

Tehnici de Programare

Tema 1 – Calculator pentru polinoame

Profesor curs: Conf. Tudor Cioara Sudent: Varga Alexandra-Ioana

Profesor laborator: Dorin Moldovan Grupa: 30223

Cuprins

1. Obiectivul proiectului
2. Analiza problemei
   1. Modelare
   2. Cazuri de utilizare
3. Proiectare
   1. Diagrama UML
   2. Structuri utilizate in implementare
   3. Relatii
   4. Pachete
   5. Algoritmi
   6. Interfata utilizator
4. Implementare si testare
5. Concluzii
6. Bibliografie
7. Obiectivul proiectului

Obiectivul acestei teme este de a ne familiariza cu operatiile pe polinoame si, totodata, este o modalitatea de a ne reaminti programarea in Java. Fiecare operatie (adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea) a trebuit implementata in cod Java, exemplificata si testata.

Datele de intrare (polinoamele) sunt introduse ca si String-uri, apoi prelucrate in clasele definite si apoi afisate tot ca si String-uri.

1. Analiza problemei

Tema referitoare la operatiile pe polinoame este un proces complex care necesita atentie la fiecare pas, atat la implementarea fiecarei operatii cat si la testarea a cat mai multe exemple in scopul verificarii acestora.

Este necesara o asemenea abordare din cauza faptului ca fiecare operatie are exceptiile sale si trebuie tratate fiecare in parte pentru ca programul sa functioneze in conditii de fata impecabil.

* 1. Modelarea proiectului

Complexitatea proiectului poate fi facuta in zeci de moduri de implementare, fiecare avand eficienta sa. Prima implementare, la care toata lumea s-ar gandi, ar fi cea in care am folosi vectori pentru a stoca coeficientii si gradul fiecarui polinom. Apoi adunarea, scaderea, inmultirea sau impartirea coeficientilor si a gradului afisat intr-un al treilea vector de unde se va procesa String-ul de afisat. Aceasta implementare ar avea cateva probleme, mai ales legate de eficienta.

O alta modalitate ar fi cea de a lucra direct pe String-uri. Ar trebui sa despartim si sa convertim fiecare grad si coeficient in tipul potrivit si sa le prelucram, dupa care ar trebui sa le salvam din nou intr- un al treilea String.

Eu am incercat sa implementez in felul urmator: am preluat fiecare String in parte, am despartit prin functii „token” in functie de semn si apoi salvarea acestora in Stringuri diferite. Fiecare e luat si prelucrat in parte, se vor extrage coeficientii si gradurile fiecaruia, apoi vor fi salvate intr-o lista de monoame (polinom format dintr-un singur termen, e.g. 4x, 3x^2). Lista va contine atat coeficientul, cat si gradul si String-ul in sine.

Dupa ce prelucram aceste monoame si le stocam intr-o lista de monoame, se pot aplica toate functiile corespunzatoare operatiilor din clasa ,,Polinom”. Pentru adunare, scadere, inmultire, impartire, derivare si integrare se vor folosi ambele polinoame.

Dupa introducerea intr-un monom nou a rezultatului, acesta va fi readus la forma unui String si in cele din urma afisat in interfata grafica.

Pentru o buna functionare a programului String-ul introdus in interfata grafica trebuie sa respecte pattern-ul de forma “coefx^exp”, fara de care se va afisa un mesaj de eroare. In cazul in care polinomul este introdus correct, optiunile sunt clare: adunare, scadere, inmultire, impartire, derivare si integrare, fiecare din acestea avand field-urile proprii pentru afisarea rezultatelor.

* 1. Cazuri de utilizare

Avand in vedere faptul ca acest proiect rezolva operatiile pe polinoame, oricare ar fi polinomul introdus, rezultatul fiind correct de fiecare data. Acest proiect poate fi folosit cu usurinta ca o aplicatie atat pentru profesori, cat si pentru elevi.

Totodata, avand in vedere ca este prezenta interfata grafica si nu este necesara intelegerea codului, aceasta poate fi folosita de oricine, fara a prezenta dificultati. Task-ul fiind simplu constand in introducerea a oricare doua polinoame si afisarea rezultatului dupa selectarea optiunii dorite, prin intermediul celor 6 butoane desenate cu icon sugestiv operatiilor, rezultatul fiind afisat in fiecare text field dedicat operatiilor efectuate.

1. Proiectare
   1. Diagrama UML

O imagine care conține text, negru, captură de ecran

Descriere generată automat

In diagrama UML de mai sus sunt prezentate toate clasele, cu atributele si relatiile dintre acestea, clasele principale sunt :

PROIECT REALIZAT DE TIP MVC:

MODEL :

Clasa “Polinom” unde sunt stocate toate operatiile pe polinoame: adunare, scadere, inmultire, impartire, derivare si integrare.

Clasa “Monom” este clasa unde se stocheaza fiecare monom dupa split, aici sunt stocate coeficientii, exponenti si String-ul pentru fiecare monom. Totodata, aici este prezenta metoda validareMonom aceasta verifica toate cazurile posibile de introducere a unui String si corectitudinea acestuia.

VIEW :

Clasa “CalculatorGUI”, unde este prezent si main-ul, este constituita din JPanel, JText si JButton, cu ajutorul carora utilizator poate apela fiecare functie din program si afisarea rezultatelor acestora. La fiecare buton, se preaiu cele doua polinoame ca si String-uri se face operatia si se returneaza rezultatul tot ca si String.

CONTROLLER:

Clasele OperatieElementara si OperatieAvansata au o stransa legatura deoarece ambele cuprind apelul functiilor ce efectueaza operatii pe polinoame in functie de comanda primita din exterior (apasarea butonului corespunzator operatiei). Clasa OperatieAvansata cuprinde efectuarea integrarii si derivarii unui polinom precum si crearea acestuia ca String de output. In clasa OperatieElementara sunt efectuate celelalte operatii si este efectuata legatura directa de apel a functiilor de control si afisarea rezultatului acestora in GUI-ul aplicatiei.

TEST JUNIT:

Clasa “OperatiePolinomTest” este o clasa de Test implementat cu Junit care verifica fiecare operatie daca este corecta sau nu, utilizand cateva exemple pentru testare. In cazul in care nu este corecta vreo operatie imi va afisa mesajul de eroare.

* 1. Structuri utilizate in implementare

Cele mai frecvente structuri utilizate in acest proiect sunt ArrayList-urile, folosite pentru monoame “Monom” in scopul faptului ca avem coeficienti reali, totodata este nevoie de introducerea si scoaterea din liste pe tot parcursul programului a coeficientilor, string-urilor si a exponentilor. Listele ne ofera o capabilitate de manevrare a datelor mult mai eficienta si mai complexa in special raportate la structure de tip array. Nu mai este necesara contorizarea, cautarea a unui anumit element utilizand un index ci, prin intermediul listelor vom accesa la fiecare pas obiectele monom “Monom” care este mult mai eficient si mai folositor.

* 1. Relatii

Clasele lucreaza ca un tot unitar, astfel incat daca lipseste una programul nu va functiona correct, nu este un proiect independent, ci dependent, dependent de implementare fiecarei clase si de relatiile dintre acestea. Relatiile se pot observa cu usurinta in diagrama UML, atat relatii de tip 1:n cat si 1:1.

* 1. Pachete

Am lucrat structurat MVC (Model-View-Controller)

PACHET MODEL: Contine clasele Monom si Polinom – ce reprezinta modelul datelor cu care lucram pe proiect. Polinoamele ce sunt introduse de utilizator vor fi „sparte” in monoame, iar mai apoi reconstituirea acestora drept date cu care sa efectuam operatii vor fi reprezentate in clasa Polinom – va contine liste de monoame cu care vom putea efectua operatiile cerute.

PACHET VIEW: Contine clasa CalculatorGUI, ce construieste intreaga interfata grafica cu care va interactiona utilizatorul, se utilizeaza in totalitate Java Swing pentru construirea GUI-ului.

PACHET CONTROLLER: Clasele OperatieAvansata si OperatieElementara fac legatura dintre partea de Model si View al proiectului.

In pachetul MODEL avem implementate operatiile la nivel de monom si polinom (ca model de dat), iar in pachetul VIEW avem interfata grafica unde dorim sa afisam rezultatele operatiilor efectuate pe polinoamele introduse => in pachetul CONTROL avem clasele ce implementeaza operatiile de la nivel intern la nivel extern => rezultatul operatiilor este vizibil utilizatorului, fiind afisat in label-urile corespunzatoare operatiilor apelate.

* 1. Algoritmi

In continuare, se vor prezenta toti algorimtii utilizati pentru operatiile pe polinoame, si anume:

1. Adunarea polinoamelor

Acest algoritm este implementat in clasa “Polinom” si functioneaza pe un principiu relative simplu. Primul pas este de a apela functia sortareDescrescatoare() in constructorul Polinomului, in scopul reducerii de monoame reductibile si de a le sorta. Cele 2 polinoame sunt stocate in 2 liste separate iar adunarea lor se face intr-un loop prin intermediul unui iterator, astfel, adunarea se face termen cu termen verificand la fiecare pas exponentii egali.

Metoda aceasta va crea un polinom nou avand ca si parametru Lista de Monoame, il va crea si in cele din urma returna.

1. Scaderea polinoamelor

Diferenta majora dintre metoda de adunare si scadere este aceea ca in loc sa adunam, vom scadea la fiecare pas, iar prin intermediul unui for, vom merge termen cu termen prin lista si vom adresa fiecarui coeficient din cel de-al doilea polinom cate un “-“ in scopul scaderii acestora.

3) Inmultirea polinoamelor

Acest algoritm este implementat in clasa “Polinom”, alaturi de cele 2 metode de mai sus. Acest algoritm este putin mai vast si cere mai multa atentie pentru a-l putea intelege. Algoritmul este implementat in metoda inmultirePolinom. Primul pas, ca si mai sus, este de a apela functia sortareDescrescator() pentru ambele polinoame in scopul reducerii termenilor reductibili si in scopul sortarii. Cele 2 polinoame sunt stocate in 2 liste separate, pe care vom lucra mai departe in algoritm. Primul pas in acest algoritm este de a inmulti toti termenii intre ei si de a rezulta o lista cu fiecare monom in parte.

Dupa aceea, vom merge cu 2 for-uri in lista rezultanta, comparand fiecare obiect cu fiecare obiect, avand o conditie sa nu fie comparat acelasi obiect. Aceste “for”-uri ne ajuta in scopul verificarii fiecarui monom cu acelasi exponent si reducerea acestora. Practic, se inmultesc toti termenii din cele 2 liste iar dupa aceea se face reducerea. Metoda aceasta, ca si cele de mai sus, va crea un polinom nou rezultant avand ca si parametru lista finala de “Polinom” si il va returna.

4) Derivarea polinoamelor

Acest algoritm este implementat in clasa “Polinom”, alaturi de metodele responsabile pentru operatiile de Adunare, Scadere, Inmultire si Impartire. Metoda “derivarePolinom” care o implementeaza primeste de data aceasta lista de monoame pe care lucreaza. Primul pas este de a apela functia sortareDescrescatoare() in constructor si in acest caz, in scopul reducerii de monoame reductibile si de a le sorta. Polinomul este stocat intr-o lista unde se va face derivarea efetiva. Din cele cinci metode aceasta este cea mai simplu de implementat datorita faptului ca trebuie modificat la fiecare monom doar coeficientul si exponentul. Mai exact, trebuie aplicata derivarea si avem nevoie doar de un if si un else implementate intr-un loop pe tot parcusul listei, anume:

Dupa aceasta, se va crea o lista cu noi coeficientii si exponenti rezultanti la fiecare Monom si se va concatenta astfel creand un polinom nou avand ca si parametru lista de monoame, il va crea si in cele din urma returna.

6) Integrarea polinoamelor

Acest ultim algoritm prezent in clasa “Polinom” alaturi de cele de mai sus functioneaza pe acelasi principiu ca si derivarea avand doar unele conditii schimbate, pe langa faptul ca rezultatul va trebui sa fie cu coeficienti reali.

Metoda “integrarePolinom” care o implementeaza primeste de data aceasta lista pe monoame pe care lucreaza. Primul pas este de a apela functia sortareDescreascatoare() si in acest caz, in scopul reducerii de monoame reductibile si de a le sorta. Polinomul este stocat intr-o lista unde se va face integrarea efectiva.

Mai exact, trebuie aplicata integrarea si avem nevoie doar de un if si un else implementate intr-un loop pe tot parcursul listei.

6) Algoritmul de transformare a unui String intr-un polinom

Acest algoritm este unul complex si este implementat in functia crearePolinom din clasa OperatieAvansata, se verifica daca se doreste sa facem derivare sau integrare (construim un polinom nou corespunzator) a polinoamelor altfel se sparge polinomul pentru a putea fi folosit la celelalte operatii unde avem nevoie de 2 polinoame in aceasta forma pentru a obtine rezultatele dorite. Se extrag monoame pe pattern-ul decis (de exemplu un monom are forma: -3x^2), iar polinoamele finale sunt liste de astfel de monoame extrase din string. Am utilizat Pattern Matching si Regular Expression pentru a determina polinoamele, de asemenea in clasa Monom exista o metoda aditionala validareMonom care verifica daca un polinom introdus sub forma de String este corect din punct de vedere al pattern-ului ales. (ax^n+bx^n-1-cx^n-2+ ...).

7) Algoritmul de transformare a polinomului intr-un String

Acest algoritm este apelat in clasa “Polinom”, unde se creeaza o lista de monoame “Monom” avand ca si parametrii coeficientii si exponentii, in momentul in care am introdus numarul de coeficienti si exponenti necesari pentru crearea polinomului vom transmite lista ca si parametru si vom crea polinomul cu ajutorul metodelor getMonom() si getMonomInt() din clasa Monom. In aceasta metoda sunt prezente toate cazurile posibile pentru a facilita afisarea corespunzatoare.

* 1. Interfata cu utilizatorul

Interfata cu utilizatorul este cat se poate de simplista, fiind o aplicatie de calculare a operatiilor pe polinoame, am introdus doar strictul necesar si anume: cele 6 butoane pentru operatii, pe langa acestea in dreptul fiecarui buton avem un JField unde se va afisa rezultatul fiecarei operatii apasate, si 2 JText cu “Polinom 1” si “Polinom 2” respective cele 2 JField unde se vor introduce polinoamele.

Polinoamele introduse trebuie neaparat sa respecte formatul “coefx^exp” altfel va aparea mesaj de eroare si nu se vor putea efectua operatiile pe polinoame.

O imagine care conține text

Descriere generată automat

1. Implementare si testare

Testarea pentru fiecare operatie este prezenta in clasa “OperatiePolinomTest” prezentata mai sus care testeaza fiecare operatie cu cate un exemplu diferit, daca se va rula aceasta se poate observa cu usurinta ca fiecare operatie merge corect, totodata se poate testa direct din interfata, dupa preferinta utilizatorului.

1. Concluzii si dezvoltari ulterioare

Dupa parerea mea, cum am spus si mai sus aplicatia asta va putea fi folosita de catre oricine va avea nevoie de a calcula eficient si rapid o operatie pe polinoame. La nivel de dezvoltari ulteriroare voi implementa mai mult ca sigur si operatia de impartire, voi adauga si alte operatii pentru ca aplicatia sa dea dovada de o complexitate de mare amploare, totodata as dori sa fac testarile mai ample si metodele mai scurte, sa aiba un aspect calitativ direct proportional cu cantitatea.

De asemenea, la nivel de interfata, pentru ca utilizatorul sa aiba o placere de a interactiona cu aceasta, voi introduce noi aspecte, diagrame, desene, butoane, poate si ceva dinamic care sa il ajute pe utilizator in aceasta maniera.

Fiind primul proiect de asa amploare, avand toate clasele complexe, cu diagrame, cu testari, cu interfata pot spune sincer ca m-a ajutat in dezvoltarea abilitatilor mele de programator atat la nivel de gandire cat si la nivel de structurare a datelor.

1. Bibliografie

* <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>
* <https://www.geeksforgeeks.org/>
* <https://www.youtube.com/>
* Cursurile, laboratoarele, temele si proiectul de POO din anul 2, semestrul 1.