Technical University of Cluj-Napoca

Dictionary

-project documentation-

Pop Andrei

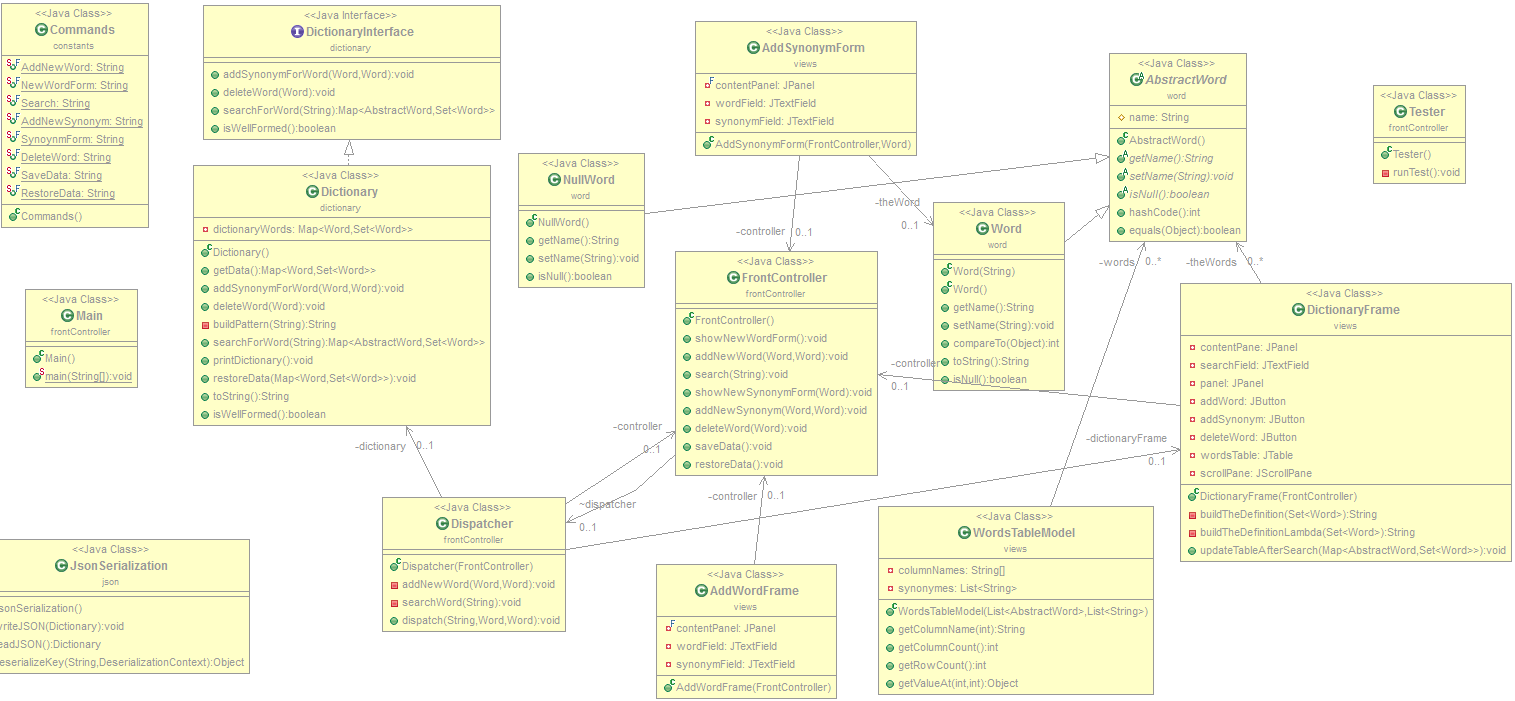
Group 30424

1. Objective

The objective of this project was to implement a dictionary simulation. Besides this objective, probably the main objective was to learn how to use the Null Object and Front Controller design patterns. The Null Object design pattern is used when we have multiple objects and some of them are null. In this case we create a dummy object which will be of type null and a predefined name. The Front Controller Design Pattern is used when dealing with multiple interaction between the user and the application. This means that when there are multiple requests, the controller takes all the requests and sends them to the controller. The controller is in charge of solving all the requests and send back a response to the user or to the object that passed a request.

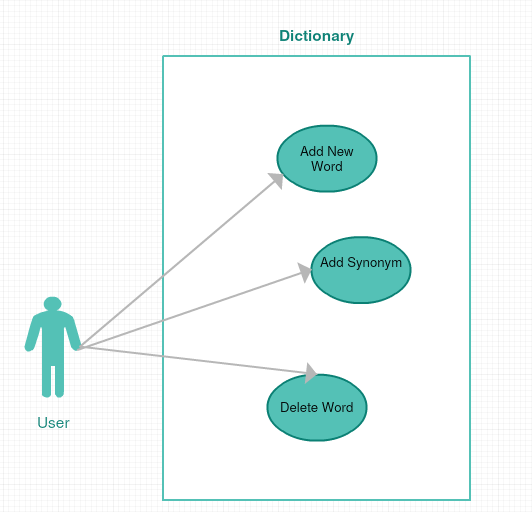
1. Problem analysis, modeling, scenarios, use cases

In order to implement a dictionary simulation, I have created multiple entities which correspond with real life objects. Each user can do certain actions like add, delete or update the dictionary. Also, there are more options like creating a new word, delete an existing word, find all word, …. Each work is unique in the dictionary. When searching for a word in the dictionary, the user has the possibility to user special characters like \* or ?. The \* character is used to substitute any string of any length. The ? character is used to substitute only a single character. I have used multiple packages to group the classes which are related to each other. When the user wants to add an new word, he or she also has to add a synonym for that word. There can’t be words without synonym in the dictionary. In case of a word that doesn’t have a synonym, an exception would be thrown, and the application would stop working. Also, in order to ensure a good functioning of the application, there have been placed asserts at the beginning of each method from the dictionary class. The assertions from the beginning are checking if the parameters which were passes to the function are valid, and if so, then the function is executed and a result will be returned. At the end of the function, there are also some assertions. The assertions from the end of the function are meant to check if the execution of the function yielded a good result. At the beginning of each function and also at the end, the method isWellFormed is called in order to check the consistency of the dictionary. If the dictionary isn’t in a consistent state, then an error is thrown. In order to implement the dictionary and design it by contract, an interface have been created. In the interface, all the requirements have been specified before the specification of the method. Also, the in the interface there have been specified the types of the arguments and what each function returns.

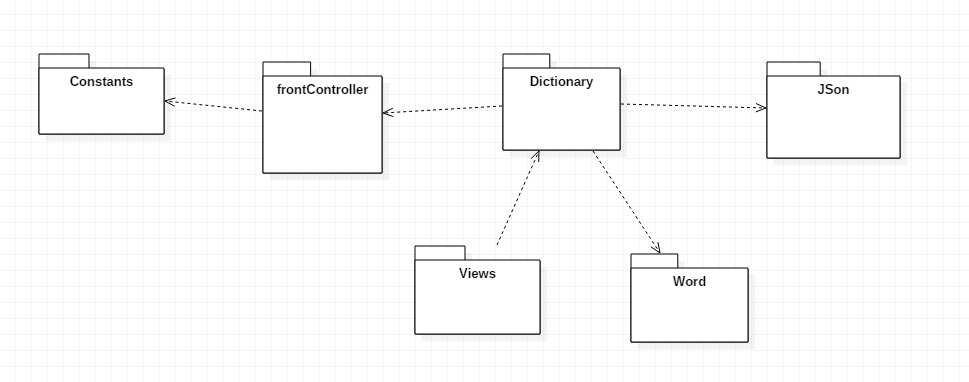
1. Design

3.1 UML Diagram

3.3 Use Case Diagram



3.4 Package Diagram



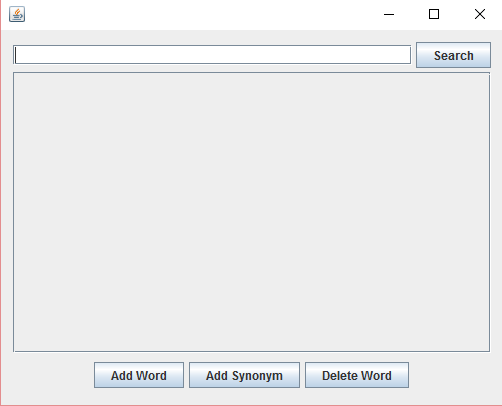
3.5 Class design

The application is composed of 15 classes and 6 packages. Each of the packages contains classes which fulfill a certain requirement. The main packages of the application are:

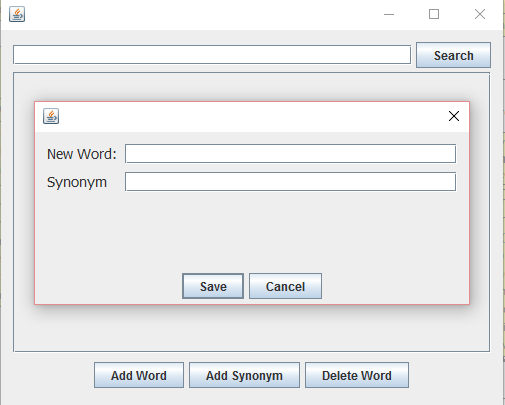
* **FrontController package**: this package was designed in order to launch the application. When the application is first started, the class Start is instantiated.
* **Word package**: this package contains the models from the application. The models are: Null Word and Word. It also contains an abstract class which specifies the behavior of the classes. The Word class contains all the information about a word from the real life. It contains the name and the description of the word. The Null Word is created in order not to return a null when the desired word is searched.
* **JSON package**: this package contains specific classes which perform the serialization and deserialization. The serialization is done each time the application is closed, by the Serialization class. The deserialization is done by the same class. There are two methods which realize the serialization and the deserialization. In order to deserialize the hash map I have implemented also the key encryption method.
* **View package**: this package contains all the classes which are part of the graphical user interface. The words from the dictionary are displayed in a JTable when they’re searched into the dictionary. If the words that match the requirements could not be found, then a Null Word is returned and a specific information is displayed in the table. Also, when adding a new word, a new frame is created. After the fields are written, then the word is added into the dictionary.
* **Dictionary package**: This package contains all the information about the dictionary. It actually contains the interface which specifies the behavior of the class dictionary and the class which contains the specific implementation of the interface. The dictionary class contains all the methods which are used in order to offer a good functionality for the dictionary.
* **Commands package**: This package contains all the commands which can be executed by the dispatcher. I have created this package in order to have a global understanding of the commands. If the commands would have been created locally, then problems of communication could occur. Also, it would have been much harder to declare the same set of the commands in each of the classed that use them in order to establish a common language for the classes that communicate between them and implement the functionality of the dictionary.

3.6 Graphical user interface

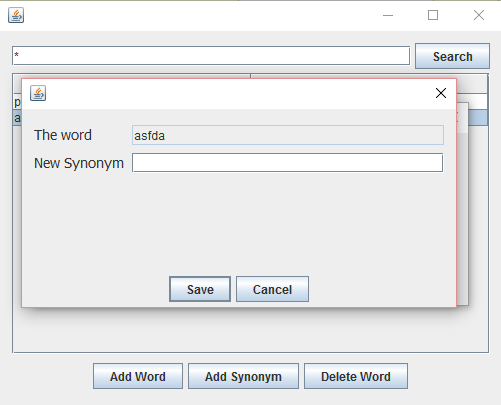
The graphical user interface was created in order to offer the possibility of simulating the dictionary operations. It has a main window which displays all the words from the dictionary and the results of the search when necessary. All the functions are called by pressing a button which will do a certain action on the dictionary.



Another window is created when having to add a new word into the dictionary. Each time a new word is added, a synonym is added with it in order to keep the dictionary consistent. It also exists the possibility to update an existing word by adding another synonym.



When updating a word into the dictionary, another window is created. In this case, the word for which a synonym is added can’t be modified, only the synonym can be modified.



1. Future improvements

The application was designed only to exemplify how some design patterns can be used an how to adapt them to a specific context. In order to have a real world application, more classes need to be added. Also another problem is that there isn’t a differentiation between the users and the administrators of the dictionary. An improvement that can be achieved would be to restrict the ordinary users to a certain set of actions that they can perform in the application. Another improvement would be to create multiple threads that can work concurrently. In this way, more persons can perform operations on their associated dictionaries without influencing each other and having the application running at the same speed and offering the same performance. Also, in order to be used in the real world, the application would need another design. The design is one of the most important aspects when designing an application which will be available on the market because the non-specialized user will evaluate the application depending on how it looks and not how it works. Another improvement would be the run the application on a web platform such that it wouldn’t be accessible only locally. In order to protect the unauthorized access in the application, another future development of the application would be to offer accounts with different access options. In order to do this only a few administrators would have full rights. The other users would have regular accounts which can be used whenever they want to perform an operation. Another remarkable improvement would be to store all the words into a database instead of storing them locally.

5.What I have learned -- conclusions

During the implementation of this application I have learned how to use and how to integrate the Null Object and the Front Controller design patterns. I have also learned that most of the design patterns are already implemented in Java so I don’t have to implement them when I’m using them. One of the most important things that I have learned during this assignment was how to serialize objects using JSON. I think the serialization is a very important thing to know when working with multiple objects. The tricky part was to perform the deserialization from the file. Another important thing that I have learned during the development of the application was to work with streams. I think it is very important to have the ability of working with streams because nowadays we are talking more and more about big data and this is a big issue. In order not to store all the data, it is much easier to work in it while receiving it an passing it over to other mechanisms in order to be further modified.

6.Bibliography

Books:

- Joshua Bloch, Effective Java (2nd Edition);

- Kathy Sierra, Bert Bates, Head First Java (2nd Edition), O'Reilly Media;

- Barry Burd, Java for Dummies (5th Edition), Wiley;

Websites:

- http:// stackoverflow.com/

- https:// www.oracle.com/java/

-http:// www.oracle.com/technetwork/articles/javase/index-142890.html

- http:// www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/

- http:// christoph-burmeister.eu/?p=1556

- http:// tutorials.jenkov.com /