Stoica Razvan Cosmin

Group 30424

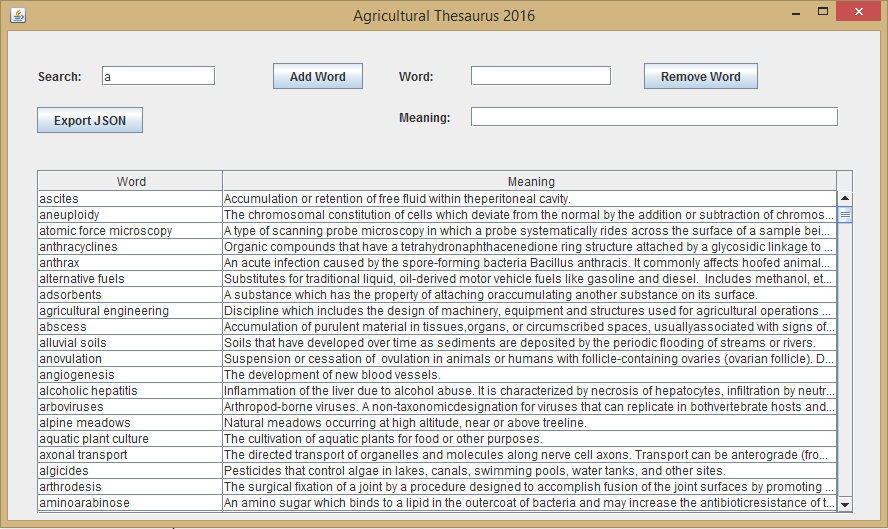
**Dictionary of synonyms application**

1. **Homework objective**

Design and implement a dictionary of synonyms (thesaurus) for Romanian or English language. It is required to use Java Collection Framework Map for the implementation.

1. **Problem analysis**

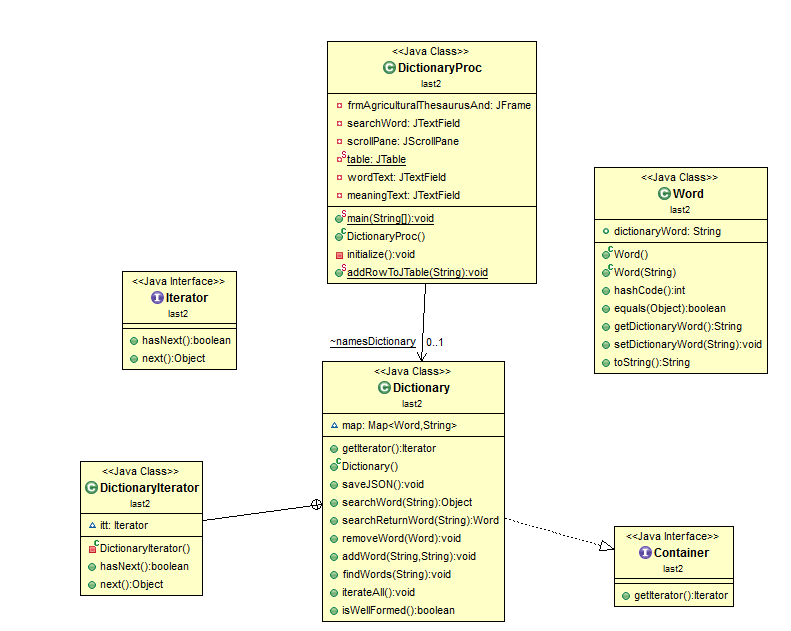
The program should be simple to use and have clear input/output display. The user interface will have 3 buttons and a search filed, where we search the words. The words and their meaning will be displayed on a JTable. We can also add or remove a specific word and export the dictionary in a JSON format.



1. **Design**

For the design of the project I’ve used the Iterator Design Pattern which was very useful with the hash tables. So the program has the following classes: DictionaryProc, Word, Dictionary and DictionaryIterator. And the following interfaces: Iterator and Container.

.



UML DIAGRAM

The Word class has a dictionaryWord string, with getters and setters and a constructor which puts a string in dictionaryWord. We also override the hashCode(), equals(), and toString() methods.

The Dictionary class implements the Container interface and overrideshe Iterator method and returns a new DictionaryIterator() class. The DictionaryIterator is a private class which implements the iterator and overrides the hasNext() and next() methods, the scope of this class is to be used when iterating the hash table. So in the Dictionary Class we have a Map <Word, String>, which is a hash map that will be used to get all the information from JSON file. In the constructor we parse the JSON file containing the thesaurus data and we add it to the map. In the saveJSON() method we save put the map to a JSON object and then we export it to a file. In the searchWord() method we iterate all the dictionary and we return a Word object with the given String word if we find it. In the searchReturnWord we do the same thing except this time we don’t return a word obect we return the key from the map. The removeWord() method removes a given word. The addWord() method adds a word and a meaning to the map if it is not null or it doesn’t exist. The findWords() method has a String argument which will be used to print all the words that match with it (kind of like how Find works in every text editor). The iterateAll() method prints all the words and their meaning to the console. The isWellFormed() method is used to check if everything in the map is correctly added. This method is used with the enabled assertion to the class. We also have invariants pre and post conditions for every method in this class.

The Iterator interface has 2 methods which will be overridden in the DictionaryIterator class, this methods are hasNext() and next().

The Container interface has a getIterator() method which will be overridden in the Dictionary class.

The DictionaryProc class has an object of the Dictionary class and an addRowToJTable() method which will add from the hash map all the words and their meaning to the JTable. Here are all the buttons and labels that are used by the interface. The JTable is inside a JScrollPane which will make us able to scroll down while searching for a specific word.

As for packages we have in the Dictionary class: java.io.FileNotFoundException, java.io.FileReader, java.io.FileWriter, java.io.IOException, java.util.Collection, java.util.HashMap, java.util.Iterator, java.util.Map, java.util.Set, org.json.simple.JSONArray, org.json.simple.JSONObject, org.json.simple.parser.JSONParser, org.json.simple.parser.ParseException. In the Container interface: java.util.Iterator. In the DictionaryProc class we have all the expected java swing awt and util packages.

1. **Implementation and testing**

The operations work correctly if they are written in the expected format, if not the program will throw an exception (the user will see a pop-up message that will warn him/her if he did something wrong). I tested each class before using the user interface, and I tried to print every possible outcome to make sure that the application doesn’t break or throws unexpected exceptions.

1. **Results**

All the given operations work correctly and the application displays error messages in case the user inputs something wrong.

1. **Conclusion and further developments**

As a conclusion the project needs a lot of improvements in order to be optimal and bug free, and as for further developments a method for checking dictionary consistency should be implemented but in our case it is impossible since we don’t have the same synonyms for the words.