**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Dicționar englez de sinonime  Tehnici de programare – Tema 5  5/9/2015  Bolba Raluca Maria, grupa 30225 |

### Cuprins

1. Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2
2. Analiza problemei, modelare, scenarii, cazuri de utilizare . . . . . . . . . . . . . . . . . . . . . . . . . .3
3. Proiectare (diagrame UML, structuri de date, proiectare clase, interfețe, relații, packages, algoritmi, interfață utilizator) . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4
4. Implementare și testare. . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
5. Rezultate . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
6. Concluzii, dezvoltări ulterioare . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
7. Bibliografie . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

### Obiectivul temei

Enunțul problemei :

1. Studiati Java Collection Framework

2. Considerati implementarea unuia din urmatoarele dictionare:

a. Un dictionar al limbii romane sau un dictionar al limbii engleze;

b. Un dictionar de sinonime pentru limba romana sau limba engleza;

Se cere folosirea Java Collection Framework pentru implementare .

Defineste si implementeaza interfata unui domeniu care sa realizeze operatiile: populeaza, adauga, sterge, copiaza, salveaza, cauta, etc. Considera implementarea unui program cu utilitati specifice pentru procesarea dictionarului . De exemplu:

- Implementeaza o metoda care verifica consistenta dictionarului. Un dictionar este consistent daca toate cuvintele care sunt folosite pentru definirea unui cuvant, sunt definite de asemenea in dictionar .

- Implementeaza o metoda de cautare pentru dictionar utilizand \* (orice sir, inclusiv null) si ? (un caracter). De exemplu se poate cauta a?t\*.

Utilizati de asemenea contracte, invarianti, preconditii si postconditii, asserturi, separati interfata de implementare si documentatie javadoc.

Am ales implementarea unui dicționar englez de sinonime, care oferă posibilitatea efectuării unor operații precum : adăugarea unui cuvânt și a sinonimului acestuia, căutarea unui cuvânt și posibilitatea de căuta un cuvânt specificând un anumit șablon, ștergerea unui cuvânt și a sinonimelor asociate, popularea dicționarului dintr-un fișier text, salvarea acestuia într-un fișier text și verificarea dacă dicționarul este consistent sau nu. De asemenea, aplicația ar trebui să ofere posibilitatea utilizatorului de a vizualiza întreg conținutul dicționarului.

### Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru o bună implementare a dicționarului englez de sinonime, trebuie mai întâi să definim conceptele folosite, să stabilim datele de intrare și cum sunt ele furnizate și procesate pentru a fi folosite în cadrul sistemului pentru a rezulta date de ieșire corecte.

Un dicționar, fie el de sinonime sau explicativ, este definit de un set de cuvinte, așezate într-o anumită ordine, cel mai frecvent alfabetic, și care conține pentru fiecare cuvânt o definiție sau o listă de sinonime. În cazul nostru, acest dictionar este implementat utilizând interfața Map, mai precis TreeMap, care constă în maparea unei valori la o cheie. Valoarea mapată va consta în cazul nostru într-o listă de sinonime. Am ales utilizarea clasei TreeMap întrucât impune o ordine alfabetică a cuvintelor.

Popularea dicționarului se face dintr-un fișier text, care conține pe fiecare linie un cuvânt, urmat de caracterul ”=” și de o listă de cuvinte despărțite prin virgule, reprezentând sinonimele primului cuvânt. Salvarea dicționarului într-un fișier va crea un fișier text cu aproximativ același format, singura diferență constând în faptul că lista de sinonime se află între paranteze drepte.

Un dicționar este consistent dacă sinonimele date pentru cuvinte, sunt și ele definite în cadrul dicționarului. Pentru a asigura această proprietate, la adăugarea unui set de cuvânt, sinonim, se va adăuga și perechea sinonim, cuvânt.

La căutarea unui cuvânt specificând un șablon, verificarea acestui șablon cu un cuvânt din cadrul dicționarului se va face în mod recursive, respective : dacă primele caractere din cele două cuvinte corespund, atunci se continua verificarea începând cu următoarele litere.

Interfața grafică cu utilizatorul trebuie să fie una simplă, intuitivă și care să permită utilizatorului să observe operațiile descrise mai sus.

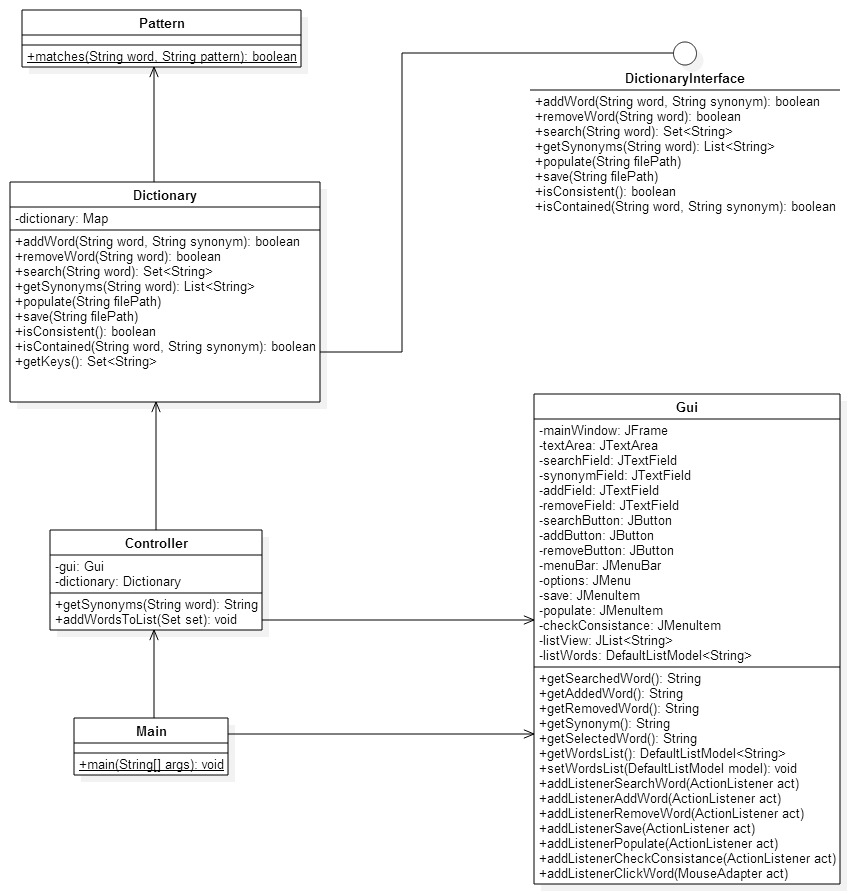
### Proiectare (diagrame UML, structuri de date, proiectare clase, interfețe, relații, packages, algoritmi, interfață utilizator)

#### Proiectare clase

Pentru implementarea acestei probleme am considerat necesare următoarele clase : Dictionary, DictionaryInterface, Pattern, Gui, Controller, Main.

* Interfața DictionaryInterface reprezintă șablonul sau contractul pe care un dicționar trebuie să îl respecte. Aceasta are definite metodele pe care clasele ce o implementează sunt constrânse să le definească.
* Clasa Dictionary implementează interfața DictionaryInterface și modelează conceptul de dicționar, în cazul nostru unul englez de sinonime. Obiectele acestei clase sunt instanțiate prin constructorul cu același nume, și sunt definite printr-un atribut de tip Map<String, List<String>> numit dictionary care mapează o listă de sinonime la un cuvânt. Inițial acest dictionary este vid, adăugarea elementelor și eliminarea acestora făcându-se prin intermediul unor metode
* Clasa Pattern modelează conceptul de șablon pentru un cuvânt. Clasa are o singură metodă definită static, utilă atunci când se realizează căutarea unui cuvânt în dicționar utilizând caractere precum ”\*” sau ”?”.
* Clasa Gui modelează conceptul de interfață grafică cu utilizatorul prin care un utilizator poate interacționa cu aplicația propriu-zisă. Ca bibliotecă de dezvoltare a interfeței grafice am folosit Swing.
* Clasa Controller modelează conceptul de controller care face legătură între partea grafică a sistemului și modelul acestuia. Printre atributele acestei clase se numără un obiect de tip Gui și instanța clasei Dictionary, între care se fac principalele legături.
* Clasa Main este clasa în care se pornește aplicația, prin instanțierea unor obiecte de tip Gui și Dictionary.

#### Diagrama UML

1. 
2. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
4. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
5. . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

#### Relațiile între clase

Pe baza diagramei UML se observă relațiile între clase .

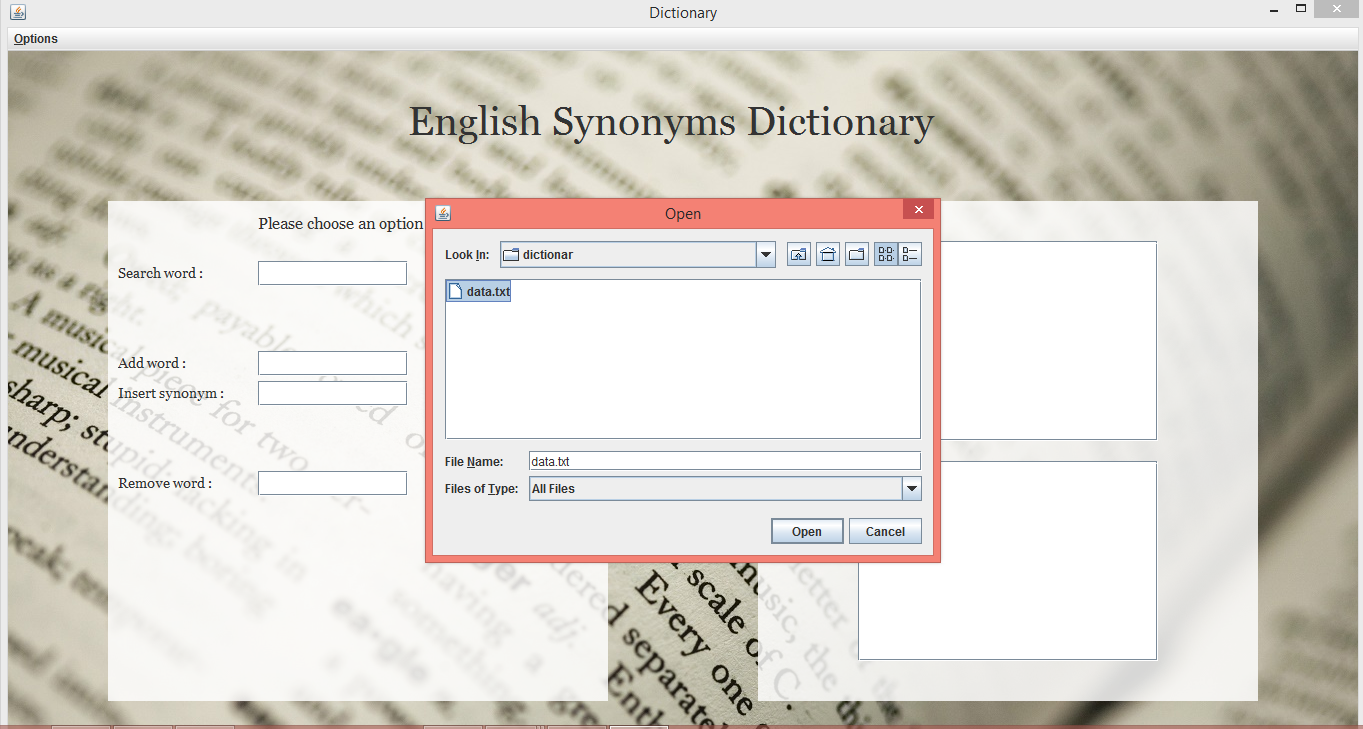
Între clasa Dictionary și Pattern, Controller și Dictionary, și de asemenea între Controller și Gui se observă relație de asociere care reprezintă cazul în care un obiect dintr-o clasă folosește serviciile unui obiect al celeilalte clase.

Între clasa Dictionary și DictionaryInterface se observă relația de implementare a interfeței, conform căreia clasa care implementează interfața este constrânsă să respecte șablonul descris de aceasta, prin implementarea metodelor definite.

#### Interfața grafică

Așa cum am mai menționat, interfața grafică cu utilizatorul am implementat-o cu ajutorul bibliotecii Swing. Principalele clase folosite în cadrul realizării acestei interfețe sunt : JButton, JMenuBar, JLabel, JTextField, JFrame , JdesktopPane, JList etc.

Fereastra de început a aplicației este împărțită, după cum se poate observa în două părți : în partea stângă observăm o serie de câmpuri pe care utilizatorul le poate completa dacă dorește să realizeze una din operațiile : căutare cuvânt, adăugare cuvânt sau ștergere cuvânt ; în partea dreapta se observă două chenare, cel de sus fiind o listă care afișează toate cuvintele din dicționar, iar cel de jos afișează sinonimele pentru cuvântul selectat din listă. De asemenea fereastra prezintă un meniu, din care utilizatorul poate alege să populeze dicționarul, să salveze dicționarul întru-un fișier text sau să verifice dacă dicționarul este consistent sau nu.

Dacă utilizatorul alege să populeze dicționarul, atunci se va deschide o fereastră rin care poate să aleagă fișierul din care să populeze dicționarul. În cazul în care fișierul selectat nu este unul text (cu extensia ”.txt”) atunci utilizatorul va fi atenționat cu un mesaj de eroare și va trebui să aleagă alt fișier. Aproximativ aceași fereastră va apărea și dacă acesta va alege să salveze dicționarul într-un fișier text, însă de data aceasta trebuie să specifice numele fișierului, nefiind nevoie să precizeze și extensia fișierului.

După popularea dicționarului, așa cum am mai precizat, în chenarul din dreapta sus va apărea o listă conținând toate cuvintele din dicționar. Dacă utilizatorul selectează un cuvânt din listă, în chenarul din dreapta jos va apărea lista de sinonime a cuvântului respectiv.

 La căutarea unui cuvânt sau a mai multor cuvinte prin specificarea unui șablon, rezultatul căutării va apărea tot în chenarul din partea dreaptă sus, după cum se poate observa mai jos :

### 4. Implementare și testare

### 4.1 Implementare

* Interfața DictionaryInterface modeleză conceptul de contract pe care clasele care o implementează trebuie să îl respecte. Metodele abstracte ale acesteia sunt :
  + **public** **boolean** addWord(String word, String synonym) **throws** Exception
  + **public** **boolean** removeWord(String word) **throws** Exception
  + **public** Set<String> search(String word)
  + **public** List<String> getSynonyms(String word)
  + **public** **void** populate(String filePath) **throws** FileNotFoundException, Exception
  + **public** **void** save(String filePath)
  + **public** **boolean** isConsistent()
  + **public** **boolean** isContained(String word, String synonym)
* Clasa Dictionary implementează interfața DictionaryInterface și modelează conceptul de dicționar . În cazul nostru este vorba de un dicționar englez de sinonime . Metodele acetei clase sunt :
  + **public** **boolean** addWord(String word, String synonym) **throws** Exception – metodă care adaugă un cuvânt împreună cu sinonimul acestuia la dicționar. În cazul în care dicționarul nu conține deja perechea de cuvinte, atunci se poate realiza adăgarea. Dacă cuvântul se află deja în dicționar, atunci se adaugă sinonimul la lista existentă a acestuia de sinonime. De asemenea, pentru a menține proprietatea de consistență, am ales ca la această operație să se adauge și perechea sinonim – cuvânt . Mai mult decât atât, pentru sinonim se adaugă și lista existentă de sinonime a cuvântului. Metoda returnează true dacă adăugarea a putut avea loc, false în caz contrar.
  + **public** **boolean** removeWord(String word) **throws** Exception – metodă care elimină un cuvânt din dicționar, împreună cu lista de sinonime asociată. Pentru început se verifică dacă cuvântul există în cadrul dicționarului, în caz contrar nu se poate elimina și se aruncă o excepție. Metoda returnează true dacă cuvântul a putut fi eliminat, false în caz contrar. În urma operației de ștergere dicționarul poate fi inconsistent, întrucât nu se șterge și sinonimul cuvântului.

* + **public** Set<String> search(String word) **throws** IllegalArgumentException – metodă care returnează un set de cuvinte ce se potrivesc cu cuvântul căutat. Această metodă este în special utilă atunci când utilizatorul caută anumite cuvinte folosind un șablon. Pentru început se verifică dacă dicționarul conține cuvântul respectiv, iar în caz afirmativ se returnează. În caz contrar se parcurge întreg dicționarul și se apelează metoda statică a clasei Pattern prin care se determină dacă cuvântul transmis ca pametru și cuvântul din dicționar se potrivesc. Toate cuvintele care se potrivesc cu șablonul specificat de utilizator vor fi returnate. În cazul în care utilizatorul introduce un șablon ce conține pe două poziții ”\*”, se va genera o excepție.
  + **public** List<String> getSynonyms(String word) – metodă care returnează lista de sinonime a cuvântului transmis ca parametru. În cazul în care dicționarul cu conține cuvântul respectiv, se returnează null.
  + **public** **void** populate(String filePath) **throws** FileNotFoundException, Exception – metodă care populează dicționarul, preluând lista de cuvinte împreună cu lista de sinonime asociate fiecărui cuvânt din fișierul a cărui cale a fost transmisă ca parametru. În cazul în care fișierul nu are extensia de fișier text, fișierul nu are formatul potrivit sau în cazul în care nu există fișierul din care se dorește să se citească, se aruncă o excepție. Dacă nu se aruncă aceste excepții, atunci se deschide fișierul pentru citire și se citește fiecare linie din cadrul acestuia. Linia este mai apoi împărțită în cuvinte, și se adaugă perechile corespunzătoare de cuvinte și sinonime. La sfârșit se închide fișierul.
  + **public** **void** save(String filePath) – metodă care salvează dicționarul într-un fișier text având calea transmisă prin parametru. Dacă numele fișierului nu conține extensia ”.txt” atunci aceasta se adaugă. Fiecare linie din fișier va conține un cuvânt, urmat de caracterul ”=” și de lista de sinonime despărțite prin virgulă.
  + **public** **boolean** isConsistent()– metodă care verifică dacă dicționarul este consistent. Un dicționar este consistent dacă sinonimele date pentru cuvinte, sunt și ele definite în cadrul dicționarului. Pentru a determina acest lucru se parcurg toate cheile dicționarului și listele asociate acestor cuvinte. Dacă vreun element din lista de sinonime nu aparține setului de chei a dicționarului, se returnează false.
  + **public** **boolean** isContained(String word, String synonym) – metodă care verifică dacă dicționarul conține perechea cuvânt – sinonim transmisă ca parametru. Metoda returnează true dacă dicționarul conține perechea respectivă, sau false în caz contrar.
  + **public** Set<String> getKeys()– metodă care returnează setul de cuvinte definite din cadrul dicționarului.
* Clasa Pattern modelează conceptul de șablon pentru un cuvânt. Singura metodă a acestei clase, statică de altfel, este :
  + **public** **static** **boolean** matches(String word, String pattern) **throws** IllegalArgumentException – metodă care verifică dacă cuvântul și șablonul transmis ca parametru se potrivesc. Această metodă este utilizată atunci când utilizatorul caută anumite cuvinte folosind un șablon, prin utilizarea caracterelor ”\*” sau ”?”. Pentru început se verifică dacă cuvântul este null sau are lungimea 0, sau dacă șablonul este null sau are lungimea 0 și cuvântul nu începe cu caracterul ”\*”, cazuri în care cele două șiruri de caractere nu se potrivesc. De asemenea dacă cuvântul conține caracterul ”\*” pe poziții consecutive, se aruncă o excepție. Dacă aceste cazuri nu au validate, atunci se determină în ce caz suntem : cuvântul începe cu ”\*” sau cu ”?”. În primul caz, trebuie să verificăm dacă subșirul cuvântului ce urmează după caracterul curent se potrivește cu vreunul din subșirurile șablonului, prin apelare recursivă. Dacă suntem în cazul al doilea, când primul caracter al cuvântului este ”?” sau corespunde cu primul caracter din șablon, apelăm recursiv pentru subșirurile celor două șiruri de caractere ce încep de la poziția următoare. Metoda returnează true dacă cele două șiruri de caractere se potrivesc, false în caz contrar.
* Clasa Gui modelează conceptul de interfață grafică cu utilizatorul. Printre metodele acestei clase se numără :
  + **public** String getSearchedWord()– metodă care returnează un șir de caractere reprezentând cuvântul pe care utilizatorul dorește să îl caute în dictionar.
  + **public** String getAddedWord()– metodă care returnează un șir de caractere reprezentând cuvântul pe care utilizatorul dorește să îl adauge în dicționar.
  + **public** String getSynonym()– metodă care returnează un șir de caractere reprezentând sinonimul pe care utilizatorul dorește să îl adauge pentru un cuvânt în dictionar.
  + **public** String getRemovedWord()– metodă care returnează un șir de caractere reprezentând cuvântul pe care utilizatorul dorește să îl șteargă din dicționar.
  + **public** String getSelectedWord()– metodă care returnează un șir de caractere reprezentând cuvântul pe care utilizatorul îl selectează din lista de cuvinte definite în dicționar.
  + **public** **void** setSynonym(String synonyms) – metodă care setează chenarul din partea dreaptă jos a ferestrei de început a aplicației cu lista de sinonime a cuvântului selectat din lista de cuvinte existente în dicționar.
  + **public** DefaultListModel<String> getWordsList()– metodă care returnează modelul listei ce va conține cuvintele definite în dicționar.
  + **public** **void** setWordsList(DefaultListModel<String> list) – metodă care setează modelul listei ce va conține cuvintele definite în dicționar cu modelul primit ca parametru.
* Clasa Controller modelează conceptul de controller care face legătură între partea grafică a sistemului și modelul acestuia. Metodele acestei clase sunt :
  + **public** String getSynonyms(String word) – metodă care returnează un șir de caractere conținând sinonimele cuvântului transmis ca parametru. Șirul de caractere conține pe fiecare linie câte un sinonim.
  + **public** **void** addWordsToList(Set<String> set) – metodă care adaugă setul de cuvinte în lista din cadrul interfeței grafice, pentru a putea fi vizualizate de utilizator.

### 4.2 Testare

În cadrul acestui proiect, testarea a constat prin specificarea și verificarea pentru fiecare metodă a unor precondiții și postcondiții pentru clasele proiectului.

Specificarea precondițiilor și a postcondițiilor s-a făcut la începutul fiecărei metode, într-un comentariu bloc. Verificarea acestora s-a făcut în cadrul metodei, prin intermediul aserțiunilor. Precondițiile au fost verificate la începutul metodei, iar postcondițiile la sfârșitul acesteia, înainte de return .

De asemenea, în cazul în care introduce date invalide, este atenționat prin mesaje de eroare corespunzătoare. Spre exemplu, dacă utilizatorul dorește să șteargă un cuvânt care nu există în dicționar, va primi un mesaj de eroare de forma „xxxxx does not exist in dictionary.”



### 5. Rezultate

În urma rulării aplicației se observă că aceasta rulează fără probleme. Se mai observă că atunci când utilizatorul introduce date de intrare invalide acesta este atenționat cu mesaje de eroare corespunzătoare.

### 6. Concluzii, dezvoltări ulterioare

În concluzie, se poate spune că am reușit implementarea unui dicționar de sinonime. În cadrul acestei teme am reușit să acumulez cunoștințe în ceea ce privește colecțiile, utilizarea expresiilor regulate precum și lucrul cu fișiere text.

Dezvoltări ulterioare :

* dezvoltarea dicționarului astfel încât acesta să ofere și definiții pentru cuvinte, pronunție și etimologie, nu doar sinonime
* modificarea interfeței grafice

### 7. Bibliografie

<https://docs.oracle.com/javase/tutorial/collections/interfaces/map.html>

<https://docs.oracle.com/javase/tutorial/essential/io/file.html>

<http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>