Thesaurus

Programming Techniques – Fifth Project

Group: 30424

First\_Name: Daiana

Second\_Name: Spatacean

Teacher: Claudia Pop

Summary:

1**. Introduction**

1.1 Problem specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2. **Description of the project**

2. 1 Problem analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2. 2 Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3. **Projection**

3. 1 UML diagrams . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . .

3. 1. 1 Use-case diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . .

3. 1.2 Class diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . .

4**. Implementation** . . . . . . . .. . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . .

5. **Implementation** **and testing** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

6. **Further developments .** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

7. **Conclusions** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

8. **References**  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. **Intoduction**

A **dictionary** is a collection of [words](https://en.wikipedia.org/wiki/Words) in one or more specific [languages](https://en.wikipedia.org/wiki/Languages), often [alphabetically](https://en.wikipedia.org/wiki/Alphabetical_order) (or by [radical and stroke](https://en.wikipedia.org/wiki/Radical-and-stroke_sorting) for [ideographic](https://en.wikipedia.org/wiki/Ideographic) languages), with usage of information, [definitions](https://en.wikipedia.org/wiki/Definitions), [etymologies](https://en.wikipedia.org/wiki/Etymologies), [phonetics](https://en.wikipedia.org/wiki/Phonetics), [pronunciations](https://en.wikipedia.org/wiki/Pronunciations), translation, and other information; or a book of words in one language with their equivalents in another, also known as a [lexicon](https://en.wikipedia.org/wiki/Lexicon). It is a [lexicographical](https://en.wikipedia.org/wiki/Lexicography) product designed for utility and function, curated with selected data, presented in a way that shows inter-relationship among the data.

A **synonym** is a word or phrase that means exactly or nearly the same as another word or phrase in the same language. Words that are synonyms are said to be **synonymous**, and the state of being a synonym is called **synonymy**. The word comes from [Ancient Greek](https://en.wikipedia.org/wiki/Ancient_Greek_language) syn ([σύν](https://en.wiktionary.org/wiki/%CF%83%CF%8D%CE%BD" \o "wikt:σύν)) ("with") and onoma ([ὄνομα](https://en.wiktionary.org/wiki/%E1%BD%84%CE%BD%CE%BF%CE%BC%CE%B1" \o "wikt:ὄνομα)) ("name"). An example of synonyms are the words begin, start, commence, and initiate. Words can be synonymous when meant in certain [senses](https://en.wikipedia.org/wiki/Word_sense), even if they are not synonymous in all of their senses. For example, if we talk about a long time or an extended time, long and extended are synonymous within that [context](https://en.wikipedia.org/wiki/Context_%28language_use%29).

**2.Description of the project**

The purpose of this assignment was to make some operations in a dictionary. The things that can be done are: add a new word and the synonyms of it, delete a word, search if a word is in the dictionary and a “special” search. This is a dictionary that contains only synonyms.

**3.Diagrams**

**3.1 Use-case diagram**

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

The user can be any person that wants to use the application and the interface must be pretty friendly.

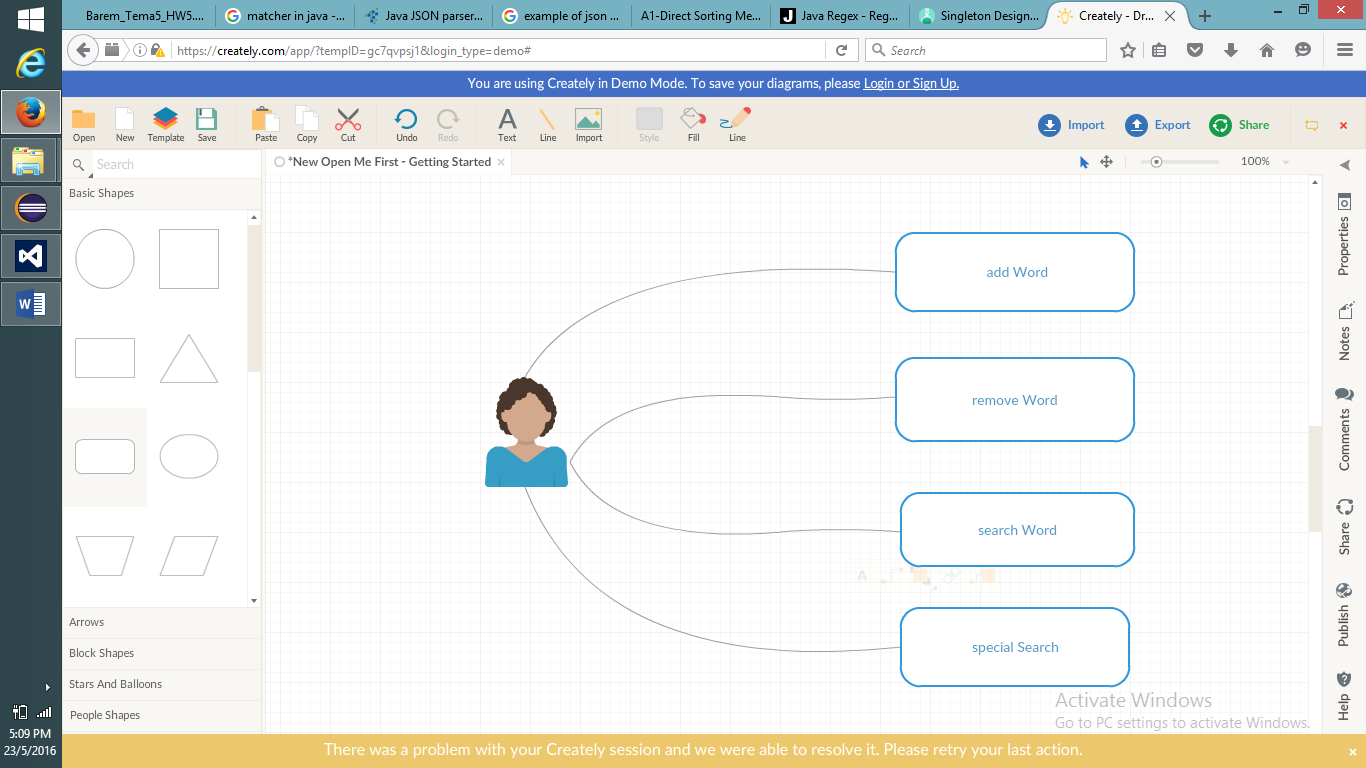
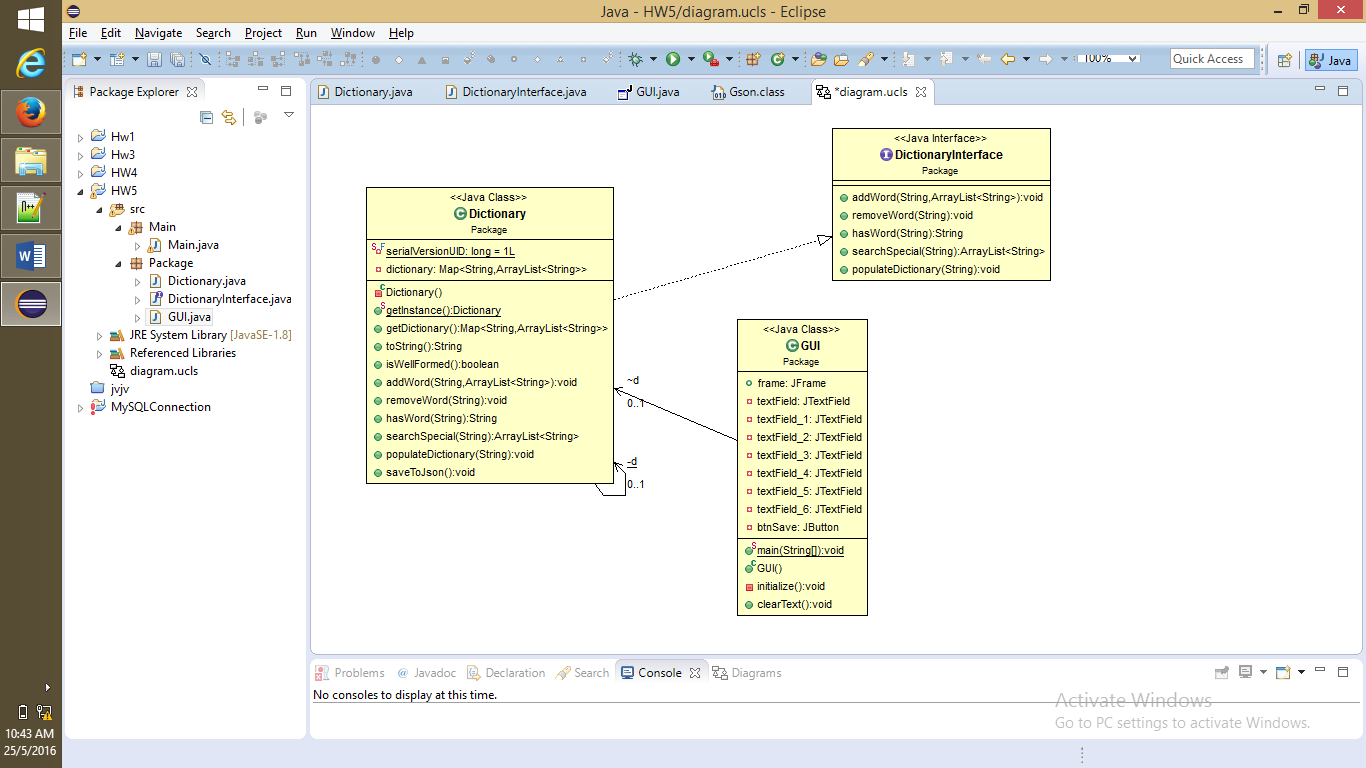


Fig. User- diagram

Fig. Package- diagram

**3.2 Class Diagram**



**4.Implementation**

This project is made following the OOP structure (classes, methods) and principles:

- **inheritance** when one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism

- **polymorphism** when one task is performed by different ways i.e. known as polymorphism. For example: to convert the customer differently, to draw something e.g. shape or rectangle etc.

- **abstraction** hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing

- **encapsulation** binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

Starting with the database:

* The table Customer contains the id of the customer which is a unique key, the name, the email address and the phone number
* The second table is called Product which also contains the id, name and quantity
* And the last one is about the Order

**Package**

* Class **DICTIONARY**

**-**here I have done the methods for the singleton pattern

**-** public static Dictionary getInstance() which returns an object of type dictionary and this would not let to make another object of that type

1. **public** **class** Dictionary implements DictionaryInterface {

Dictionary = new HaspMap <String, ArrayList<String>> ()

}

1. **public** **boolean** isWellFormed(){

**}**

1. **public** **void**  addWord()

{

* add a new word and synonyms

**}**

1. **public** **void**  removeWord()

{

* delete the word and all the synonyms

**}**

1. **public** ArrayList<String > searchSpecial ()

{

* find a word, with some special properties

**}**

1. **public String** hasWord()

**{**

* if a word is in the dictionary

}

1. **public** **void**  PopulateDictionary()

{

* populate the dictionary from the file

}

1. **public** **void**  saveToJson()

**{**

* save the words in the dictionary

}

* Class **DICTIONARYINTERFACE**
* Same classes as the bank class because this class is an interface that is implemented by the dictionary class
* Postconditions and preconditions for each method

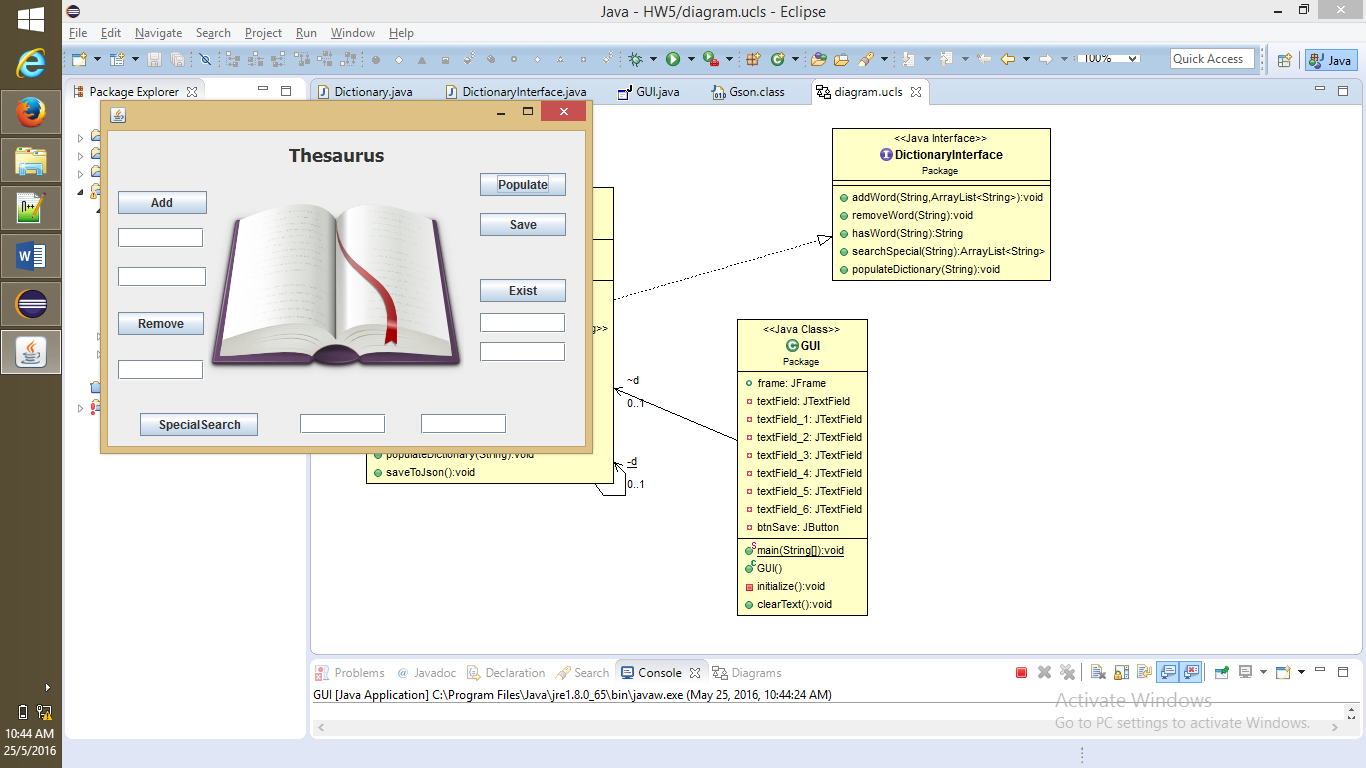
**Public**  **interface** DictionaryInterface

{

**}**

5. **Implementation** **and testing**

This is how the interface looks like:



* The **user** should:
* Enter a word and the synonyms for that word and the press the button
* Then remove the word using the remove button
* Search if a word is in the dictionary
* And the last one is a special search

Regarding the implementation process, I used as program Eclipse IDE. During the implementation of the project I made a lot of changes. The classes had many methods Before the interface was made I print the results in the console to see if they are correct and to know if I continue in that manner or I had to make some changes. The project contains only one package because is not very complex.

5. **Results**

I am pretty satisfied with my work and I think that the project is very easy to understand and the most important thing is that any type of user is able to use the interface and will have no problem with it. And in the end I think that I implemented all the petitions of the “client”.

6. **Further developments**

I think that I can make a very long list with the things that I want to improve related to my project with the dictionary:

* Starting from the “dictionary” of the project: I like to make more operations with the words and put some antonyms
* Also the window can be modified because it doesn’t look so nice
* More OOP with more classes and method with less lines of code and not so many duplicate code in some
* I want to have a more structured code
* And for the interface I want to make it much pretty
* For the buttons if I will make them with colors and with another type of writing and for the toolbar I want to implement help and menu and so on
* It will be nice if the size of the window can be modified from the user

7. **Conclusions**

To conclude, I can say that this project meant hard work, a lot of new things learned, focusing, development and creativity. Even if I encountered a lot of problems, I was able to fix them after all, by searching on the internet or asking a colleague for advice. I think that my application satisfies the requirements and the users will have at their disposal all its functionalities. And in the end I found out that is very interesting to work with database in java because you make the connection and everything that you modify in the database in any table it can be seen from the java program that you use.

8. **References**

* <https://en.wikipedia.org>
* <http://stackoverflow.com>
* <http://www.oracle.com/technetwork/articles/java/index-137868.html>
* <http://www.tutorialspoint.com/java/java_data_structures.htm>

And others!

.