Universitatea Tehnica din Cluj-Napoca

Martie 2021

Tehnici de Programare

*~ Calculator de polinoame ~*

Halas Denisa Izabela Grupa 30224

# Cuprins

**Specificatiile problemei**…………………………………………….………………...…3

**Analiza Problemei**………………………………………………………………..…......3

Operatii pe monoame………………………………………………………….4

Operatii pe polinoame…….…………………………………………………...4

Modelarea problemei……………..……………………………………….…..5

Use Case………………………………………………………………… ……5

**Design**……………………………………………………………………………………7

Diagrama de clase……………………………………………………………..7

Monom…………………………………………………………………………8

Polinom………………………………………………………………………...9

Operatii…………………………………………………………………………9

MVC……………………………………………………………………………9

**Testare**…………………………………………………………………………………..13

**Rezultate**………………………………………………………………………………...13

**Concluzii**…………………………………………………………………………………14

**Webografie**………………………………………………………………………………15

# Specificatiile problemei

Proiectați și implementați un calculator de polinoame cu o interfață grafică dedicată prin care utilizatorul poate insera polinoame, selecta operația matematică (adică adunare, scădere, înmulțire, divizare, derivată, integrare) care urmează să fie efectuată și vizualiza rezultatul.

# Analiza problemei

Pentru a înțelege pe deplin o astfel de problemă, matematica este instrumentul de bază care trebuie utilizat.

Prin urmare, este necesara cunoasterea definiției unui polinom matematic.

Un **polinom** este o expresie construită dintr-una sau mai multe  variabile și constante, folosind doar operații de adunare, scadere, inmultire și ridicare la putere constantă pozitivă întreagă.

Conceptul matematic implică examinarea unui polinom în forma sa algebrică astfel:

𝑃(𝑥) = 𝑎0 + 𝑎1𝑥1 + 𝑎2𝑥2 + ⋯ + 𝑎𝑛𝑥𝑛 = 𝑎𝑖𝑥𝑖

Specificația problemei implică faptul că coeficienții polinomului sunt numere întregi și că sunt dintr-o singură variabilă (matematic, putem defini polinoame cu mai multe variabile, deci această specificație este relevantă).

Un polinom valid pentru problema specificată ar fi:

𝑃(𝑥) = 𝑥4 + 2𝑥 +1

Se poate observa din exemplul anterior că polinomul are o variabilă notată x, mai mulți coeficienți întregi, cum ar fi 1, 2, 1 și puteri întregi nenegative, cum ar fi 4, 1, 0. În general, coeficienții pot lua diferite valori, cu mai multe cifre, asemenea puterilor.

Vom trata un polinom ca o lista de termeni algebrici, așa cum afirmă în definiția. Acești termeni algebrici se vor numi monomi.

Un **monom** este un termen algebric caracterizat prin 2 numere: un coeficient și o putere.

Este foarte important de specificat că un polinom poate fi compus din zero, unul sau mai multe monoame. Prin urmare, din această definiție se poate afirma că un singur monom poate fi considerat ca fiind un polinom.

Trebuie să menționăm câteva caracteristici importante ale unui polinom:

• Termen dominant – este monomul cu cea mai mare putere;

• Termen liber – este monomul cu puterea 0;

• Grad – este puterea termenului dominant;

• Polinom nul – este vpolinomul care conține 0 monomi.

Un avantaj al polinoamelor este că efectuarea de operații pe ele este o sarcină relativ de simplă, semnificativ mai ușoară decât efectuarea operațiilor pe alte funcții, cum ar fi exponențiale sau funcții trigonometrice.

Aceste operații au ca punct de plecare operațiile efectuate pe monomi. Cu toate acestea, operațiile efectuate pe monomi au unele condiții ce trebuie specificate in prealabil.

# Operatii pe monoame

1. Adăugare - cu mențiunea că este necesar ca cele 2 monoame să aibă aceeași putere.
2. Scădere - cu mențiunea că cele 2 monoame, de asemenea, este necesare sa aiba aceeași putere.
3. Înmulțirea - aici nu se impun condiții.
4. Împărțire - cu mentiunea ca divizorul trebuie să aibă un coeficient diferit de 0.
5. Derivarea - cu mentiunea ca derivarea termenului liber este egala cu 0
6. Integrarea

**! Notă**: Este foarte important de mentionat faptul că există 2 operații care sunt deosebite datorită „rezultatului” obtinut in urma efectuarii lor. Acestea sunt:

1. **Împărțire** - împărțirea a 2 monomii (precum și cea a 2 polinoame) poate schimba tipul matematic al coeficienților. Și anume, atunci când se calculează rezultatul împărțirii a 2 numere întregi, rezultatul poate fi un alt număr întreg sau un număr real.
2. **Integrare** – unde problemă similară impartirii, fiind posibil ca rezultatul sa fie unul real

𝑀𝑜𝑛𝑜𝑚 m1 = 𝑎𝑥𝑛, 𝑀𝑜𝑛𝑜𝑚 m2 = 𝑏𝑥𝑚 →

Acest fapt va ghida implementarea spre a utiliza coeficienti reali.

# Operatii pe polinoame

Pornind de la operațiile efectuate pe monomii, operațiile polinomiale sunt mai mult sau mai puțin o sarcină clară. Este vorba despre aceleasi 6 operatii mentionate mai sus, dar aplicate de data aceasta, la un numar mai mare decat un singur monom.

# Modelarea problemei

Modelarea unei probleme este procesul de împărțire a problemei in subprobleme care, teoretic, ar trebui să fie mai ușor de rezolvat.

În primul rând, ideea de a considera un polinom ca o listă de monoame este primul pas făcut pentru modelarea problemei. Monoamele sunt mai ușor de tratat decât polinoamele, deoarece sunt obiecte caracterizate exclusiv de un coeficient și o putere.

Dezvoltarea soluției începe cu găsirea de soluții pentru definirea monomiilor și efectuarea operațiilor de bază pe acestea. În virtutea definițiilor anterioare, a monomului și a polinomului, problema a fost modelată astfel încât un polinom sa fie văzut ca o listă de monoame. Un polinom conține doar acele monoame care au un coeficient diferit de zero.

Matematic, putem vedea un polinom în 2 moduri. De exemplu, reluând polinomul din secțiunea anterioară, îl putem vedea ca:

Varianta 1: 𝑃(𝑥) = 𝑥4 + 2𝑥 +1

Varianta 2: 𝑃(𝑥) = 𝑥4 + 0𝑥3 + 0𝑥2 +2𝑥 +1

Ar fi o abordare destul de ineficientă să luăm în considerare modelul unui polinom similar cu varianta 2, așa că am ales să consider că polinomul nu ia în considerare monomiile cu coeficientul 0. Această opțiune ajută, de asemenea, la definirea polinomului nul.

# Use Case

Icon

Description automatically generatedIcon

Description automatically generatedIcon

Description automatically generatedIcon

Description automatically generatedA close-up of a skeleton

Description automatically generated with medium confidence

Utilizatorul trebuie să interacționeze cu sistemul selectând o operație. Din acel moment, sistemul va decide dacă intrările pe care le-a furnizat sunt corecte și va efectua calculele necesare pentru a oferi utilizatorului un rezultat corect.

# Exemplu 1: Suma a 2 polinoame efectuata de catre utilizator:

**Scenariu de succes:**

• Utilizatorul lansează aplicația cu succes.

• Utilizatorul furnizează primul polinom tastând un șir în primul câmp de text furnizat de aplicație

• Utilizatorul furnizează al doilea polinom tastând un șir în al doilea câmp de text furnizat de aplicație.

• Utilizatorul selectează operațiunea „+” din lista furnizată.

• Primul șir furnizat este verificat și transformat cu succes într-un (obiect) polinom.

• Al doilea șir furnizat este verificat și transformat cu succes într-un (obiect) polinom.

• Adăugarea se realizează cu succes, fără erori.

• Rezultatul este afișat în al treilea câmp de text furnizat de aplicație.

**Eroare:**

• Dacă utilizatorul introduce o secvență de intrare greșită în oricare dintre cele două câmpuri de text, va fi atentionat cu o serie de avertismente și va fi nevoit să introduca o secvența de intrare corecta și valida.

# Exemplu 2: Integrarea unui polinom efectuata de catre utilizator:

**Scenariu de succes:**

• Utilizatorul lansează aplicația cu succes.

• Utilizatorul furnizează primul polinom (în primul câmp de text). Utilizatorul poate furniza sau nu o secvență de intrare pentru al doilea polinom. Indiferent de al doilea șir furnizat, operația va fi efectuată DOAR pe primul polinom.

• Utilizatorul apasă butonul „”.

• Șirul furnizat este transformat cu succes într-un obiect de tip polinom.

• Integrarea se realizează cu succes, fără excepții sau erori.

• Rezultatul este afișat utilizatorului.

**Eroare:**

• Dacă utilizatorul introduce o secvență de intrare greșită în primul câmp de text, va fi atentionat cu o serie de avertismente și va fi nevoit să introduca o secvența de intrare corecta și valida.

# Design

# Diagram, schematic Description automatically generatedDiagrama de clase

Problema necesită înțelegerea cât mai bună a operațiilor pe polinoame. După cum se poate observa, deși este necesar ca monomul să aibă doar coeficientul întreg, o proiectare corectă a problemei implică și lucrul cu coeficienți reali. Acesta este motiv pentru care am ales sa

inlocuiesc coeficientul intreg cu unul real.

1. **Modele:**

* Monom
* Polinom

1. **Aplicatii:**

* Operatii - servește ca model principal pentru aplicație: efectueaza operații pe polinoame și monoame.
* MainOperatii – apeleaza operatii pe monoame si polinoame
* Main – lanseaza ecranul aplicatiei
* Controller – leaga interfata de operatiile din clasa Operatii

1. **Teste:**

* Pachetul TesteOperatiiMonoame - testeaza operatiile efectuate pe monoame
* SumaTest, InmultireTest, ImpartireTest,

DerivareTest, IntegrareTest

* Pachetul TesteOperatiiPolinoame - testeaza operatiile efectuate pe polinoame
* SumaTest, DiferentaTest, InmultireTest, ImpartireTest, DerivareTest, IntegrareTest,

SortareTest,GradPolTest, PolinoameEgaleTest,

MonomDominantTest

Aplicatia respecta principiile MCV.

# Monom

Graphical user interface

Description automatically generated with medium confidenceClasa “Monom” are doua campuri private, anume coeficient de tipul Double si grad de tipul Integer (exponentul monomului).

Totodata, aceasta clasa are un constructor fara parametrii, un constructor cu doi parametrii (coeficient si grad) si unul care primeste ca parametru un monom, necesar mai tarziu pentru copierea in adancime (Deep Copy).

Pentru a putea accesa campurile din clasa Monom este nevoie de settere si gettere, dupa cum urmeaza: “getCoeficient”, “getGrad”, “setCoeficient” si “setGrad”.

Penru o mai buna afisare a monomului este nevoie sa fie suprascrisa metoda “toString”, astfel ca monomul va arata dupa cum urmeaza: “(a.0)x^b”, unde a este coeficientul real (Double), iar b gradul (exponentul).

# Polinom

Graphical user interface, text, application

Description automatically generated

Clasa “Polinom” are un singur camp care este privat, acesta constand intr-o lista de monoame (ArrayList).

De asemenea, si aceasta clasa are un constructor fara parametrii, un constructor cu un parametru (Lista de monoame) si unul care primeste ca parametru un polinom, necesar mai tarziu pentru copierea in adancime (Deep Copy).

Pentru a putea accesa campurile din clasa Polinom este nevoie de settere si gettere, dupa cum urmeaza: “getPolinom”, “getPolinom”.

Adaugarea unui monom intr-un polinom se va efectua cu metoda ”adaugareMonom”. Metoda are tipul void, astfel a nu returneaza nimic, ci doar schimba polinomul initial dat ca parametru.

Penru o mai buna afisare a monomului este nevoie sa fie suprascrisa si aici metoda “toString”, astfel ca polinomul va arata dupa cum urmeaza: “(a.0)x^b + (c.0)x^d + … (y.0)x^z”.

Pentru a putea sorta un polinom este necesar ca aceasta clasa sa implementeze “Comparator<Monom>”. Astfel ca va fi necesara suprascrierea metodei compare. Pe aceata am gandit-o in asa fel incat, in urma apelarii sa obtin un polinom sortat avand monomul cu cel mai mare grad in stanga, respectiv monomul cu cel mai mic grad in dreapta.

O ultima metoda necesara in aceasta clasa este aceea care returneaza polinomul cu toate semnele coeficientilor inversate, anume metoda “invPol”. Aceata va fi necesara mai tarziu, cu atat mai mult pentru o mai buna afisare a rezultatului.

# Operatii

Table

Description automatically generated

Clasa “Operatii” detine metodele necesare pentru a efectua operatii asupra monoamelor, dar si asupra polinoamelor. Aceasta nu are campuri specifice.

Metoda ”sum” primeste ca parametru doua monoame si returneaza un singur monom, acesta reprezentand suma celor doua monoame primite ca parametru. Daca exponentii sunt egali, atunci se efectueaza suma coeficientilor, altfel nu se efecyueaza suma.

Metoda ”inm” primeste ca parametru doua monoame si returneaza un singur monom, acesta reprezentand produsul celor doua monoame primite ca parametru.

Metoda ”impartire” primeste ca parametru doua monoame si returneaza un singur monom, acesta reprezentand impartirea celor doua monoame primite ca parametru.

Metoda ”der” primeste ca parametru un monom si returneaza tot un monom, acesta reprezentand derivata monomului primit ca parametru.

Metoda ”integr” primeste ca parametru un monom si returneaza tot un monom, acesta reprezentand integrarea monomului primit ca parametru.

Metoda “sortare” primeste ca parametru un polinom si returneaza, de asemenea, un polinom, dar de data aceasta sortat, avand coeficientul mai mare in partea stanga sic el mai mic in partea dreapta. Se utilizeaza metoda din clasa “Monom” mentionata anterior, anume “compare” pentru a realiza sortarea.

Metoda “gradPol” primeste ca parametru un polinom si returneaza gradul cel mai mare existent in acesta, ca Integer. Aceata metoda are in spate o sortare a polinomului (utilizand metoda de sortare mentionata anterior), luand apoi gradul monomului de pe pozitia 0 (deoarece indexarea in sine se face de la 0), adica de pe prima pozitie si returnandu-l.

Metoda “monomDominant” primeste ca parametru un polinom si returneaza un monom, anume monomul care are gradul “gradPol”, obtinut cu metoda mentionata anterior.

Metoda “polEgale” primeste ca parametrii doua polinoame, urmand sa returneze true daca polinoamele au atat coeficientii cat si gradele egale si false in caz contrar. Asa cum este evident din cele mentionate mai devreme, aceasta returneaza o valoare de tip Boolean.

In cele ce urmeaza voi explica pe scurt opratiile asupra polinoamelor la care utilizatorul are acces, urmand sa pun accent pe aceea care mi s-a parut cea mai solicitanta, anume impartirea a doua polinoame.

Metoda “sumPol” primeste ca parametrii doua polinoame si returneaza un alt polinom reprezentand suma celor doua primate la intrare. Pentru a le insuma este nevoie sa adaug ambele polinoame intr-un nou polinom (acela pe care il voi si returna ulterior, sortat).

Dupa ce le-am adaugat parcurg acest nou polinom cu 2 for uri: unul care incepe de la indexul i = 0 si celalalt de la j = i + 1. Astfel, daca monoamele de pe pozitiile curente au exponentii egali se efectueaza suma acestora si sunt scrise pe pozitia i. Monomul de pe pozita j este sters.

Metoda “difPol” este similara metodei anterior mentionate, ”sumPol”, singura diferenta constand in faptul ca pentru cel de-al doilea polinom apelez metoda de inversare (“invPol”) chiar inainte de a-l adauga in noul polinom creat in metoda, pe care de asemenea il voi parcurge si ulterior sort apt a fi pregatit sa fie returnat.

Metoda “inmultirePol” primeste de asemenea doi parametrii de intrare pe care ii inmulteste, obtinandu-se un singur polinom, anume acela care va fi si returnat. Pentru aceasta metoda am ales sa parcurg cele doua polinoame de intrare cu for each si sa inmultesc fiecare monom. Un truc pe care il folosesc aici pentru a nu obtine moname cu acelasi grad in polinomul returnat este ca apelez pe acesta si un polinom gol metoda de insumare “sumPol”. Astfel reszultatul este adus la o forma normala.

Metoda “derivarePol” primeste ca parametru de intrare un singur polinom. Acesta este parcurs cu for each si intr-un polinom nou voi adauga de fiecare data monomul de pe pozitia curenta derivat cu metoda mentiinata mai sus ”der”. Polinomul nou creat va fi returnat.

Similar metodei de derivare a polinomului este metoda ”integrarePol”, care primeste ca parametru un polinom si, la fel ca mai sus, il parcurge si adauga intr-unul nou monomul cu metoda ”integr” apelata. De asemenea este returnat polinomul nou creat.

Metoda ”impartirePol” primeste doua polinoame prin parametrii de intrare. Algoritmul pe care l-am folosit este acesta:

Text, letter

Description automatically generated

Aceasta functie foloseste fie in mod direct, fie in mod indirect, mare parte din metodele mentionate mai sus, fapt care ii ofera titlul de cea mai complexa metoda dintre cele enumerate.

Pentru inceput verific daca al doilea polinom de intrare este nul, caz in care impartirea nu poate avea loc. Daca acesta nu este nul merg mai departe si declar doua polinoame noi, q (catul) care este gol si r (restul) care va primi primul polinom de intrare, dar avand grija sa fie copiat in profunzime (Deep Copy). Cat timp r nu este nul si gradul lui r este mai mare sau egal decat gradul celui de-al doilea polinom de intare, se vor efectua urmatoarele:

Un nou polinom, t, va tine impartirea celor doua monoame dominante ale lui r cu t.

Apoi q va primi suma polinoamelor q si t

La final r va primi diferenta dintre r si produsul lui t si cel de-al doilea polinom de intrare.

Dificultatea acestei metoda nu consta neaparat in implementarea propriu zisa, ci mai mult in asigurarea corectitudinii celorlalte care o alcatuiesc.

Spre deosebire de toate celelalte si pentru ca aceasta ar trebui sa returneze nu unul, ci doua polinoame, am ales sa returnez in schimb un string in care voi afisa atat catul, cat si restul, avand grija sa le separ estetic.

# MVC

Acesta este principiul folosit tocmai pentru ca am ales sa lucrez pe un proiect JavaFX. Asadar a fost nevoie sa scriu codul pentru interfata, sa completez o clasa, ”Controller”, cu legaturile intre interfata si operatii, dar si un main in care am lansat practice aceasta aplicatie.

Interfata este simpla, cuprinzand doua textbox-uri pentru date de intrare si unul pentru date de iesire, acesta din urma fiind needitabil; si 6 butoane pentru operatii, asa cum am evidentiat si un diagrama de Use Case.

Pentru o utilizare corecta si o experienta mai placuta la impactul cu aplicatia am ales sa mentionez in partea de jos forma in care polinoamele trebuie introduce si, pe de alta parte, am ales o paleta de culori placuta, cu o imagine de fundal contrastanta cu complexitatea conceptelor matematice. Voi atatsa o imagine a acesteia pentru a exemlifica mai bine cele mentionate:

A picture containing text

Description automatically generated

# Testare

In ceea ce priveste testele unitare, am ales sa le separ in doua pachete separate: un pachet pentru operatiile asupra monoamelor si altul pentru operatiile asupra polinoamelor, pe care le-am numit sugestiv. Asa cum se poate observa si in diagrama UML prezentata mai sus, am facut cate o clasa de test pentru fiecare metoda din clasa “Operatii”. Ca si tipar pe care l-am respectat la fiecare dintre acestea mentionez ca am lansat cate doua teste adevarate si unul fals, la acesta din urma utilizand assertNotEquals. Pentru a demonstra existenta si functionalitatea acestor clase voi atasa o imagine cu programul in urma rularii testelor.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Rezultate

Aplicația rezultată este un program foarte ușor de utilizat pentru oricine este ușor familiarizat cu domeniul polinoamelor. Deși calculele nu sunt atât de complicate, aplicația este utilă pentru a face față unor polinoame mari, cu grade mari. De exemplu, efectuarea inmultirii și a impartirii în algebra analitică este o treabă destul de costisitoare în cee ace priveste timpul consumat. Prin introducerea unei intrări corecte în aplicație, programul este capabil să multiplice și să împartă polinoame în cateva milisecunde.

# Concluzii

# Ce am invatat in urma acestui proiect?

Aceasta aplicație a fost un mod foarte util de revizuire a noțiunilor de bază ale OOP. Daca pana acum anumite probleme nu as fi stiut cum sa le rezolv, odata cu acest proiect m-am ambitionat sa duc la bun sfarsit fiecare cerinta. Un alt lucru benefic este acela ca am revazut si operatiile asupra polinoamelor, care aveau tendinta de a ramane uiate.

# Posibilitati de dezvoltare

O imbunatatire semnificativa a acestui program ar fi actualizarea functiei de “Regular Expression” care ar permite utilizatorului aplicatiei sa introduca datele sub o forma mai simpla: spre exemplu termenul liber care va putea fi reprezentat printr-un singur numar, si nu un printr-un numar urmat de x, apoi caracterul “^” si dupa care numarul pentru exponent.

O alta imbunatatire ar putea consta in adaugarea in interfata a celorlalte metode din clasa “Operatii”. Utilizatorul ar beneficia astfel de posibilitatea de a sorta un polinom, de a gasi gradul polinomului, dar si termenul dominant.

Nu în ultimul rând, o problemă foarte importantă în algebra polinomială este găsirea rădăcinilor unui polinom. Cu siguranta aceasta posibilitate ar face programul mult mai util si mai des utilizat.

# Webografie

1. Java ArrayList <https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html>
2. Polinoame <https://en.wikipedia.org/wiki/Polynomial_long_division>[http://www.mathportal.org/calculators/polynomials-solvers/polynomials-operationscalculator.php](http://www.mathportal.org/calculators/polynomials-solvers/polynomials-operations-calculator.php)

<http://lkrv.fri.uni-lj.si/~ajurisic/tec3/gradiva/ECClassroom/Files/BG_8.htm>

1. Raspunsuri la neclaritati: <http://stackoverflow.com/>
2. <https://www.jetbrains.com/help/idea/javafx.html#vm-options>
3. <https://www.jetbrains.com/help/idea/testing.html>
4. Calculator de polinoame: <https://www.emathhelp.net/calculators/algebra-1/polynomial-calculator/>