**DOCUMENTATIE TEMA 1**

**CALCULATOR POLINOAME**

**Nume prenume: Pașca Maria**

**Grupa 30229**

**Profesor Laborator Assist Antal Marcel**

Contents

[1. Cerinte Functionale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv Principal: 3](#_Toc476131447)

[2.2. Obective Secundare: 3](#_Toc476131448)

[3. Analiza Problemei 3](#_Toc476131449)

[4. Proiectare 3](#_Toc476131450)

[4.1. Structuri de date 3](#_Toc476131451)

[4.2. Diagrama de clase 3](#_Toc476131452)

[4.3. Algoritmi 3](#_Toc476131453)

[5. Implementare 4](#_Toc476131454)

[6. Testare 4](#_Toc476131455)

[7. Concluzii si Dezvoltari Ulterioare 4](#_Toc476131456)

[8. Bibliografie 4](#_Toc476131457)

# Cerinte Functionale

Cerința a fost dezvoltarea unui calculator de polinoame care să permită operația de citire a uni polinom de la tastatură și realizarea operațiilor de adunare, scădere, înmulțire și împărțire a două polinoame, respectiv derivarea și integrarea unui polinom.

Pentru o bună funcționare a aplicației, polinoamele pot fi introduse atât cu semn de înmulțire între coeficient și x, cât și fără, însă și pentru x^1 trebuie introdusă puterea, altfel regex-ul nu va putea recunoaște corect termenul. Cu toate acestea, termenul liber nu necesită să fie însoțit de x^0. De asemenea, nu este indicată nici folosirea spațiilor între termenii polinomului, fie ei coeficienți și semnul înmulțirii sau între un monom și plus sau minus.

Pentru operațiile care necesită două polinoame și nu sunt comutative, anume în cazul scăderii și împărțirii, polinomul 1 reprezintă descăzutul, respectiv deîmpărțitul, iar al doilea scăzătorul/împărțitorul, iar pentru adunare și înmulțire, termenii operației pot fi adăugați oricum. În cazul derivării și integrării, calculele se vor face cu polinomul 1.

Pentru realizarea calculelor și afișarea rezultatului, se va da click pe butonul cu operația pe care dorim să o facem, iar rezultatul va veni afișat în dreptul labelului cu ”Result: ”.

# Obiective

## Obiectiv Principal:

Obiectivul principal este proiectarea și implementarea unui calculator polinomial cu o interfață grafică dedicată prin care utilizatorul poate insera polinoame și poate selecta operația matematică pe care vrea să o facă și să vadă rezultatul afișat în fereastră.

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii |  | 3 |
| Alegerea structurilor de date |  | 4 |
| Impartirea pe clase |  | 4 |
| Dezvoltarea algoritmilor |  | 4 |
| Implementarea solutiei |  | 5 |
| Testare |  | 6 |

# Analiza Problemei

Use Case Name: Adunarea a două polinoame

Actori:

* Utilizatorul
* View-ul
* Controller-ul
* Modelul

Trigger:

* Utilizatorul a dat click pe butonul de adunare

Precondition:

* Utilizatorul a introdus în câmpuri polinoamele

Normal Flow:

1. Utilizatorul introduce corect polinoamele în textFieldurile aferente.
2. Utilizatorul apasă butonul de adunare.
3. Listenerul butonului de adunare din controller dă mai departe comanda model-ului, cu tot cu cele două stringuri ca parametru.
4. În model, cele două stringuri sunt transformate în obiecte de tip Polynomial.
5. Din model este apelată metoda add pentru primul polinom, care are al doilea polinom ca parametru.
6. Din acea metodă este apelată metoda addition din operation.
7. Rezultatul este returnat din operation înapoi în metoda add din primul polinom.
8. De acolo este returnat addition din model.
9. Din model rezultatul este trimis în controller.
10. Din controller este trimis în view în labelul pentru rezultat.
11. Utilizatorul va vedea rezultatul calculului pe ecran.
12. Utilizatorul va închide programul.

Alternate Flow:

12A. Utilizatorul dorește să mai facă o operație.

1. Se reiau pașii de la 1.

Use Case Name: Derivarea unui polinom

Actori:

* Utilizatorul
* View-ul
* Controller-ul
* Modelul

Trigger:

* Utilizatorul a dat click pe butonul de derivare

Precondition:

* Utilizatorul a introdus în câmpul aferent (primul textField) polinomul

Normal Flow:

1. Utilizatorul introduce corect polinomul în primul textfield.
2. Utilizatorul apasă butonul de derivare.
3. Listenerul butonului de derivare din controller dă mai departe comanda model-ului, alături de polinom.
4. În model, stringul este transformat în obiecte de tip Polynomial.
5. Din model este apelată metoda deriv pentru polinom.
6. Din acea metodă este apelată metoda derivation din operation.
7. Rezultatul este returnat din operation înapoi în metoda deriv din polinom.
8. De acolo este returnat în derivation din model.
9. Din model rezultatul este trimis în controller.
10. Din controller este trimis în view în labelul pentru rezultat.
11. Utilizatorul va vedea rezultatul calculului pe ecran.
12. Utilizatorul va închide programul.

Alternate Flow:

12A. Utilizatorul dorește să mai facă o operație.

1. Se reiau pașii de la 1.

# Proiectare

## Structuri de date

Structura de date cea mai răspândită în proiect este Singly-LinkedList-ul, mai exact List<Monomial>, care sunt folosite pentru a ține listele de monoame din care sunt compuse polinoamele. Deși este cea mai răspândită structură de date din proiect, pentru a ține matchurile în regex am folosit un array de stringuri, deoarece am folosit un toArray să nu mai parcurg cu un for match-urile regexului.

## Diagrama de clase

Diagram

Description automatically generated

# Implementare

Pentru acest proiect am ales să merg cu patternul architectural MVC (Model, View, Controller). Astfel, sunt 3 clase principale: Model, View și Controller. Pe urmă, pentru calcule se mai folosesc clasele Polynomial, Monomial și Operations. De asemenea, pentru a testa corectitudinea aplicației, am folosit un JUnit.

În clasa Model, String-urile citite de la tastatură prin interfață vor fi prelucrate și transformate în Polinoame, astfel că și obiectele prezente sunt Polynomial (polynomial1 și polynomial2). Aceasta conține metoda createPolynomial, care ia un String, în acest caz polinomul din textField-ul polinomului citit, și folosește pattern recognition (regex) pentru a împărți polinomul, prima dată în monoame, ca pe urmă să separe coeficientul de putere (în variabila denumită rank, deoarece reprezintă rangul monomului în cauză). Pe baza informației extrase se construiește monomul, care este adăugat în lista de monoame a polinomului care urmează să fie returnat. Această metodă este private fiindcă este folosită doar în clasa Model și nu vrem să fie apelată din exterior. În afară de această metodă, clasa mai are și metodele addition, subtraction, multiplication, division, derivative și integration, prin care se apelează metodele respective din clasa Polynomial, dar care returnează polinomul rezultat în urma calculelor sub formă de String și care urmează în Controller să fie atribuit labelului de output.

Clasa Controller este cea care se controlează și clasa Model și clasa View astfel: Listener-ul fiecărui buton este prezent în această clasă și tot în această clasă îi sunt trimise obiectului model polinoamele citite din textBoxurile prezente în interfață și îi este zis ce să facă cu ele. În această clasă sunt definite Listener-urile pentru fiecare buton, din care se trimit modelului polinoamele citite și se apelează metoda specifică fiecărui buton, ca mai apoi rezultatul obținut să fie trimis view-ului, care îl pune în labelul pentru output.

Clasa Monomial, după cum îi spune și numele, este folosită pentru monoame. Aceasta conține câmpurile coef, în care e păstrat coeficientul, rank, care păstrează puterea termenului, respectiv rangul coeficientului, și un Boolean isBrowsed, prin care la adunare și scădere ne asigurăm că, în cazul în care unul dintre polinoame are un termen de un rang care nu apare în celălalt polinom, se va adăuga la rezultatul final. Pe lângă getters & setters pentru fiecare câmpuri și constructori, în clasă se mai găsesc și metodele add, sub, mul, div, deriv, integr, care, după cum le spune și numele, sunt folosite pentru operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare. Primele 4 metode au ca parametru un al doilea monom cu care se fac calculele, iar toate returnează rezultatul ca un nou monom.

Clasa Polynomial are ca și câmpuri, rank, care e de tip int și care stochează rangul polinomului, monomials, care e o listă de obiecte Monomial, care reprezintă colecția de monoame din care e format polinomul, și operation, de clasă Operations, clasă în care sunt păstrate toate metodele pentru calcule și unde se întâmplă efectiv tot. Metodele folosite în clasă, în afară de constructori și getters & setters, sunt metodele add, sub, mul, div deriv și integr. Acestea sunt folosite pentru a apela metodele addition, subtraction, multiplication, derivation și integration din operation. De asemenea, aceasta mai are și metodele getMaxMonomial, care returnează monomul de rang maxim din polinom, și metoda toString, care returnează polinomul ca String.

În clasa Operations au loc toate calculele cu polinoame. Aceasta cuprinde metodele addition, subtraction, multiplication, division, derivation și integration, dar și checkMonomials, care se asigură că în lista de monoame nou formată nu există monoame care să aibă coeficientul 0. Metoda addition folosește un for care trece prin toate monoamele ale primului polinom, care mai are înăuntru un for care parcurge toate monoamele celui de-al doilea polinom și, dacă au același rang, le adună și le marchează ca parcurse. Astfel, dacă după parcurgeri rămân elemente care nu au fost folosite în adunare, să le adauge la lista de monoame, deoarece înseamnă că nu a fost un monom în celălalt polinom cu același rang cu care să facă adunarea. Metoda subtraction este dezvoltată în același mod ca metoda addition, singura diferență fiind faptul că, în cazul monoamelor neparcuse din polinomul 2, acestea sunt adăugate la noul polinom cu semn schimbat. Metoda multiplication are la bază tot un for (pentru monoamele din primul polinom) în for (pentru monoamele din al doilea polinom), în care monoamele sunt înmulțite, iar monomul nou obținut este comparat cu monoamele obținute anterior și, dacă unul dintre ele are același rang, să îl adauge la rezultatul anterior. Atât metoda derivative, cât și integration se folosesc de un for, care parcurge lista de monoame ale polinomului și calculează derivata/integrala fiecăreia și o adaugă în lista de monoame a noului polinom construit.

Clasa View este folosită pentru realizarea interfaței GUI cu Java Swing. În realizarea ei s-au folosit un frame, 5 paneluri, 6 butoane, 2 text field-uri pentru a scrie polinoamele care urmează să fie folosite în operații și 4 label-uri, dintre care unul este pentru a afișa rezultatul operației. Metodele prezente în această clasă sunt getInput1 și getInput2, care returnează conținutul text field-urilor, adică polinoamele introduse de utilizator, setTxtOutput, prin care labelul pentru output primește rezultatul operației, și metodele prin care butoanelor le sunt atribuite listener-urile.

Cele 6 butoane sunt folosite pentru selectarea operației matematice pe care utilizatorul dorește să o facă și au listener-urile în clasa Controller.

Panelurile input1 și input2 sunt folosite pentru a păstra labelurile și textfieldurile pentru introducerea polinoamelor, ambele fiind pe urmă încadrate într-un panel cu GridLayout numit panelInput. Următorul panel ține butoanele pentru operații, iar ultimul este folosit pentru labelurile pentru output, cel în care se păstrează stringul rezultat și cel care precizează că în primul e rezultatul.

Frame-ul desemnează fereastra aplicației, în care sunt adăugate cele 5 paneluri cu toate elementele aferente și care, la finalul constructorului din View, este făcută vizibilă.

Obiectele celor trei clase, Model, View și Controller sunt instanțiate în clasa MVC (ModelViewController), în care se află si metoda main a programului.

Clasa TestPolynomialCalculator este cea care în care se folosește JUnit pentru testarea programului. În cadrul constructorului acestei clase se construiesc doi polinomi cu care se vor face calculele, care ulterior vor fi verificate prin folosirea de assertEquals, care testează dacă rezultatul obținut în urma calculelor făcute cu cei doi polinomi este același cu rezultatul pe care îl așteptam. În cazul în care rezultatele nu sunt aceleași, programul va semnala eroarea în consolă.

Toate clasele, mai puțin cele ce țin de UI, au până în 300 de linii, iar metodele până în 30 de linii.

# Testare

Pentru JUnit se folosește assertEquals, care verifică dacă este egalitate între rezultatul obținut și cel pe care îl așteptam. Acesta trece pe rând prin fiecare assert și dacă ajunge la unul care e false, programul dă eroare și se oprește. Astfel, prima dată merge în assertEquals pentru adunare, iar dacă totul merge bine continuă și merge în assertEquals pentru scădere, pe urmă merge în assertEquals pentru înmulțire, pe urmă în derivare, iar la final în integrare.

# Concluzii si Dezvoltari Ulterioare

Programul reușește să citească polinoamele pe care le introduce utilizatorul și să facă operațiile de adunare, scădere, înmulțire derivare și integrare, însă fără operația de împărțire, care undeva are o greșeală pe care încă nu am reușit să o găsesc.

Dezvoltările pe care le văd să fie făcute în viitor sunt, în primul rând, crearea unui regex de recunoaștere a polinomului, respectiv a monoamelor, astfel încât utilizatorul să le poată adăuga oricum vrea, cu spații fără spații, fără x la puterea 1, adică să fie cât mai apropiat de scrierea naturală. De asemenea, tot pe scrierea naturală, văd ca string-ul de output să fie făcut astfel încât, în cazul în care monomul de rang maxim are un coeficient pozitiv, acesta să nu înceapă cu + și coeficientul, ci direct cu coeficientul. În plus, dacă polinomul rezultat are termen liber, acesta să nu mai vină însoțit de x^0, iar în cazul monoamelor de rangul 1, să afișeze x simplu, fără ^1.

În plus, pe viitor văd și realizarea unui exception handler care să sesizeze dacă polinomul introdus de la tastatură nu respectă normele, spre exemplu dacă are două monoame de același rang să dea eroare sau dacă unul dintre monoame are coeficientul 0 să avertizeze utilizatorul.

În afară de aceste aspecte tehnice, prin acest proiect am reușit să stăpânesc mai bine limbajul Java și crearea programelor prin implementarea paradigmei MVC, totodată învățând și ce este și cum se folosește regex, care va fi foarte util și în proiectele viitoare.

# Bibliografie

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

<https://www.regular-expressions.info/refquick.html>

<http://tynerblain.com/blog/2007/04/09/sample-use-case-example/>