Documentație tema 1

Calculator de polinoame

Sigoiu George

Grupa 30224

Cuprins

[1. Obiectivul temei 3](#_Toc67004935)

[a) Obiectivul principal 3](#_Toc67004936)

[b) Obiective secundare 3](#_Toc67004937)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 4](#_Toc67004938)

[a) Cerințe funcționale 4](#_Toc67004939)

[b) Descriere cazuri de utilizare 4](#_Toc67004940)

[ Use-case: adunare, scădere, înmulțire, împărțire 4](#_Toc67004941)

[ Use-case: derivare, integrare 5](#_Toc67004942)

[3. Proiectare 6](#_Toc67004943)

[a) Diagrama UML 6](#_Toc67004944)

[b) Structuri de date 6](#_Toc67004945)

[c) Algoritmi 6](#_Toc67004946)

[d) Interfața utilizator 7](#_Toc67004947)

[4. Implementare 8](#_Toc67004948)

[a) Clasa Monomial 8](#_Toc67004949)

[b) Clasa Polynomial 8](#_Toc67004950)

[c) Clasa Operator 8](#_Toc67004951)

[d) Clasa Model 10](#_Toc67004952)

[e) Interfața grafică (clasa View și clasa Controller) 10](#_Toc67004953)

[5. Rezultate 12](#_Toc67004954)

[6. Concluzii 13](#_Toc67004955)

[7. Bibliografie 14](#_Toc67004956)

# Obiectivul temei

## Obiectivul principal

Obiectivul principal al proiectului este acela de a crea o aplicație care sa implementeze o interfață grafică pentru calculul cu polinoame. Un polinom este alcătuit din mai multe monoame. Un monom conține două constante, una de tip double „coeficient” și una de tip int „exponent”. Un monom are forma: coeficient \* x ^ exponent.

## Obiective secundare

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Determinarea operațiilor | Descrierea operațiilor care trebuie implementate | [2](#_Cerințe_funcționale) |
| Dezvoltarea de use-case-uri și scenarii | Într-un sistem software un ‘use-case’ este o listă de acțiuni sau pași de eveniment care definesc în mod obișnuit interacțiunile dintre un rol ( cunoscut ca un actor în limbajul unificat de modelare ( UML ) ) și un sistem in atingerea unui obiectiv.  În cazul nostru, rolul este de a utiliza aplicația. | [2](#_Descriere_cazuri_de) |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capăt obiectivul principal | [3](#_Structuri_de_date) |
| Interfața utilizator | Se descrie modul cum utilizatorul poate folosi aplicația | [3](#_Interfața_utilizator) |
| Împărțirea pe clase | Crearea mai multor clase pentru a ușura lucrul, clase pentru stocarea datelor, clase pentru efectuarea operațiilor, clase pentru interfața grafică | [4](#_Implementare) |
| Implementarea algoritmilor | Descrierea algoritmilor necesari efectuării operațiilor între polinoame | [4](#_Clasa_Operator) |
| Interfața grafică | Implementarea unei interfețe grafice care să afișeze rezultatele operațiilor cu polinoame | [4](#_Interfața_grafică_(clasa) |
| Testare | Vor fi descrise câteva scenarii de testare a operaților pe polinoame, folosind ca instrument un Junit Test. | [5](#_Rezultate) |

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## Cerințe funcționale

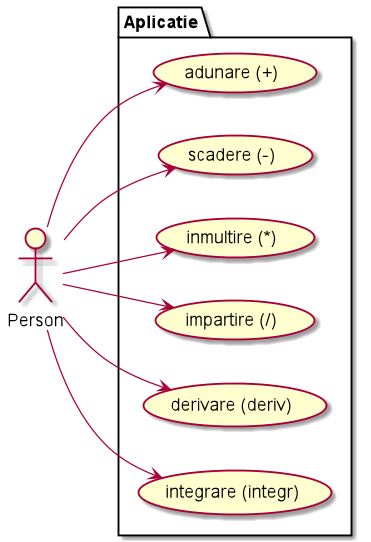
Operațiile propriu zise cu polinoame sunt următoarele:

1. Citirea a unui polinom sau a două polinoame de la tastatură

!Atenție: polinoamele trebuie să aibă următoarea forma: , sunt numere întregi. Pot lipsi termeni, de exemplu 3x^2+1 sau x^5-x^2 sunt forme acceptate.

1. Polinomul trebuie scris fără spații, de exemplu 3 x + 2 sau 3 x^2 +x-1 nu sunt forme corecte.
2. Adunarea a două polinoame:
3. Scăderea a două polinoame:
4. Înmulțirea între două polinoame:
5. Împărțirea unui polinom cu alt polinom de grad mai mic sau egal:
6. Obținerea unui polinom prin derivarea unui polinom:
7. Obținerea unui polinom prin integrarea unui polinom:

## Descriere cazuri de utilizare



### Use-case: adunare, scădere, înmulțire, împărțire

Main success scenario:

1. Utilizatorul (Person din diagrama use-case) introduce polinoamele in cele 2 text field-uri .
2. Utilizatorul alege una dintre cele 4 operații (adunare, scădere, înmulțire, împărțire) apăsând pe butonul respectiv.
3. Aplicația analizează cele 2 polinoame introduse.
4. Aplicația execută operația aleasă de utilizator.
5. Aplicația afișează rezultatul în ultimul text field.

Alternative sequences:

1. S-a introdus un polinom sub o formă greșită

* Aplicația va afișa un pop-up în care va fi explicat cum trebuie introdus polinomul
* Se repetă de la pasul 1

1. S-a introdus polinomul 2 cu cel mai mare grad mai mare decât cel mai mare grad al polinomului 1 și s-a ales operația de împărțire

* Aplicația va afișa un pop-up în care va fi specificat că polinomul 1 trebuie să aibă gradul maxim mai mare decât gradul maxim al polinomului 2
* Se repetă pasul 1

1. S-a introdus 0 pentru polinomul 2 și s-a ales împărțirea ca operație

* Aplicația va afișa un mesaj de eroare de tip pop-up
* Se repetă pasul 1

1. S-a ales ca operație împărțirea și nu s-a specificat polinomul 2 (polinomul împărțitor)

* Aplicația va afișa un mesaj de eroare de tip pop-up
* Se repetă pasul 1

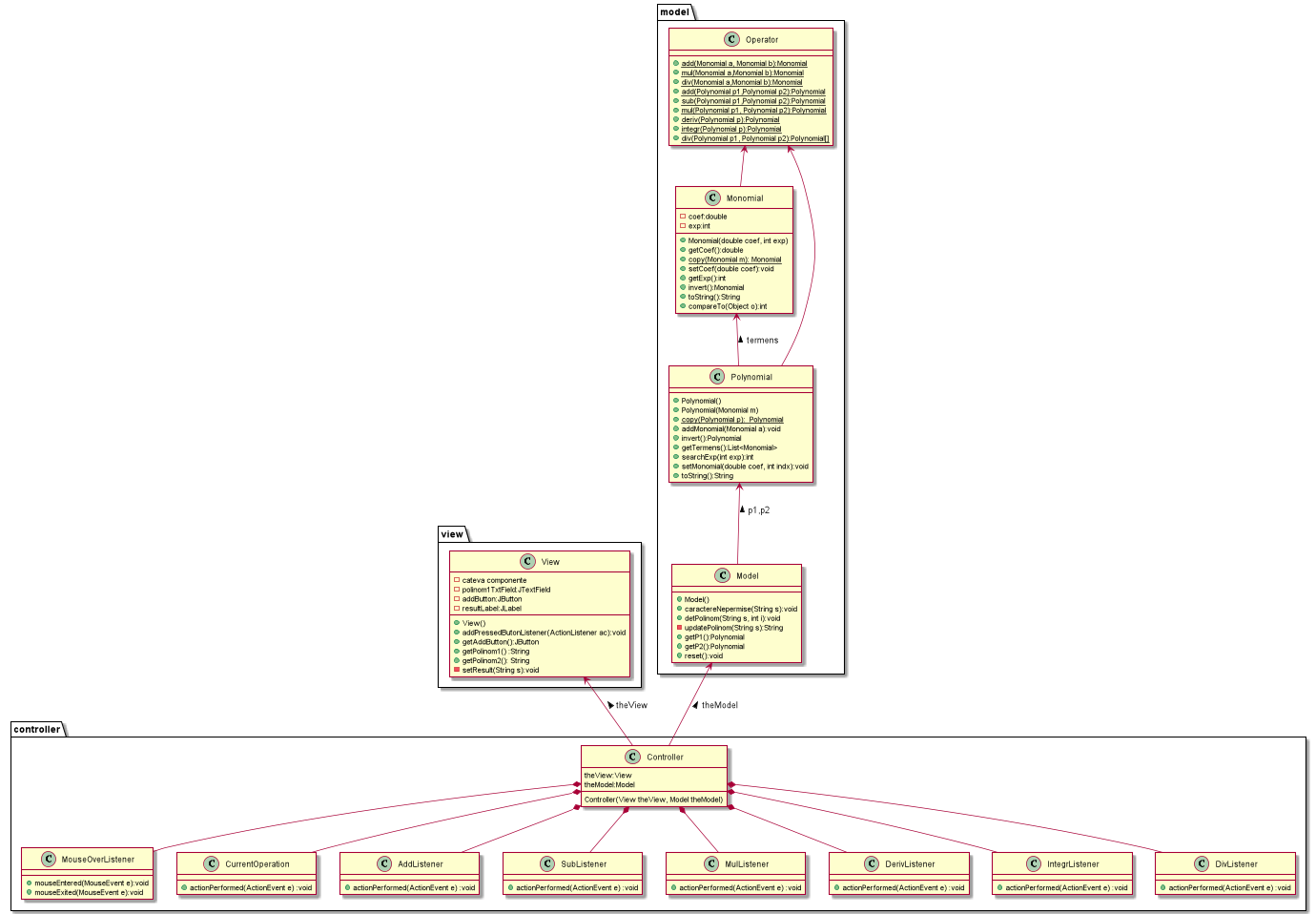
### Use-case: derivare, integrare

Main success scenario:

1. Utilizatorul (Person din diagrama use-case) introduce polinoamele in cele 2 text field-uri .
2. Utilizatorul alege una dintre cele 2 operații (derivare, integrare) apăsând pe butonul respectiv.
3. Aplicația analizează polinomul 1, doar pe polinomul 1 se va efectua operația aleasă
4. Aplicația execută operația aleasă de utilizator.
5. Aplicația afișează rezultatul în ultimul text field.

# Proiectare

## Diagrama UML



## Structuri de date

Ca și structuri de date am folosit o structură Polynomial, definită de mine care conține un ArrayList de Monomial (deoarece un polinom este format din mai multe monoame, un arraylist ar fi fost cel mai potrivit in rezolvarea acestei probleme). Monomial este o structură definită de mine care conține ca variabile instanță un număr zecimal (double) „coeficient” și un număr întreg (int) pozitiv „exponent”. „Coeficient” este de tip double doarece în urma efectuării operației de integrare există posibilitatea ca polinomul rezultat să aibă coeficienți numere zecimale.

## Algoritmi

Am folosit algoritmul de sort din Collections pentru a sorta polinoamele după gradul fiecărui monom. De exemplu, după efectuarea operației de adunare sau înmulțire pe polinoame, polinomul rezultat ar fi avut, la afișare, ordinea monoamelor aleatorie, adică dacă adunăm x^4+x cu x^3-3x^2, rezultatul ar fi fost afișat astfel: x^4+x+x^3-3x^2, iar după sortare: x^4+x^3-3x^2+x, ceea ce este mai ușor de urmărit. Pentru a implementa această sortare a fost nevoie de a suprascrie metoda compareTo astfel încât monoamele să fie ordonate în ordine descrescătoare în funcție de gradul acestora.

## Interfața utilizator



Interfața utilizator conține 3 text field-uri (doar primele 2 sunt editabile), 3 label-uri și 6 butoane.

În primele 2 text field-uri utilizator introduce cele 2 polinoame care trebuie să fie de forma: , unde sunt numere întregi. Polinoamele trebuie scrise folosind doar caracterele: +, -, x, ^ și numere întregi, fără spații între termenii polinomului. Fiecare buton reprezintă una dintre operațiile implementate, astfel butonul „+” reprezintă adunarea dintre 2 polinoame, „ - ” – scăderea, „ \* ”- înmulțirea, „ / ”- împărțirea, „deriv” – derivarea polinomului 1, „integr”- integrarea polinomului 1. Pentru primele 4 operații trebuie specificate ambele polinoame, iar pentru derivare și integrare trebuie specificat polinomul 1. Rezultatul operației alese va apărea în ultimul text field.

# Implementare

## Clasa Monomial

Conține ca atribute un număr de tip double care semnifică coeficientul unui termen din polinom și un număr de tip int care semnifică exponentul termenului.

Metode:

* public static Monomial copy(Monomial m) – creează un monom nou cu același coeficient si exponent și îl returnează.
* public Monomial invert() – inversează semnul monomului și returnează un monom nou cu semnul invsersat

## Clasa Polynomial

Conține ca atribut o listă de monoame (List<Monomial>) .

Metode:

* public static Polynomial copy(Polynomial p) – copiază un polinom dat ca argument și returnează această copie.
* public Polynomial invert() – inversează semnul polinomului, adică transformă polinomul P(x) în -P(x).
* public int searchExp(int exp) – caută într-un polinom monomul care are exponentul egal cu exp și returnează poziția pe care se află acest monom sau -1 în caz că nu a fost găsit.

## Clasa Operator

Metode:

* public static Monomial add(Monomial a, Monomial b) – returnează un nou monom ce reprezintă suma dintre cele 2 monoame date ca argumente, dacă acestea au același grad, altfel returnează null.
* public static Monomial mul(Monomial a,Monomial b)- returnează un nou monom ce reprezintă produsul dintre cele 2 monoame date ca argumente.
* public static Monomial div(Monomial a,Monomial b)- returnează un nou monom ce reprezintă raportul dintre cele 2 monoame date ca argumente.
* public static Polynomial add(Polynomial p1,Polynomial p2)- returnează un nou polinom care reprezintă suma dintre cele 2 polinoame date ca argumente

Algoritm:

1. Se creează un nou polinom „pol”
2. Se parcurg elementele din p1 și din p2 (polinomul 1 si polinomul 2 ).
3. Dacă termenul curent din p1 are același grad cu termenul curent din p2, se adaugă un termen nou în pol cu gradul egal cu cel al termenilor și coeficient egal cu suma coeficienților celor 2 termeni
4. Se verifică dacă fiecare termen din p1 a fost adăugat în polinomul nou, dacă nu a fost adăugat se adaugă.
5. Se verifică dacă fiecare termen din p2 a fost adăugat în noul polinom, în caz contrar se adaugă termenul respectiv.

OBS: Pentru a verifica dacă un termen a fost adăugat în polinomul nou :

* Pentru termenii din p1, la fiecare iterație se actualizează variabila booleană „adaugat” dacă termenul respectiv are gradul egal cu unul dintre termenii lui p2, dacă „adaugat” este false atunci se adaugă în noul polinom
* Pentru termenii din p2, se parcurg din nou termenii și se verifică dacă fiecare termen din p2 are gradul egal cu unul dintre termenii lui p1, în acest caz înseamnă că deja a fost introdus, în caz contrar se adaugă
* public static Polynomial sub(Polynomial p1,Polynomial p2) – returnează un nou polinom care reprezintă diferența dintre polinoamele date ca argumente. Polinomul p2 se inversează (p2 își schimbă semnele, devine -p2) și se adună cu p1.
* public static Polynomial mul(Polynomial p1, Polynomial p2) - returnează un nou polinom care reprezintă produsul dintre polinoamele date ca argumente.

Algoritm pentru înmulțire:

1. Se creează un polinom nou „pol”.
2. Se parcurg termenii din p1 și p2.
3. Fiecare termen din p2 este înmulțit cu fiecare termen din p1.
4. După ce s-a obținut un termen nou din înmulțirea celor 2 termeni se caută în pol dacă mai există un termen cu același exponent, dacă da, atunci se adună termenul rezultat cu cel existent deja, altfel se adaugă termenul rezultat in pol.

* public static Polynomial deriv(Polynomial p) – returnează un polinom nou obținut prin derivarea polinomul dat ca argument.

Algoritm pentru derivare:

1. Se creează un polinom nou
2. Se parcurg termenii polinomului.
3. Se creează un termen nou cu atributele: și .
4. Dacă exponentul nou este negativ se setează pe 0.
5. Noul termen se adaugă în polinomul nou.

* public static Polynomial integr(Polynomial p) - returnează un polinom nou obținut prin integrarea polinomul dat ca argument.

Algoritm pentru integrare:

1. Se creează un polinom nou
2. Se parcurg termenii polinomului
3. Se creează un termen nou cu atributele: și .
4. Termenul nou creat se adaugă în polinomul nou.

* public static Polynomial[] div(Polynomial p1, Polynomial p2) – returnează un vector de 2 elemente de tip Polynomial în care se află câtul, respectiv restul împărțirii celor două polinoame date ca argumente.

Algoritm pentru împărțire:

1. Se creează un nou polinom Q care reprezintă câtul împărțirii.
2. Cât timp gradul maxim al polinomului p1 (deîmpărțitul) este mai mare decât gradul maxim al polinomului p2 (împărțitorul)
3. Se împarte cel mai mare monom din p1 la cel mai mare monom din p2
4. Rezultatul împărțirii se adaugă în Q
5. Se scade din p1 valoarea polionului (p2\*rezultatul de la punctul 3).
6. Se sare la pasul 2.
7. După ce se iasă din structura repetitivă, în Q vom avea câtul împărțirii iar în p1 vom avea restul. Cele 2 polinoame se introduc într-un vector de polinoame de 2 elemente și se returnează rezultatul obținut.

## Clasa Model

Conține ca atribute 2 polionoame.

Metode:

* private void caractereNepermise(String s) – aruncă una dintre cele 2 excepții (SpaceException, WrongFormatInputException) dacă string-ul dat ca parametru conține spații sau un set de caractere declarat în interiorul metodei.
* public void detPolinom(String s, int i)- transformă string-ul dat ca argument într-o structura Polynomial. Folosind regex se determină fiecare grup (termen) și se transformă într-o structură Monomial, care apoi este adăugat în polinomul p1 daca i=1, altfel este adăugat în polinomul p2.
* private String updatePolinom(String s) – returnează un string nou format prin adăugarea în string-ul dat ca argument a unor caractere „1”. String-ul dat ca argument reprezintă un polinom, iar acest string este parcurs și dacă între semnul de adunare (+) sau semnul de scădere (-) și „x” nu se află un număr sau dacă string-ul începe cu „x” atunci se va adăuga caracterul „1” între semn și x, respectiv se va pune un „1” în fața x-ului.

## Interfața grafică (clasa View și clasa Controller)

Clasa View reprezintă interfața grafică propriu zisă, iar clasa Controller descrie ce face fiecare buton din interfață.

**View** este contruită dintr-un panel principal care conține alte 4 panel-uri. Panelul principal este de tip BoxLayout, ceea ce înseamnă că cele 4 panel-uri vor fi așezate într-o coloană. Figura de mai jos arată cum sunt dispuse cele 4 panel-uri (chenarele roșu, albastru, portocaliu, verde reprezintă fiecare un panel).



Primul, al treilea și ultimul panel (cel roșu, cel portocaliu și cel verde) conțin un label(pe care scrie „Polinomul 1”, „Polinomul 2”, respectiv „Rezultat”) și un text field (în care utilizatorul să introducă polinomul respectiv, iar în ultimul text field se va afișa rezultatul după efectuarea operației alese), fiecare. Panel-ul al doilea conține 6 butoane care reprezintă operațiile posibile cu polinoame.

Metodele:

* public void addAddListener(ActionListener listener),
* public void addSubListener(ActionListener listener),
* public void addMulListener(ActionListener listener),
* public void addDerivListener(ActionListener listener),
* public void addIntegrListener(ActionListener listener),
* public void addDivListener(ActionListener listener)

adaugă fiecărui buton un ActionListener care descrie ce trebuie să facă butonul respectiv. Aceste ActionListener-uri sunt descrise în clasa Controller.

**Controller** conține 2 atribute de tip Model și View și face legătura între interfață grafică și ce se întâmplă în spatele interfeței grafice. Această clasă mai conține 8 clase membru care implementează ActionListener și MouseListener pe care apoi le atribuie butoanelor astfel încât fiecare buton, când este apăsat, să efectueze o anumită operație.

# Rezultate

Pentru a testa adunarea, scăderea și înmulțirea am luat în considerare următoarele cazuri:

* Dacă ambele polinoame au exact aceleași monoame (dacă au aceeași exponenți)
* Dacă polinoamele au exponenții complet diferiți
* Dacă polinoamele conțin și grade comune și grade diferite.

Pentru a testa derivata și integrata am luat în considerare cazurile:

* Fiecare monom din polinom să conțină coeficient diferit de 1 și grade diferite și diverse
* Fiecare monom din polinom să conțină coeficientul 1 și grade diverse
* Un polinom care este format doar dintr-un termen liber fără x.

Pentru a testa împărțirea am ales teste în care polinomul1 să aibă grad mai mare decât polinomul2.

# Concluzii

Ca o dezvoltare anterioară a programului s-ar putea crea o interfață grafică mai prietenoasă cu utilizatorul, în sensul introducerii unor imagini sau personalizarea butoanelor, a scrisului în text field-uri, precum și adăugarea posibilității rezolvării calculelor mai complexe de exemplu: (4x^2+3x)\*(6x+1)+(x^3-2x^2)\*(x+6), această secvență să fie introdusă într-un singur text field și după apăsarea unui buton să se calculeze expresia. În momentul de față ar trebui introduse primele 2 polinoame și înmulțite, apoi rezultatul trebuie ținut minte, apoi trebuie înmulțite ultimele 2 polinoame, iar acest rezultat trebuie adunat cu cel precedent. O altă dezvoltare ar fi introducerea unei zone în interfața grafică care să afișeze ultimele 5 operații efectuate și rezultatul acestora.

Realizând această temă am învățat să lucrez cu regex (cât de cât) și să împart proiectul (clasele java) în MVC (Model-View-Controller).

# Bibliografie

Regex

<https://regexr.com/>

<https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression>