Dictionar

Student:Mocan Ioana

Grupa:30223

Cuprins

1. Obiectivul temei

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

3. Proiectare (diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4. Implementare si testare

5. Rezultate

6. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

7. Bibliografie

1. **Obiectivul temei**

Obiectivul temei prezentate este acela de a crea o aplicatie care sa scoata in evidenta cum se poate realiza un dictionar care sa fie consistent si care sa contina cuvinte sinonime pentru fiecare cuvant.Comenzile realizate in aceasta aplicatie este de cautare a unui cuvant si de adaugare de cuvinte noi si de sinonime noi.

**2. Analiza problemei, modelare, scenarii, cazuri de utilizare**

Prin analiza problemei ne dam seama ca avem nevoie de concepte care privesc lucrul cu programare obiectuala.Problema a fost analizata din punct de vedere al unui dictionar online in care se pot face cautari de cuvinte si sa afiseze cuvintele cu tot cu sinonime.Problema este una care permite diferite abordari si nu este exclusa ideea ca fiecare abordare sa aiba rezultate diferite.In cazul dezvoltari problemei elementele incluse in rezolvarea problemei variaza de la ca la ca ele sunt fie mai bine conturate in ceea ce priveste incapsularea, mostenirea, polimorfismul, abstractizarea.Pentru a putea implementa corect problema este necesar sa fim atenti la datele necesare si la comunicarea cu exteriorul.

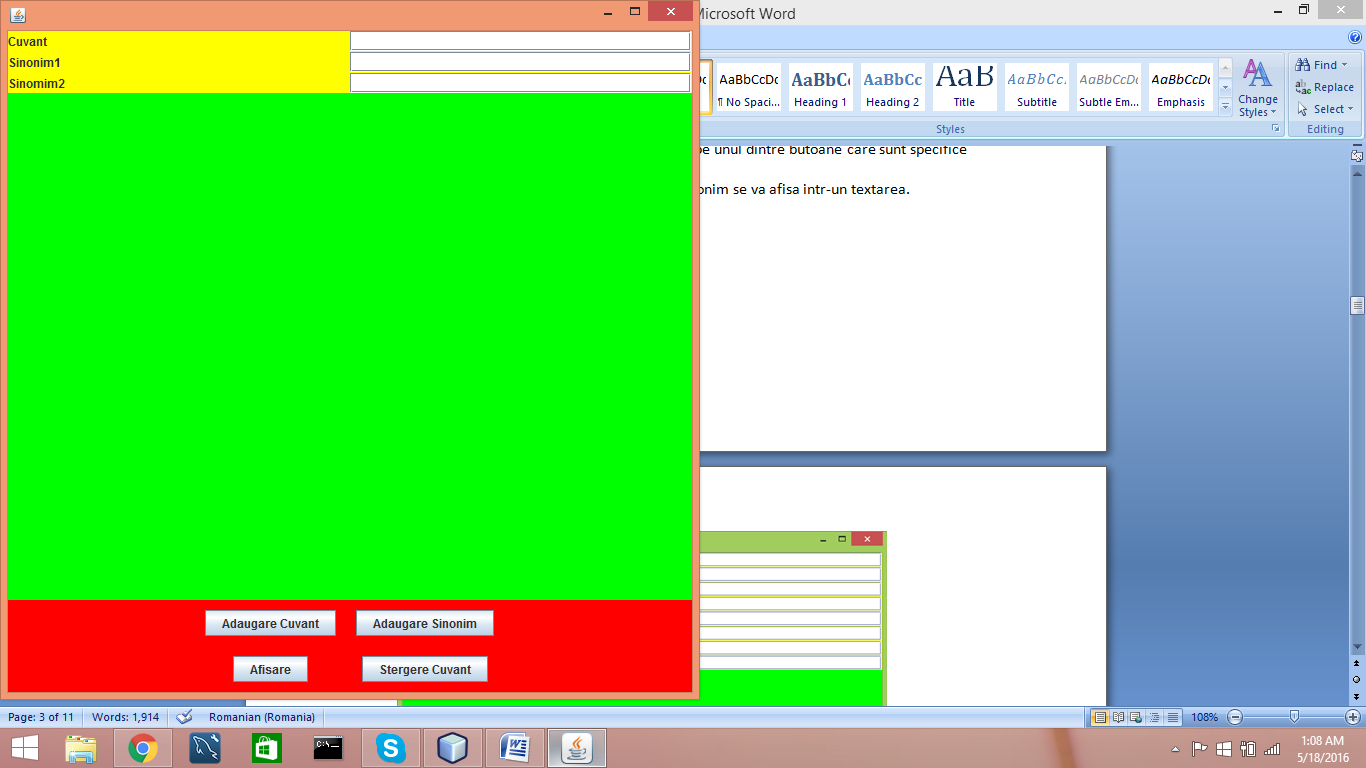
In analiza problemei am luat in considerare faptul ca avem un dictionar online in care trebuie sa adugam cuvinte noi aparute impreuna cu tot cu sinonimele lor si mai avem nevoie sa putem afisa cuvintele pe care vrem sa le cautam impreuna cu listele de sinonime.Fiecare cuvant poate fi gasit in dictionar printr-o combinatie de sir lipsa sau caracter .Daca cineva doreste sa caute un cuvant si nu stie exact cu se scrie atunci poate fi inlocuit cu \* pentru un sir de caractere sau ? pentru un caracter dar se poate scrie si combinatie de \* si ? pentru a gasi cuvantul cautat.La un anumit cuvant putem introduce si sinonime noi si daca dorim sa afisam tot dictionarul scriem \* si afisam.Inacelasi timp putem sa stergem si un cuvant din interiorul dictionarului daca din intamplare nu a fost pus unde trebuie sau daca cuvantul nu mai exista.

Un aspect care conteaza in relatia utilizator-produs este crearea unei interfete cat mai prietenoase si care sa scoata in evidenta cat mai bine utilizarea codului.

Din punct de vedere a implementari aplicatiei exista mai multe optiuni de implementare ceea pe care am ales-o eu este una care poate fi imbunatatita. Varianta pe care am ales-o sa o implementez poate sa lase putin de dorit.Am incercat sa evidentiez cat mai bine ce se intampla intr-un dictionar online si sa se demonstreza ca tot ce se intampla intr-un dictionar si cum se pot introduce noi cuvinte si cum se realizeze consistenta in acelasi timp cu introducerea de noi cuvinte si de noi sinonime.

Utilizarea aplicatiei este descrisa in ceea ce urmeaza:

* In functie de operatia care trebuie efectuata se vor lua in considerare unul dintre textfiel-urile care sunt evidentiate in interfata grafica
* O data completate textfielul se apasa pe unul dintre butoane care sunt specifice diferitelor obtiuni prezente.
* Informatia despre cuvant si despre sinonim se va afisa intr-un textarea.



**3. Proiectare (diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)**

Penru implementarea aplicatiei este necesara utilizarea a 7 clase, fiecare fiind specializata in solutionarea unor operatii specifice.

**public class Main**

Aceasta clasa are rolul de a apela clasa View care creeaza interfata si utilizeaza operatiile aplicate in interiorul unui depozit.Practic are rolul de a rula aplicatia si preluand tema sistemului de operare pentru a crea casetele cu interfata grafica si pentru a o expune pentru utilizator.In acesata clasa se mai realizeaza si deserializarea.

**public abstract class AbstractWord**

Aceasta functie este folosita pentru a putea crea si NullObjectWord care este creat doar pentru a verifica daca daca exista obiecte nule.Aceasta clasa este una abstracta care ulterior este folosita in clasa Word si NullWord.

**public class Word extends AbstractWord implements Cloneable**

Aceasta clasa este pentru a crea un cuvant aceasta clasa extinde clasa abstracta cuvant si implementeaza Cloneable pentru a putea crea o clona a unui cuvant.

**public class NullWord extends AbstractWord**

Aceasta clasa este este clasa facuta speciala pentru a putea verifica daca un obiect de tip word este null. Aceasta clasa extinde clasa abstracta word pentru a putea impementa metodele acelea de set si de verificare de null.

**public interface DictionarProc**

Aceasta clasa este una de tip interfata si aici sunt declarate antetele metodelor.Aceste metode vor fi mai apoi folosite in clasa Dictionar care implementeaza HashMap-ul

**public class Dictionar implements DictionarProc**

In aceasta clasa se utilizeaza un HashMap care ne ajuta sa retinem informatia pe care o avem de retinut adica avem ca si cheie obiectul de tip cuvant iar ca si lista avem o lista de cuvinte acestea reprezinta sinonimele cuvantului cheie.In aceasta clasa pe langa constructor mai avem metodele implementate din interfata DictionarProc si mai avem si doua functii care una care verifica daca dictionarul este bine format si o functie care vefica daca un cuvant este in interiorul dictionarului.

**public class View extends JFrame implements ActionListener,ItemListener**

Clasa View are ca rol implementare a interfetei grafice si asigurarea ciclului de informatie dinspre interior sper exterior si invers.Toata implementarea grafica se realizeaza cu ajutorul a trei panel-uri care sunt raspunzatoare de datele de intrare ,de butoane si de datele de iesire.

Interfata grafica este realizata din 4 butoane care se ocupa de realizarea operatiilor.Pe langa butoane se mai gasesc si label-uri care ajuta la intalegerea corecta a semnificatiei datelor introduse si apoi afisate intr-un textarea.

Fiecare buton este responsabil sa indeplineasca o operatie singulara si care sa faca ceea ce ii este cerut.

Butonul „Adaugare Cuvant” are rolul de a aduga un cuvant inpreuna cu lista de sinonime in dictionar.

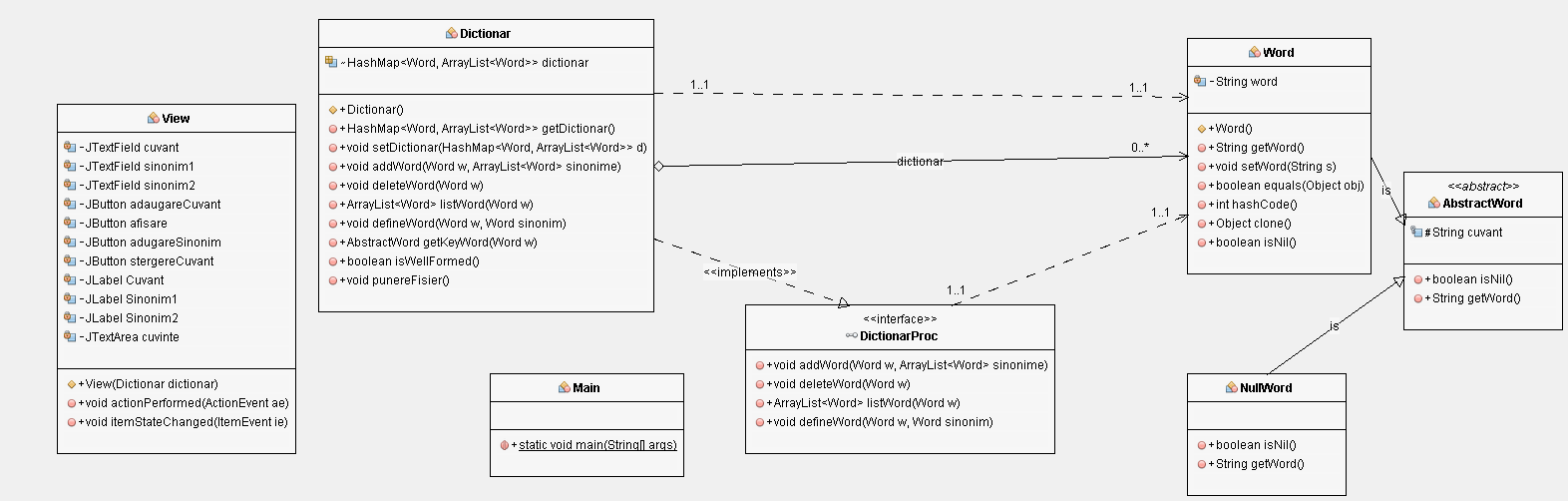
Butonul „Afisare” are rolul de a afisa continutul dictionarului sau de a afisa cuvintele pe care le cauta.

Butonul „Adaugare Sinonim” are rolul de a adauga un sinonim in lista de cuvinte din map.

Butonul „Stergere Cuvant” are rolul de a sterge un cuvant din lista de cuvinte din map .

**4. Implementare si testare**

Dupa cum se observa din diagrama UML s-au respectat paradigmele programarii orientate pe obiect. Abstractizarea se poate observa prin evidentierea elementele indispensabile.Conceptul de incapsulare se verifica la declararea private a elementelor din fiecare clasa , acestea putand fi accesate doar cu ajutorul unor metode de tip get si set.



In ceea ce urmeaza o sa explic metodele mai importante din fiecare clasa:

**public abstract class AbstractWord**

**public abstract boolean isNil();**

**public abstract String getWord();**

Una din metodele importante pe care o mentionam aici este aceea de **public abstract boolean isNil()** acesata metoda o sa o implementam in clasa word si nullword in clasa word o sa aiba valoarea **false** iar in nullword o sa aiba valoarea **true**.Si **public abstract String getWord()** este o metoda in care la word o sa folosim de obicei adica un get normal adica returneaza un cuvant si in clasa nullword o sa returneze faptul ca nu exista in dictionar.

**public class Word extends AbstractWord implements Cloneable{**

In aceasta clasa sunt metodele implementam cele doua metode din clasa abstracta dar pe langa mai avem si metoda de set si avem metodele de equals si hascode pe care trebuie sa le suprascriem pentru ca sa nu avem dubluri in interiorul mapului. Aici am implementat si metoda de clonare care am folosit-o pentru a utiliza design patternul prototype pattern care face o clona si suprascrie metoda **clone().**

**public class NullWord extends AbstractWord**

Una din metodele importante pe care o mentionam aici este aceea de **public boolean isNil()** care are valoarea **false**.Si **public String getWord()** este o metoda care o sa returneze **"Nu exista in dictionar"** acesata functie este specifica unu obiect null.

**public interface DictionarProc**

**public void addWord(Word w,ArrayList<Word> sinonime);**

**public void deleteWord(Word w);**

**public ArrayList<Word> listWord(Word w);**

**public void defineWord(Word w,Word sinonim)throws IOException;**

In aceasta clasa scriem antetele metodelor care mai apoi trebuie implementate in clasa de Dictionar .In principiu sunt metode care aduga sau sterg ceva din map si de afisare a elementelor din interiorul mapului.

**public class Dictionar implements DictionarProc**

**public boolean isWellFormed()**  acesata functie este special creata pentru a verifica daca map-ul este gol sau nu aceasta conditie este verificata de fiecare data cu ajutorului assertului.

**public void addWord(Word w, ArrayList<Word> sinonime)** este pentru a aduga un cuvant impreuna cu lista de cuvinte si se verifica daca mapul este bine format si apoi se aduga in map.

**public void deleteWord(Word w)** se realizeaza stergerea din interiorul mapului,a unui cuvant impreuna cu lista de sinonime.

**public ArrayList<Word> listWord(Word w)** se realizeaza afisarea cuvantului cautat impreuna cu lista de cuvinte,aici folosim clone care a fost implementata in interiorul clasei word.

**public void defineWord(Word w, Word sinonim) throws IOException** aici se introduce un nou sinonim pentru un anumit cuvant.

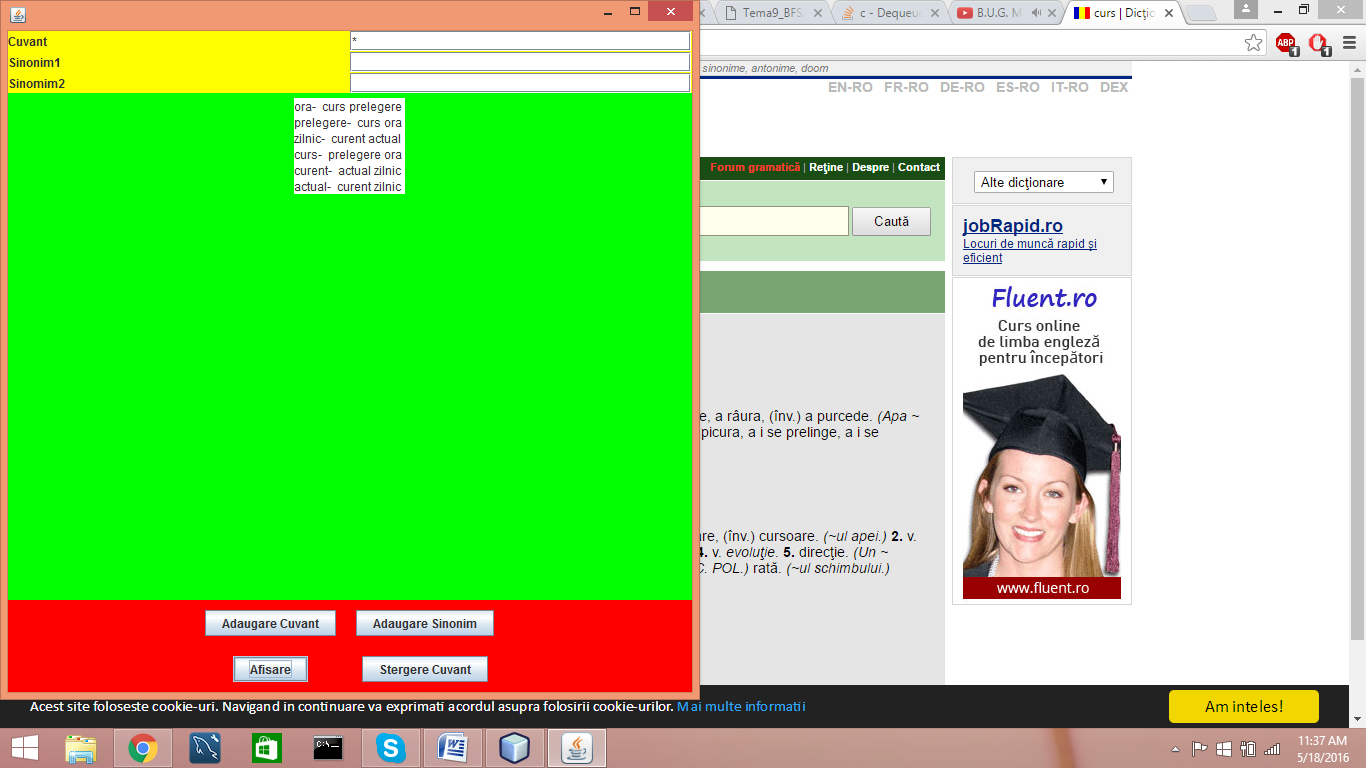
**public AbstractWord getKeyWord(Word w)** acesata clasa returneaza un obiect de tip null sau cuvantul pe care il cautam.Acesata functie este folosita pentru a verifica daca un cuvant se afla in dictionar.

**public void punereFisier()**  aceasta functie este folosita pentru serializarea mapului cu ajutorului json care este un alt tip de serializare cu o anumita forma in interiorul fisierului.

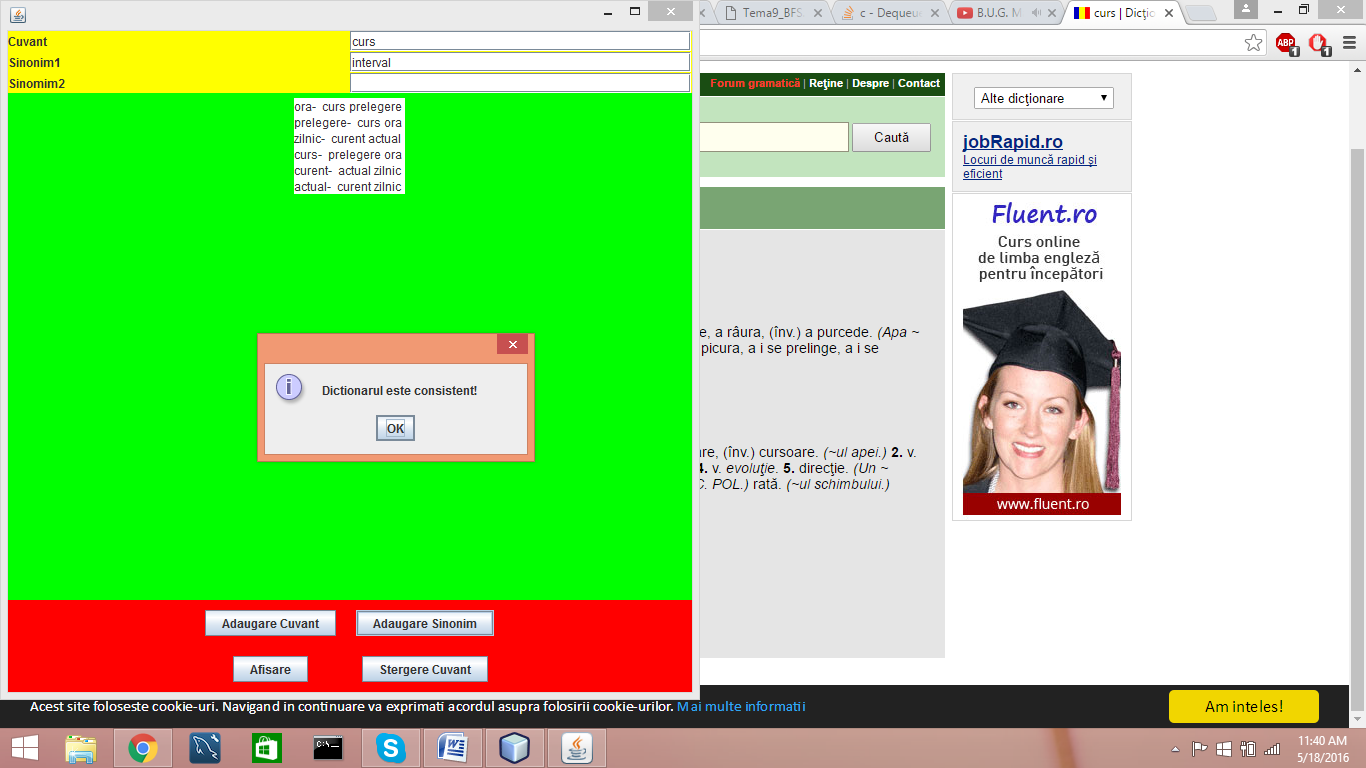
Am folosit doua design patternuri acestea sunt prototype si nullObject. Prototype pattern se refera la crearea obiectului duclicat dar sa se tina in minte performanta.Acest tip de pattern vine sunt un model creational si este un unuldintre cele mai bune metode dea a crea un obiect. Acest tipar implică punerea în aplicare a unei interfețe prototip care spune ca cream o clonă a obiectului curent. NullObject pattern se refera ca un obiect null inlocuieste instanta obiect NULL.In loc sa verificam daca un obiect este null ,obiectulNull reflecta o relatie de a face nimic.

**5. Rezultate**

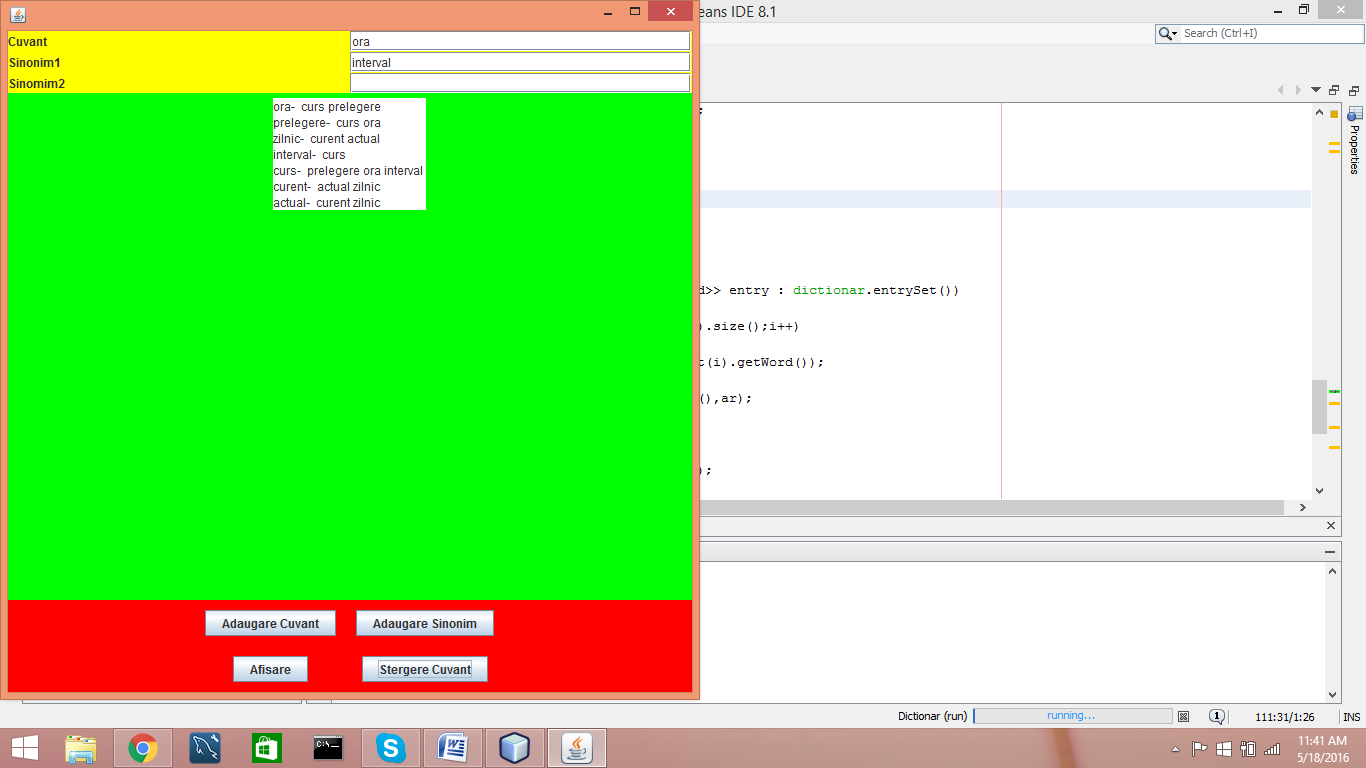
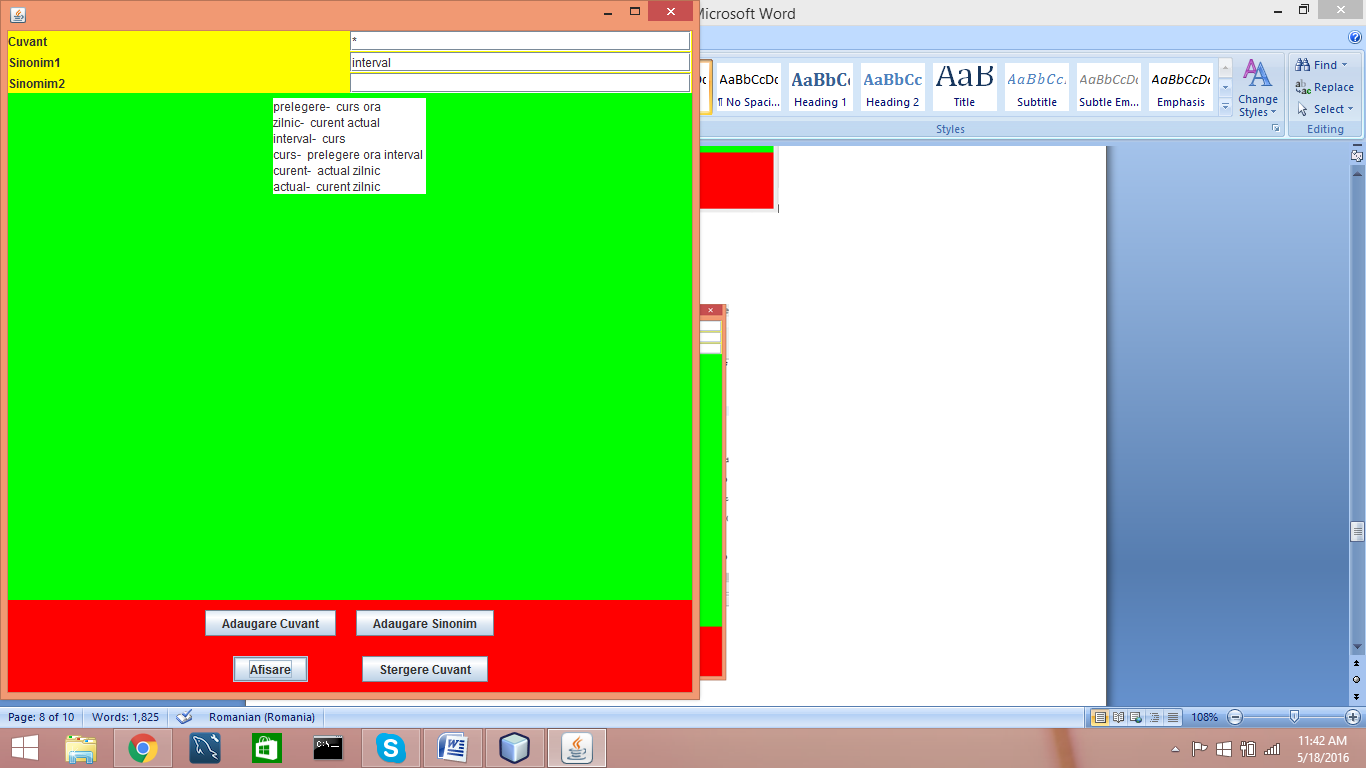
In acesata etapa se va testa functionalitatea aplicatiei pentru functiile pe care acestea trebuie sa le implementeze verificand astfel si corectitudinea algoritmilor implemnentati.

**Afisare map**

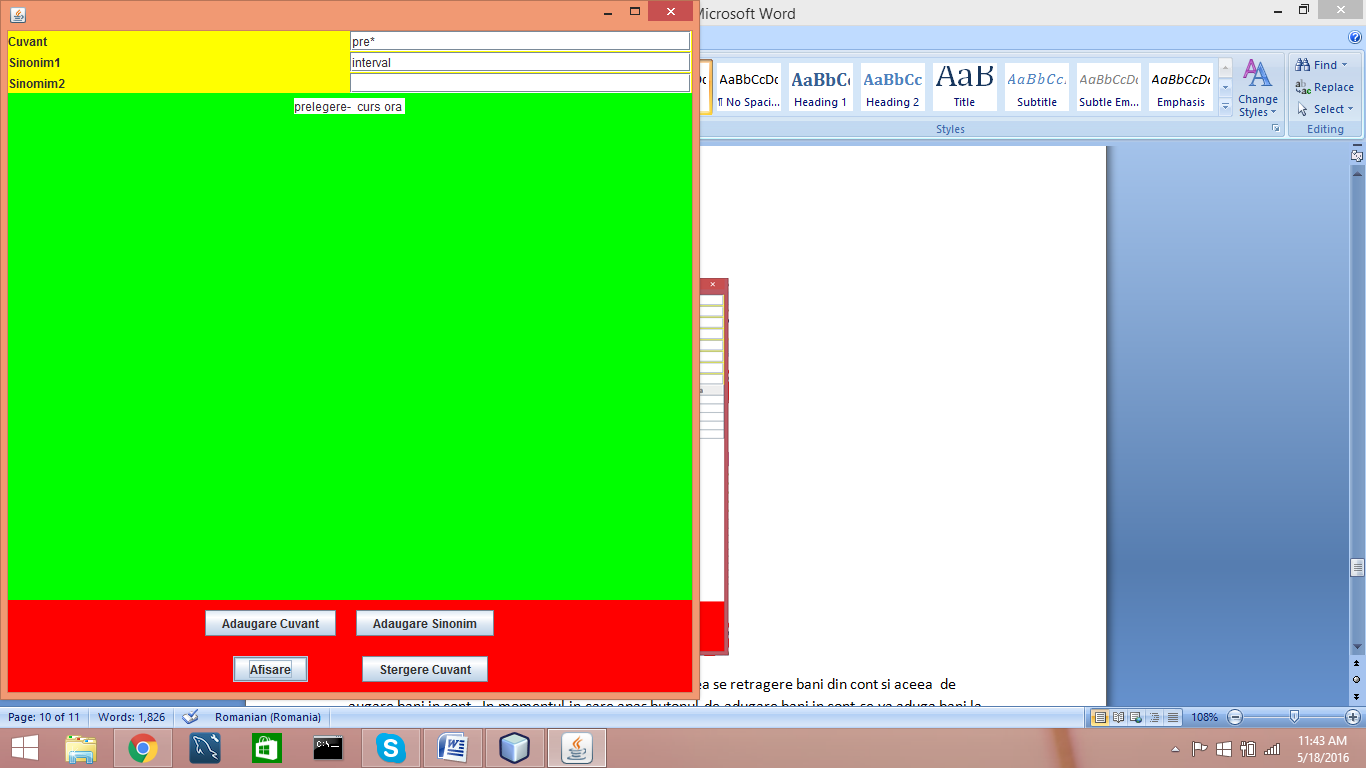
**Adugare sinonim**



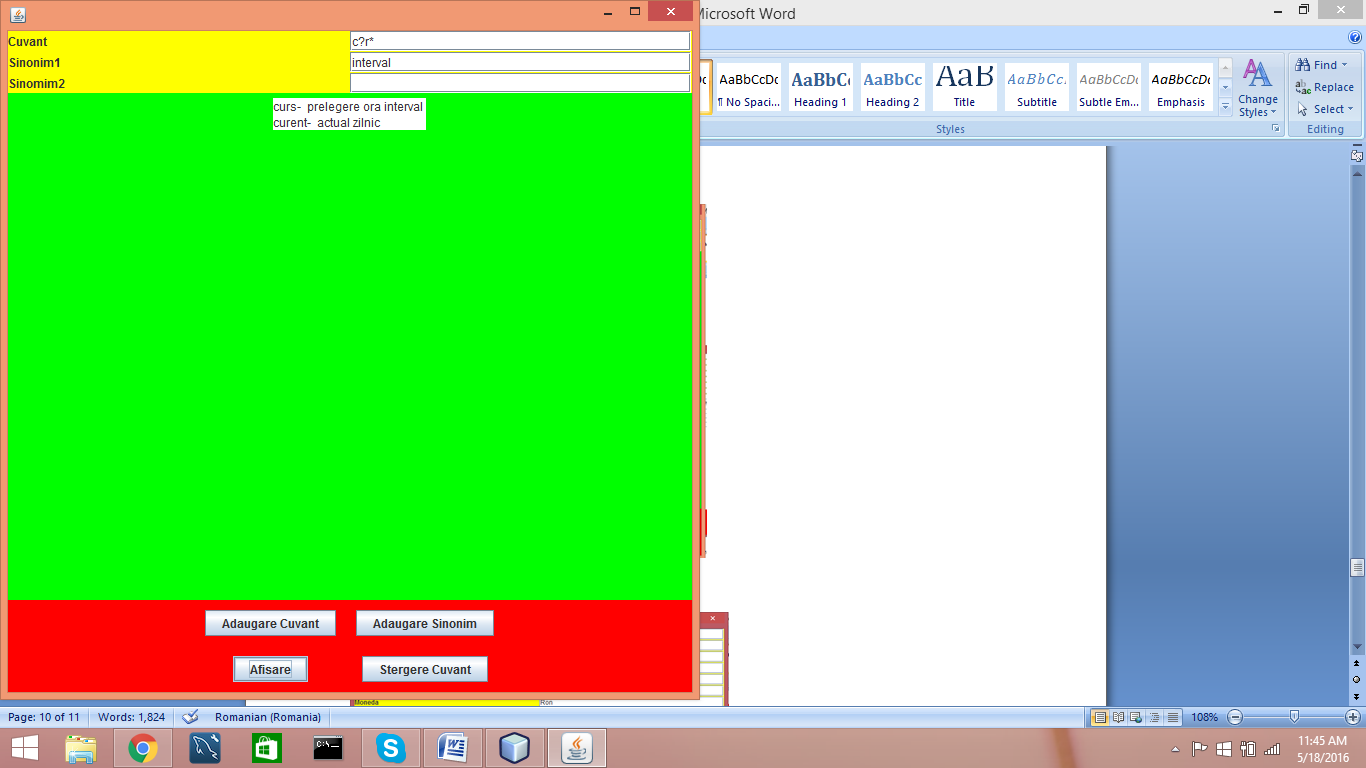
**Stergere cuvant**

**Afisare cu \***



**Afisare cu ?si \***



**6. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare**

Prin aceasta tema s-au evaluat conceptele de baza ale ale programarii orientate pe obiect concepte care m-au ajutat sa imi insusesc mai bine unele notiuni legate pe programarea orientata pe obiect.

Dezvoltari ulterioare:

* Modificarea interfetei pentru a o face mai atragatoare fata de client si adaugarea de obtiuni noi care sa ii satisfac clientului preferintele.
* Adaugarea de mai multe cuvinte si realizarea dictionarului cat mai reala.

**7. Bibliografie**

<http://www.json.org/>

<https://sites.google.com/site/gson/gson-user-guide>

<http://wiki.fasterxml.com/JacksonInFiveMinutes>

<http://www.tutorialspoint.com/design_pattern/>