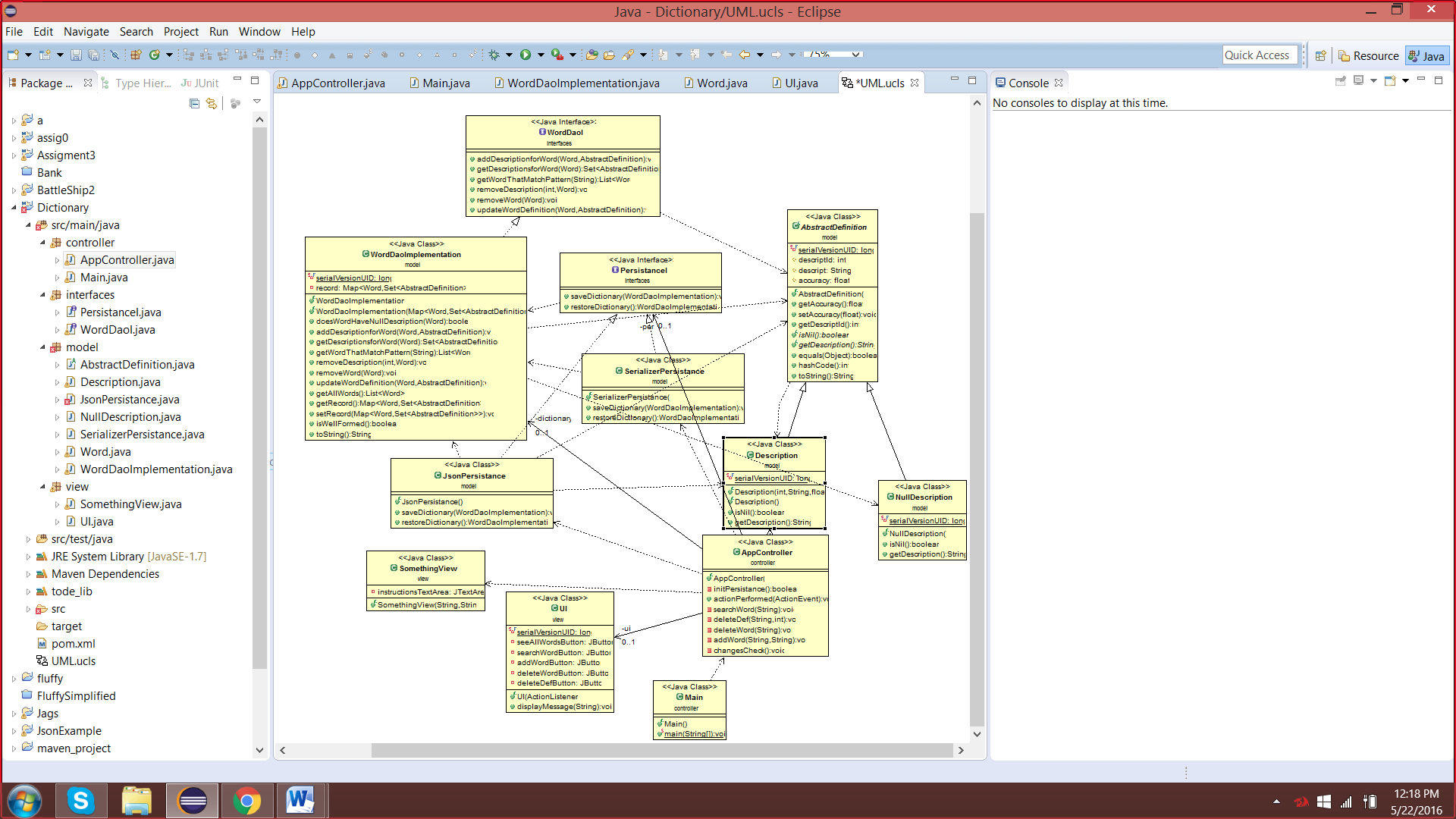
# Design

## Relational Diagram

The entity-relational model is a data model that describes aspects of a business domain or its process requirements, in such an abstract way that it can lead to being implemented in a database such as a relational database. The main components of ER models are entities and the relationships that can exist between them.

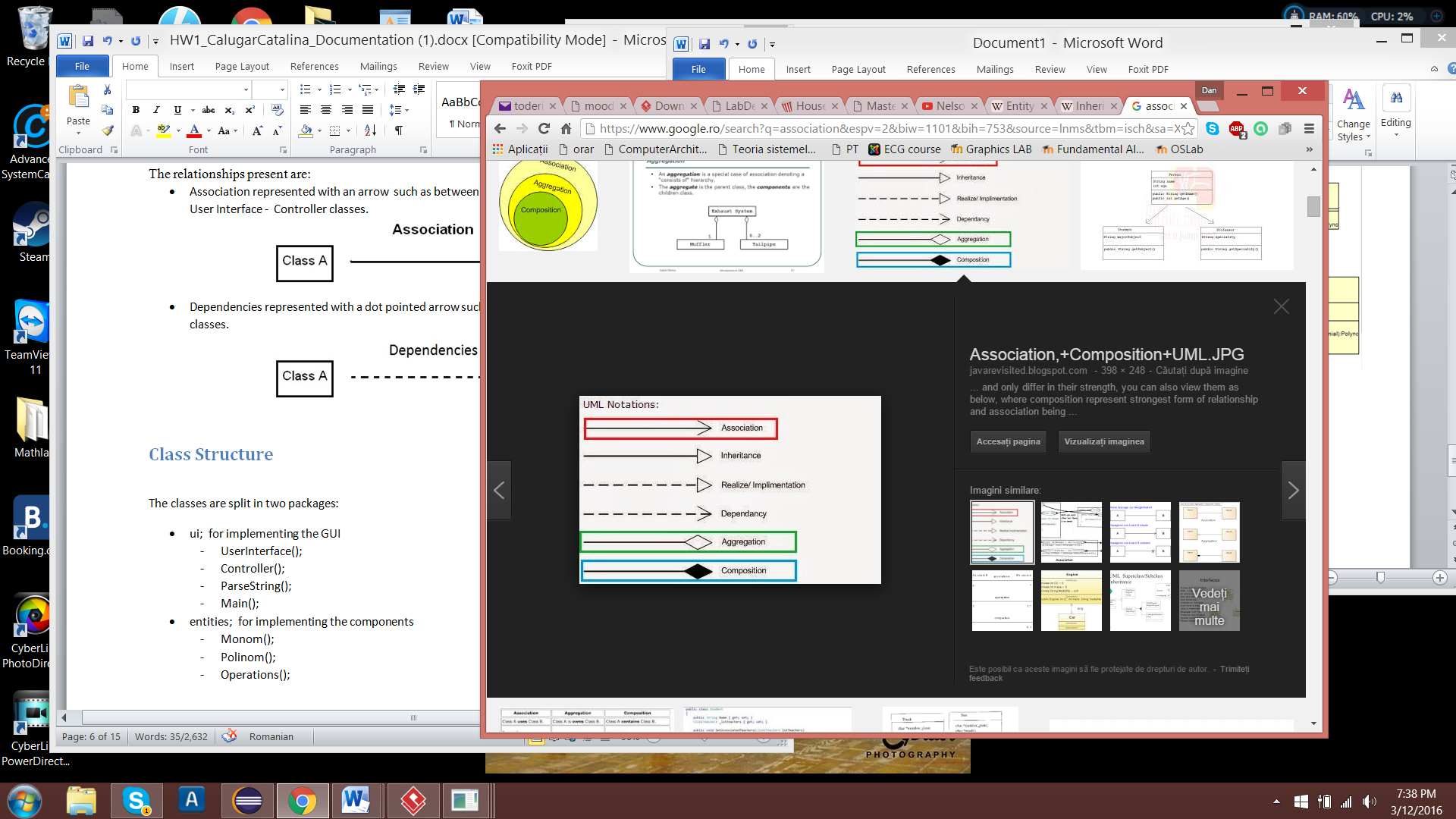
In most cases the ER design is the first step in solving any OOP problem. I have chosen to implement each operation into a separate class and also to add a class for testing. However, testing is needed only for verifying if the persistence classes work properly, because of the dimensions of the project and the number of operations done on the database, we must make sure that those classes are perfect. Also controllers, can have or not a hierarchic structures.

This is the relationship diagram containing associations and dependencies type relations:

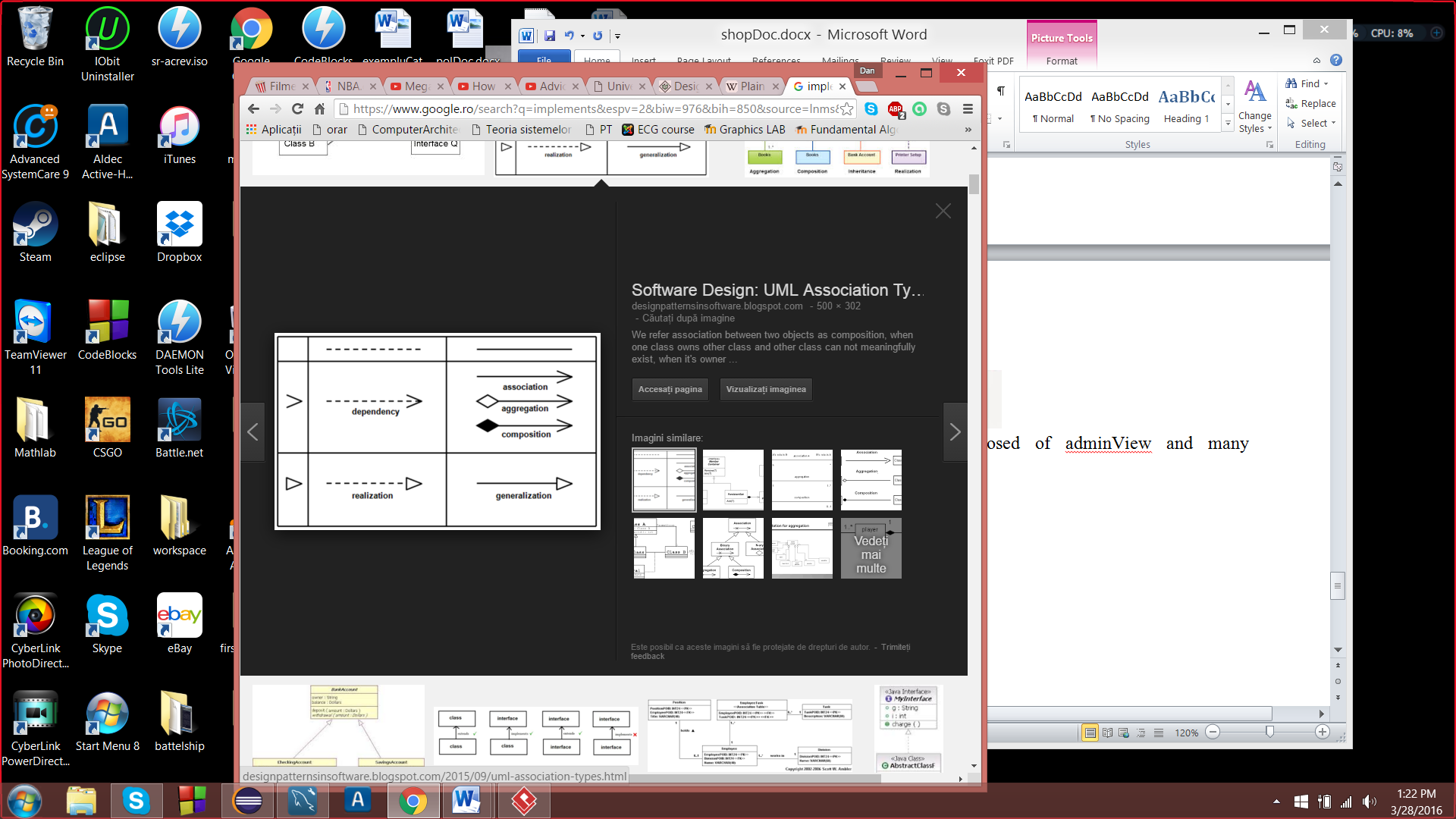


The relationships that are shown in the diagrams are:

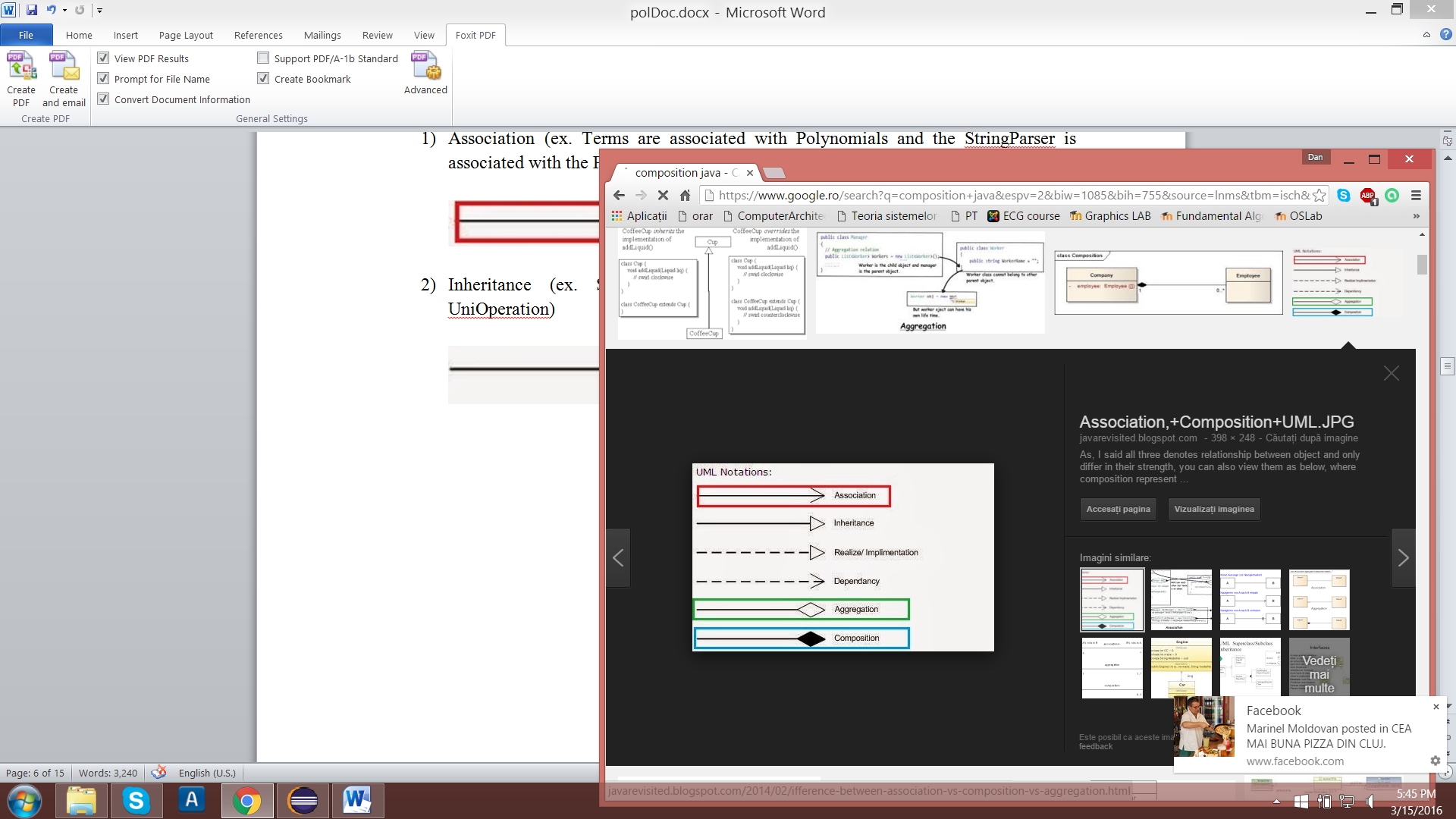
1. Association (ex. Word is associated with the Definition).



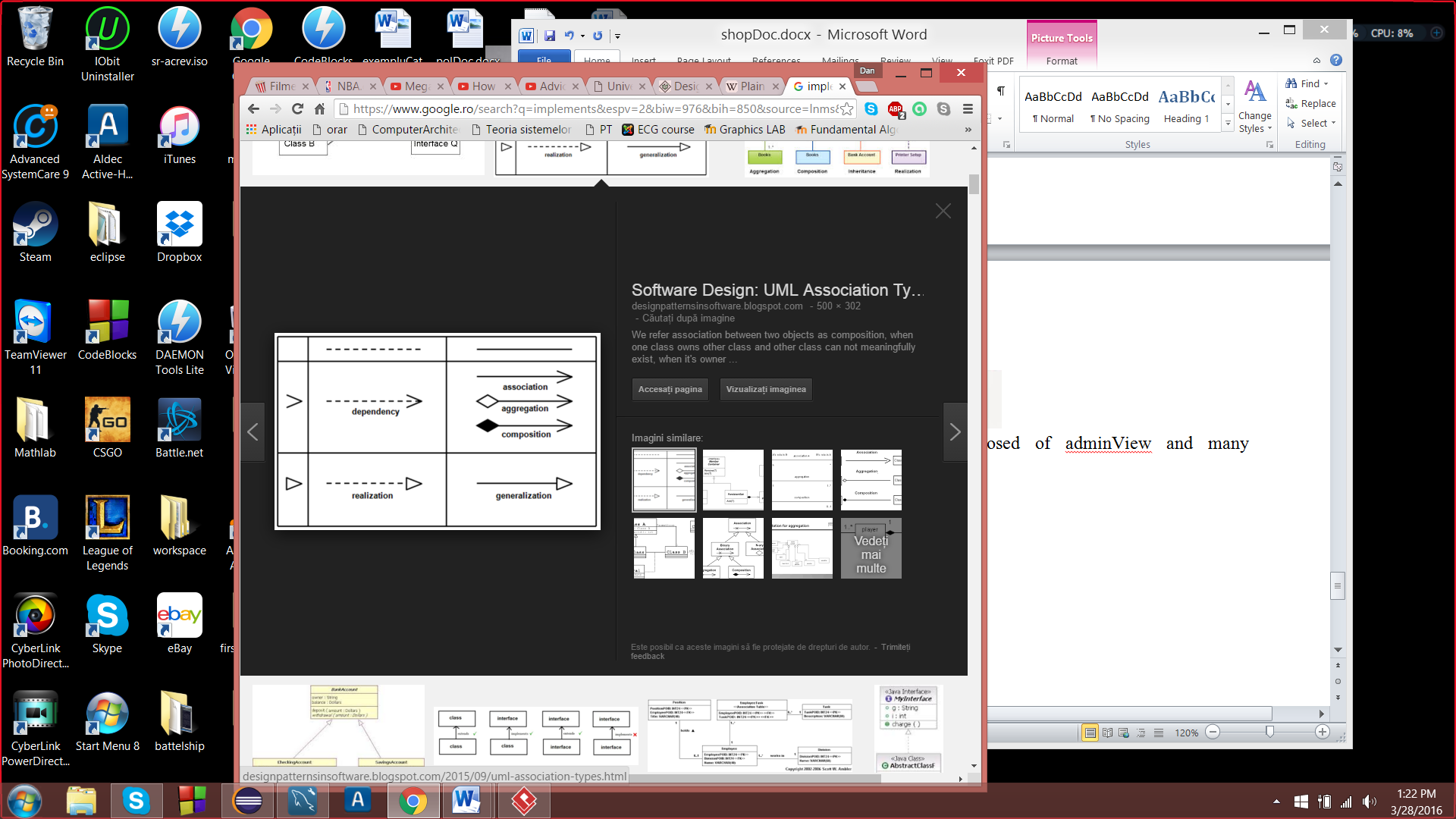
1. Implements (ex. Null Definition and Description implements DefinitionI)



1. Composition (ex. UIController is composed of Ui).



1. Dependency (ex. Serialiazer Persistance classes depends on Word Dao)



# Conclusion and further developments

Arriving to the last point of this presentation, I personally considered this project a good exercise for many things, such as layered pattern design implementation, MVC design and DAO classes creation and usage. Though, I firstly tried to implement a setVisibility() method that would help me in switching between views, however when putting it in Controller nothing seemed to work. I left my thoughts behind and I have fallen in the disgrace of opening each view and setting setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE).

Having more than one problem in hand, the persistence class came as a relief and helped me a lot in implementing my functionalities. In addition, I could have done many more functions based on such architecture, but maybe in the near future or in another project.

Some improvements and future development plans would be:

1. Check the formats of the inputted data if they have the right format, using regex statements or simple methods.
2. Creating a server, for more than one user to use this application at a time and for real time data to be transmitted
3. Have a multi-thread design for fetching and storing data
4. Design a better user interface with much more functionalities and prettier view
5. More notifications for the user, based upon his or her actions.

# References

<http://www.tutorialspoint.com/java/java_serialization.htm>l

<https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>

<https://en.wikipedia.org/wiki/Design_by_contract>