Facultatea de Automatică și Calculatoare

**Documentație**

**Calculator de polinoame**

-Tema I-

Nume student: Ivan Alexandra-Ștefania

Grupa: Grupa 30227

**CUPRINS**

[1. Obiectivul temei 3](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297885)

* 1. Obiectiv principal …………………………………………...3
  2. Obiective secundare ………………………………………...3

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 4](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297886)

[3. Proiectare 6](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297887)

3.1 Diagrama UML de clase ……………………………………...6

3.2 Structuri de date folosite ……………………………………...7

[4. Implementare 8](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297888)

4.1 Clasa Monom …………………………………………………8

4.2 Clasa Polinom ………………………………………………...9

4.3 Clasa Operații …………………………………………………9

4.4 Clasa View …………………………………………..………12

4.5 Clasa Controller …………………………………..…………13

[5. Rezultate 14](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297889)

[6. Concluzii 21](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297890)

[7. Bibliografie 22](file:///C:\Users\Stefi\Downloads\PT2023_Template_Documentatie_RO.doc#_Toc95297891)

1. **Obiectivul temei** 
   1. **Obiectivul principal** al temei constă în implementarea unui calculator polinomial care să aibă o interfață grafică dedicată, prin intermediul căreia utilizatorul să poată introduce polinoame, să poată selecta operația matematică dorită(adunare, scădere, înmulțire, împărțire, derivare sau integrare) și să poată vedea polinomul rezultat.
   2. **Obiectivele secundare** reprezintă mai mulți pași intermediari care vor conduce la îndeplinirea obiectivului principal.Aceștia sunt prezentați în tabelul de mai jos:

|  |  |
| --- | --- |
| 1. | Folosirea conceptelor programării orientate pe obiecte |
| 2. | Folosirea unui Map pentru modelarea clasei Polinom |
| 3. | Folosirea *foreach* înlocul for-ului clasic |
| 4. | Implementarea unei interfețe grafice pentru utilizator folosind Java Swing sau JavaFX. |
| 5. | Implementarea adunării și scăderii |
| 6. | Implementarea claselor cu maxim 300 de linii și a metodelor cu maxim 30 de linii |
| 7. | Folosirea convențiilor Java pentru denumirea pachetelor, claselor și metodelor |
| 8. | Buna organizare a codului sursă |
| 9. | Implementarea operației de înmulțire |
| 10. | Implementarea operației de împărțire |
| 11. | Implementarea operației de derivare |
| 12. | Implementarea operației de integrare |
| 13. | Folosirea expresiilor regulate pentru extragerea coeficientului și a exponentului |
| 14. | Folosirea JUnit pentru testare |

1. **Analiza problemei**

Analiza problemei pentru proiectarea unui calculator de polinoame include

numeroase aspecte.

Problema este definită în felul următor:

* Înțelegerea matematică a conceptelor prezentate și implementarea acestora

prin intermediul unei interfețe grafice care sa constituie un calculator de polinoame funcțional.

***Ce este un polinom?***

**In matematică, un polinom este o expresie construită dintr-una sau mai multe variabile și constante, folosind doar operații de adunare, scădere, înmulțire și ridicare la putere constantă pozitivă întreagă.**

**Polinoamele sunt construite din termeni numiți monoame, care sunt alcătuite dintr-o constantă (numită coeficient) înmulțită cu una sau mai multe variabile. Fiecare variabilă poate avea un exponent constant întreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile în acel monom. Coeficientul unui monom poate fi orice număr, inclusiv fracții, numere iraționale sau negative.**

**Cerințe funcționale**:

* Calculatorul polinom ar trebui să permită utlizatorului introducerea polinoamelor
* Calculatorul polinom ar trebui să permită utlizatorului să selecteze operația matematică dorită
* Calculatorul polinom ar trebui să adune două polinoame
* Calculatorul polinom ar trebui să scadă două polinoame
* Calculatorul polinom ar trebui să înmulțească două polinoame
* Calculatorul polinom ar trebui să împartă două polinoame
* Calculatorul polinom ar trebui să poată integra un polinom
* Calculatorul polinom ar trebui să poată afișa rezultatul unui calcul într-un format ușor de înțeles

**Cerințe non-funcționale:**

* Calculatorul polinom ar trebui să fie intuitiv și ușor de utilizat de către utilizator
* Calculatorul polinom ar trebui să aibă o interfață de utilizator receptivă, cu intrare și ieșire rapidă și precisă
* Calculatorul polinom ar trebui să fie capabil să gestioneze polinoame mari și calcule complexe rapid și eficient
* Calculatorul polinom ar trebui să fie precis și fiabil, cu erori minime în calcul
* Calculatorul polinom ar trebui să fie sigur, protejând datele utilizatorului Calculatorul polinomial ar trebui să urmeze bune practici și standarde de codare, cum ar fi buna organizare a codului și mentenabilitatea
* Calculatorul polinom ar trebui să fie adaptabil la nevoile și cerințele în schimbare ale utilizatorilor
* Calculatorul polinom ar trebui să aibă o documentație bună, cu manuale de utilizare clare și concise și specificații tehnice
* Calculatorul polinom ar trebui testat, cu un plan de testare care să acopere diferite scenarii și cazuri de utilizare.

1. **Proiectare** 
   1. **Diagrama UML de clase**

UML este un limbaj de modelare bazat pe notaţii grafice folosit pentru a

specifica, vizualiza, construi şi documenta componentele unui program. UML este un limbaj cu ajutorul căruia se pot construi (descrie) modele. Un model surprinde un anumit aspect al unui program şi acelaşi model poate fi descris la diferite nivele de abstractizare. Fiecărui model îi corespunde o diagramă.

Diagrama UML de clase a proiectului este prezentată în imaginea de mai

jos:

A screenshot of a computer

Description automatically generated with medium confidence

Între clasele implementate se poate observa prezența unor diferite relații. **O relaţie** modelează o conexiune semantică sau o interacţiune între elementele pe care le conectează.

-Relația de **compoziție** între clasa Controller și clasa View

* Compoziţia este o formă mai puternică a agregării.
* Relaţia de agregare este o asociere ce modelează o relaţie parte-întreg.
* Partea are „timpul de viaţă” al întregului.
* Întregul poate avea responsabilitatea directă pentru crearea sau distrugerea părţii sau poate accepta o altă parte creată şi mai apoi să paseze „responsabilitatea” altui întreg.

Astfel, clasa Controller este responsabilă în mod direct pentru creearea clasei View. Spunem că View este parte a clasei Controller.

-Relația de **dependență** între clasa Operations și clasa Polynomial și între clasa Monomial și ea însăși

* O dependenţă indică o relaţie semantică între două elemente ale modelului.
* Se foloseşte în situaţia în care o schimbare în elementul destinaţie poate atrage după sine o schimbare în elementul sursă.

Clasa Operations „folosește” clasa Polynomial pentru realizarea operațiilor

matematice implementate.

* 1. **Structuri de date folosite**

O structură de date este o metodă sistematică de stocare a informațiilor și

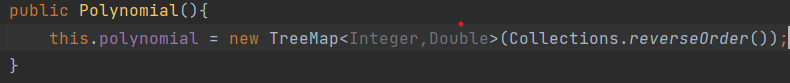
datelor într-un calculator, în așa fel încât ele să poată fi folosite în mod eficient. Deseori o alegere bine făcută a structurii de date va permite și implementarea unui algoritm eficient.

În implementarea proiectului meu, am folosit ca și structură de date, un „Tree

Map”, în cadrul clasei Polynomial, pentru stocarea valorilor coeficienților și puterilor pe care le va conține un polinom.

**O structură de date TreeMap** este o colecție care stochează perechi cheie-

valoare. Clasa TreeMap păstrează datele într-o structură de arbore folosind o interfață Map. Această clasă extinde clasa AbstractMap, și asemenea clasei părinte, TreeMap are două tipuri de parametri. Unul dintre parametri reprezintă cheile din TreeMap, iar celălalt reprezintă valorile. În mod implicit, TreeMap sortează toate intrările în funcție de ordinea lor naturală. Pentru un număr întreg, aceasta ar însemna ordine crescătoare, iar pentru șiruri, ordine alfabetică. Astfel încât ne dorim ca monoamele ce alcătuiesc un polinom să fie afișate în ordinea descrescătoare a valorii exponentului, atunci când creăm obiectul TreeMap trebuie să transmitem metoda **Collections.reverseOrder()** constructorului TreeMap. Metoda Collections.reverseOrder() indică TreeMap-ului să sorteze cheile în funcție de ordinea inversă.



1. **Implementare** 
   1. **Clasa Monom**

Clasa Monom modelează structura unui monom. Această clasă are două variabile instanță: coeficientul, de tip double și puterea, de tip întreg. În momentul în care se creează un monom nou, acesta primește ca argumente, în cadrul constructorului, coeficientul și puterea.

Metodele importante implementate în clasa Monom sunt următoarele:

-metoda getCoefficient() și metoda getPower() – acestea ne permit să preluăm valoarea coeficientului și a puterii, întrucât am folosit modificatorul de acces **private** la declararealor.

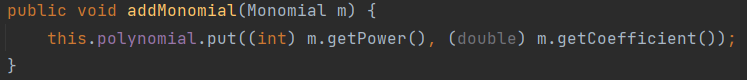
-metoda setCoefficient() și metoda setPower() – acestea ne permit să atribuim valori noi coeficientului și puterii.

Prin intermediul acestor metode, am evidențiat utilizarea unei paradigme de bază a programării orientate pe obiecte, și anume, **încapsularea**.

* 1. **Clasa Polinom**

**Clasa Polinom** implementează o reprezentare fizică a unui polinom, asupra căruia se vor putea efectua operații matematice de adunare, scădere, înmulțire, împărțire, derivare, integrare.

Mai exact, un obiect de tip Polynomial stochează în structura de date TreeMap mai multe obiecte de tip Monomial. Acest lucru se realizează prin intermediul metodei addMonomial, care adaugă un monom la un polinom:



Clasa Polinom conține și alte metode, precum:

* setPolynomial(TreeMap<Integer,Double> polynomial) – care permite atribuirea unei noi valori unui polinom;
* getPolynomial() – care permite preluarea valorii unui anumit polinom;
* int degree() – această metodă returnează gradul unui polinom;
* Double firstNonZeroCoefficient() – returnează valoarea primului coeficient nenul dintr-un polinom;
* Integer firstNonZeroPower() – returnează valoarea puterii primului termen al unui polinom care are un coeficient nenul;
* String toString() – afișează polinomul într-o manieră de forma



* equals() – metoda verifică dacă obiectul dat ca parametru este identic cu obiectul curent, apoi dacă obiectul dat ca parametru este o instanță a clasei Polynomial, iar în cele din urmă compară reprezentările sub formă de șiruri de caractere ale obiectului dat ca parametru și a obiectului curent;
  1. **Clasa Operații**

**Clasa Operații** implementează funcționalitatea operațiilor matematice de

adunare, scădere, înmulțire, împărțire, derivare, integrare.

**Adunarea -** Această metodă implementează operația de adunare a două

polinoame p1 și p2 și returnează un nou obiect Polynomial care reprezintă suma acestor polinoame. Metoda creează un nou obiect Polynomial care are aceeași reprezentare ca și polinomul p2 primit ca parametru. Acest lucru se face pentru a asigura faptul că noul obiect rezultat va conține toți termenii din polinomul p2. Apoi se creează un obiect TreeMap numit auxiliar, care conține aceeași reprezentare ca și polinomul p2. Acest obiect va fi folosit ulterior pentru a restabili reprezentarea inițială a polinomului p2, deoarece aceasta va fi modificată în timpul adunării. Mai întâi, metoda iterează prin toți termenii din polinomul p1. Pentru fiecare termen, se extrage puterea și coeficientul corespunzător, și apoi se verifică dacă termenul respectiv se găsește deja în polinomul rezultat. Dacă acest lucru este adevărat, se adună coeficientul curent cu coeficientul corespunzător din polinomul rezultat și se adaugă termenul rezultat în polinomul rezultat. Dacă termenul nu se găsește deja în polinomul rezultat, se adaugă direct termenul curent în polinomul rezultat. După ce toți termenii din polinomul p1 au fost adăugați la polinomul rezultat, metoda trece prin toți termenii din polinomul p2 și adaugă fiecare termen la polinomul rezultat folosind metoda addMonomial(). La final, reprezentarea polinomului p2 este restabilită folosind obiectul auxiliar. În cele din urmă, metoda returnează obiectul „result” care reprezintă suma polinoamelor p1 și p2.

**Scăderea** – Această metodă funcționează după aceeași idee pe care se

bazează și operația de adunare, cu excepția faptului că, termenii polinomului rezultat, care primește valoarea polinomului p2, sunt înmulțiți cu -1. În continuare, se parcurg aceeași pași ca și în metoda precedentă. La final, metoda returnează polinomul „result”, care reprezintă diferența polinoamelor p1 și p2.

**Înmulțirea** - Metoda iterează prin toți termenii din polinomul p1. Pentru

fiecare termen, se creează un nou obiect Polynomial numit „intermediateResult” care va conține rezultatul înmulțirii acestui termen cu toți termenii din polinomul p2. În continuare, metoda iterează prin toți termenii din polinomul p2. Pentru fiecare termen, se calculează puterea și coeficientul rezultat din înmulțirea termenilor din p1 și p2. Apoi, un nou obiect Monomial este creat cu coeficientul și puterea calculate și este adăugat la obiectul intermediateResult folosind metoda addMonomial(). După ce toți termenii din polinomul p2 au fost înmulțiți cu termenul curent din polinomul p1, se adaugă intermediateResult la result folosind metoda addition(), care realizează operația de adunare a două polinoame. După ce toți termenii din polinomul p1 au fost înmulțiți cu toți termenii din polinomul p2, reprezentarea polinomului p2 este restabilită folosind obiectul auxiliar. În cele din urmă, metoda returnează obiectul result care reprezintă produsul polinoamelor p1 și p2.

**Împarțirea –** funcționează pe baza următorului algoritm

**Text, letter

Description automatically generated**

Această metodă implementează algoritmul de împărțire a două polinoame p1

și p2 și returnează câtul și restul împărțirii ca un vector de două polinoame. Mai întâi, se inițializează polinoamele q, r și t. Se verifică dacă gradul primului polinom (p1) este mai mare decât gradul celui de-al doilea polinom (p2) cu ajutorul metodei **degree()**, deoarece împărțirea se poate face doar atunci când gradul p1 este mai mare sau cel puțin egal cu gradul p2. În caz contrar, este aruncată o excepție "Division can not be performed!". Se calculează coeficientul și puterea termenului t pentru a fi adăugat la câtul q prin împărțirea primului termen nenul din r la primul termen nenul din p2. Apoi, se actualizează q prin adăugarea termenului t și se actualizează restul prin scăderea dintre r și produsul dintre t și p2. Acest proces continuă până când gradul lui r devine mai mic decât gradul lui p2 sau r devine gol. La final, se returnează vectorul ans care conține polinoamele q și r.

**Derivarea –** Metoda „derivation” primește ca parametru un obiect de tipul

Polynomial și returnează un nou obiect de tipul Polynomial care reprezintă derivata polinomului dat ca parametru. Metoda parcurge fiecare termen din polinomul dat ca parametru și calculează coeficientul și puterea noului termen rezultat din derivare. Dacă puterea este mai mare sau egală cu 0, atunci noul termen este adăugat la polinomul rezultat. Mai precis, se înmulțește coeficientul termenului curent cu puterea sa și se scade 1 din putere pentru a obține noua putere.

**Integrarea -** Metoda „integration” primește ca parametru un obiect de tipul

Polynomial și returnează un nou obiect de tipul Polynomial care reprezintă integrala polinomului dat ca parametru. Metoda parcurge fiecare termen din polinomul dat ca parametru și calculează coeficientul și puterea noului termen rezultat din integrare. Se înmulțește coeficientul termenului curent cu 1 împărțit la puterea sa plus 1 și se adaugă 1 la putere pentru a obține noua putere.

* 1. **Clasa View**

**Clasa View** definește interfața GUI pentru calculatorul polinomial. Interfața

conține mai multe panel-uri, care încorporează elemente esențiale în ceea ce privește funcționalitatea proiectului.

**Graphical user interface, application

Description automatically generated**

În primul rând, interfața conține un titlu „Polynomial Calculator”, 5 text

field-uri(doua dintre ele sunt alocate introducerii celor două polinoame asupra cărora se vor efectua operațiile, iar restul pentru furnizarea rezultatelor), 2 căsuțe check-box care vor selecta polinomul care va fi primit ca parametru în cazul derivării și integrării, un buton de „Reset”, care șterge toate datele și alte 6 butoane, care permit selectarea operației matematice dorite.

Aspectul interfeței este definit folosindu-se cateva obiecte JPanel, ce

folosesc diferite manière de afișare a componentelor. În cazul panoului pentru butoane se folosește **GridLayout**, pentru panoul final **BoxLayout**, iar pentru restul panourilor **FlowLayout**.

Metodele addAdition, addSubtraction, addMultiplication etc. adaugă un

obiect de tip „ActionListener” butoanelor corespunzătoare, astfel încât să poată fi apelate metodele din acest obiect când butoanele sunt apăsate. Metodele „getP1Input” și „getP2Input” returnează valorile introduse de utilizator în câmpurile de text. Metodele updateResultLabel, updateQuotientLabel etc. actualizează valorile afișate în câmpurile de text cu noile valori primite ca parametru. Metoda „showNumberError” afișează o fereastră de dialog cu un mesaj de eroare dat de parametrul „errorMessage” în cazul în care valorile introduse în câmpurile de text nu sunt valide.

* 1. **Clasa Controller**

Clasa Controller primește datele de intrare de la View, le procesează cu

ajutorul unui model și apoi returnează rezultatul către interfață.

Controller primește ca și parametru în constructor un obiect de tip View, iar

apoi adaugă mai multe obiecte de tipul „ActionListener” pentru manipularea operațiilor. Fiecare ActionListener preia datele de intrare, introduse de utilizator, cu ajutorul metodelor getP1Input() și getP2Input() și folosește expresia regulată de forma ”([+-]?\d\*\.?\d\*)[a-z]\^(\d\*)” pentru a extrage coeficientul și puterea fiecărui monom. Un monom este împărțit în două grupuri. Coeficientului îi revine grupul 1, iar puterii grupul 2. Cu ajutorul grupurilor se creeaza noi obiecte de tip Monom, care se adaugă la obiecte de tip Polinom. În cadrul fiecarui ActionListener se efectuează cate una din cele 6 operații, iar rezultatul este returnat sub formă de String. Astfel, Controller-ul trimite rezultatul final înapoi către View, pentru a putea fi citit de către utilizator.

1. **Rezultate**

Pașii necesari în vederea obținerii unor rezultate sunt următorii:

-se introduce valoarea polinomului 1 în căsuța corespunzătoare din dreptul său;

-se introduce valoarea polinomului 2 în căsuța corespunzătoare din dreptul său;

-dupa introducerea valorilor aferente celor două polinoame, se apasă butonul corespunzător operației matematice dorite;

-în acest caz există mai multe scenarii: fie va apărea un mesaj de eroare care indică faptul că au fost introduse date invalide, fie nu se va întampla nimic, în cazul în care am ales operația de integrare sau derivare și nu am selectat checkbox-ul aferent polinomului asupra căruia dorim sa efectuăm acest proces, fie vom vedea rezultatul în căsuța „Result” (pentru adunare, scădere, înmulțire, derivare, integrare) sau în căsuțele „Quotient” și „Remainder”.

**Testarea Unitară**

• Este o metodă simplă si rapidă de testare a codului sursă de către programatori. • Are loc în faza de dezvoltare și este un instrument destinat programatorilor.

• Un unit test este o secvență de cod scrisă de un programator pentru a evalua o parte bine definită, de mici dimensiuni, din codul sursă testat – clasă sau metodă • Un unit test evaluează modul de funcționare al unei metode într-un context bine definit.

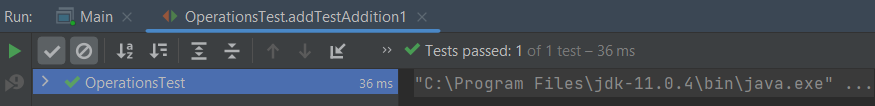
Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence Text

Description automatically generated

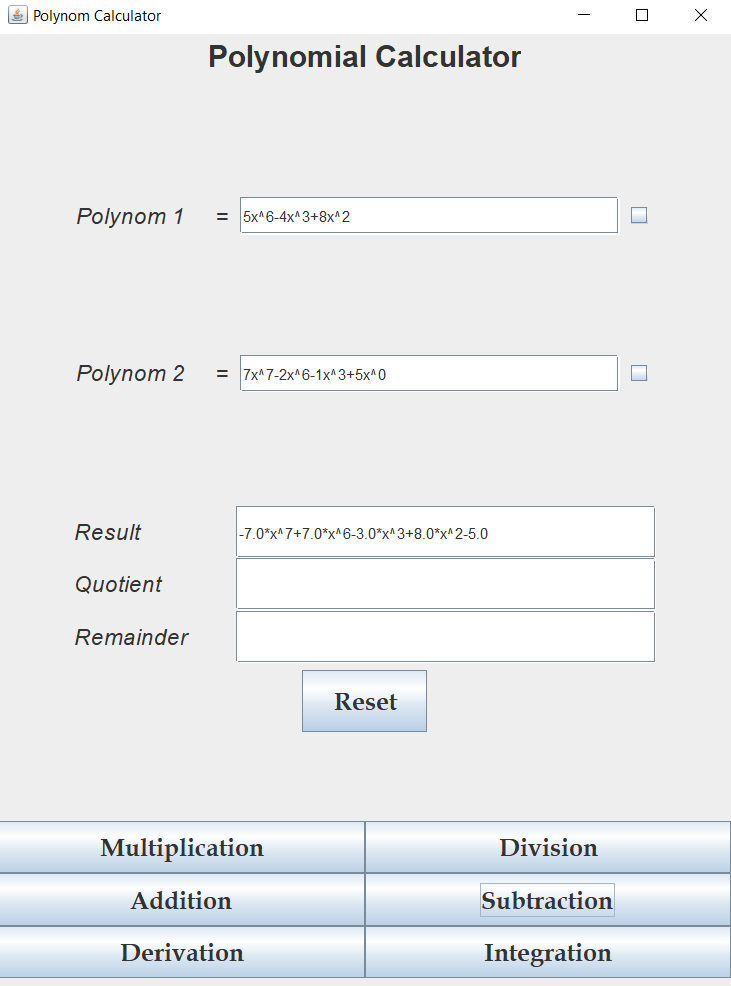


**Adunare**

Graphical user interface, application

Description automatically generated

**Scădere**



**Înmulțire**

Graphical user interface, application

Description automatically generated

**Derivare**

Graphical user interface, application

Description automatically generated

**Integrare**

Graphical user interface, application

Description automatically generated

**Împărțire**

Graphical user interface, application

Description automatically generated

1. **Concluzii**

În concluzie, proiectul a avut ca obiectiv principal dezvoltarea unei interfețe care să permită utilizatorului să efectueze operații matematice asupra polinoamelor. În timpul dezvoltării, am întâmpinat numeroase obstacole, însă procesul a fost cu atât mai satisfăcător pe măsură ce reușeam să descopăr soluții și noi idei. Parcurgerea acestei teme mi-a dat ocazia să aprofundez cunoștiințele mele în Java, să aflu noi informații, precum utilizarea expresiilor regulate și să pun la punct o interfață funcțională.

O primă dezvoltare ulterioară în ceea ce privește calculatorul de polinoame ar putea fi simplificarea introducerii datelor de către utilizator, astfel încât să nu fie necesară specificarea coeficientului „1” sau a puterii „x^0”. De asemenea, implementarea unor funcții de căutare a rădăcinilor polinomului ar putea reprezenta o altă modalitate de îmbunătățire a proiectului.

1. **Bibliografie**

<https://www.jetbrains.com/help/idea/class-diagram.html#analyze_class>

<https://cadredidactice.ub.ro/sorinpopa/files/2018/10/L1_diagrame_use_case.pdf>

<https://www.baeldung.com/java-treemap>

<https://ro.wikipedia.org/wiki/Structur%C4%83_de_date>

<https://www.javaprogramto.com/2021/01/java-traverse-treemap-reverse-order.html>

<https://regexr.com/>

<https://www.w3schools.com/java/java_regex.asp>

<https://www.javatpoint.com/java-swing>

<https://www.acs.ase.ro/Media/Default/documents/cts/curs/Curs%20CTS%20%20-%20Testare%20unitara%20cu%20JUnit.pdf>