

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

Calculator de polinoame

Documentație

Dorofte Andrei

Grupa 302210

Cuprins

1. Obiectivul temei

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

1. **Obiectivul temei**

Obiectivul principal este de a implementa, in limbajul Java, un calculator ce poate realiza operații cu 2 polinoame, respectiv adunare, scădere, înmulțire, împărțire, derivare si integrare, prin intermediul unei interfețe grafice ușor de utilizat. Coeficienții si puterile monoamelor vor fi de tip real. Ca oricare alt sistem de calcul, si acesta prezinta date de intrare si date de iesire, generate dupa efectuarea calculelor.

|  |
| --- |
| **Obiective secundare** |
| * Crearea claselor necesare |
| * Definirea metodelor de baza pentru fiecare clasa |
| * Implementarea algoritmilor operațiilor |
| * Construirea interfeței grafice |
| * Construirea unei expresii de tip Regex pentru separarea textului extras din interfață |
| * Implementarea unității de testare |

1. **Analiza problemei, modelare, scenarii, cazuri de utilizare**

Un polinom reprezinta o expresie, formata din mai multe monoame, care, la randul lor sunt construite cu ajuorul unor coeficienti si exponenti, intregi sau reali. Exista o multitudine de metode prin care se poate realiza un calculator de polinoame. Modul prin care se introduc informatiile (coeficienti si exponenti, polinoame), modul in care sunt ulterior asezati, sortati , aspectul final, si aspectul interfetei grafice sta doar la latitudinea programatorului.

Trebuie sa avem in vedere faptul ca rezultatul tuturor operațiilor ce trebuie implementate, este de asemenea un polinom. Astfel, am ales ca polinomul sa fie introdus de la tastatura, la fel cum ar trebui sa fie introdus intr-un sistem digital, de exemplu: „2x^2+1x^1+0x^0”. Fiecare termen al polinomului este denumit monom, astfel, cea mai intuitiva abordare ar fi sa folosim colecții pentru stocarea acestora. Interfața grafica va include butoane corespunzătoare fiecărei operații, câmpuri de introducere a datelor de intrare si câmp de afișare a datelor de ieșire.

Extragerea coeficientului si a puterii fiecărui monom din string-ul introdus se va face cu ajutorul unei expresii regulate, care va detecta si împarți fiecare monom din string in coeficient si putere.

* Sistemul trebuie sa realizeze suma a doua polinoame introduse corespunzător de la tastatura
* Sistemul trebuie sa realizeze diferența a doua polinoame introduse corespunzător de la tastatura
* Sistemul trebuie sa realizeze produsul a doua polinoame introduse corespunzător de la tastatura
* Sistemul trebuie sa realizeze derivarea unui polinom
* Sistemul trebuie sa realizeze integrarea unui polinom

Polynomial Calculator

Male icon

Add

Substract

User

Multiply

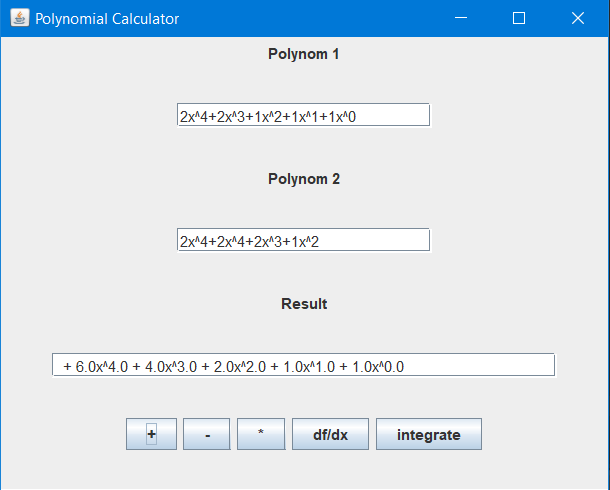
Derivative

Integrate

Result

Caz de utilizare pentru adunare:

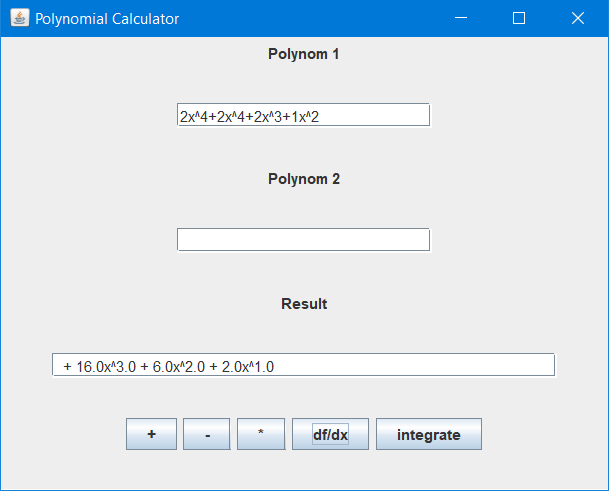
1. Utilizatorul introduce corespunzător cele doua polinoame in câmpurile Polinom 1 si Polinom 2, de exemplu: „5x^3+ 2x^2+1x^0”.
2. Utilizatorul apasă pe butonul corespunzător operației pe care dorește sa o efectueze.
3. Sistemul afișează rezultatul obținut in urma efectuării operației in câmpul Result, aflat deasupra butoanelor operațiilor.



Pentru scădere si înmulțire, cazul de utilizare este același cu cel al adunării.

Caz de utilizare pentru derivare:

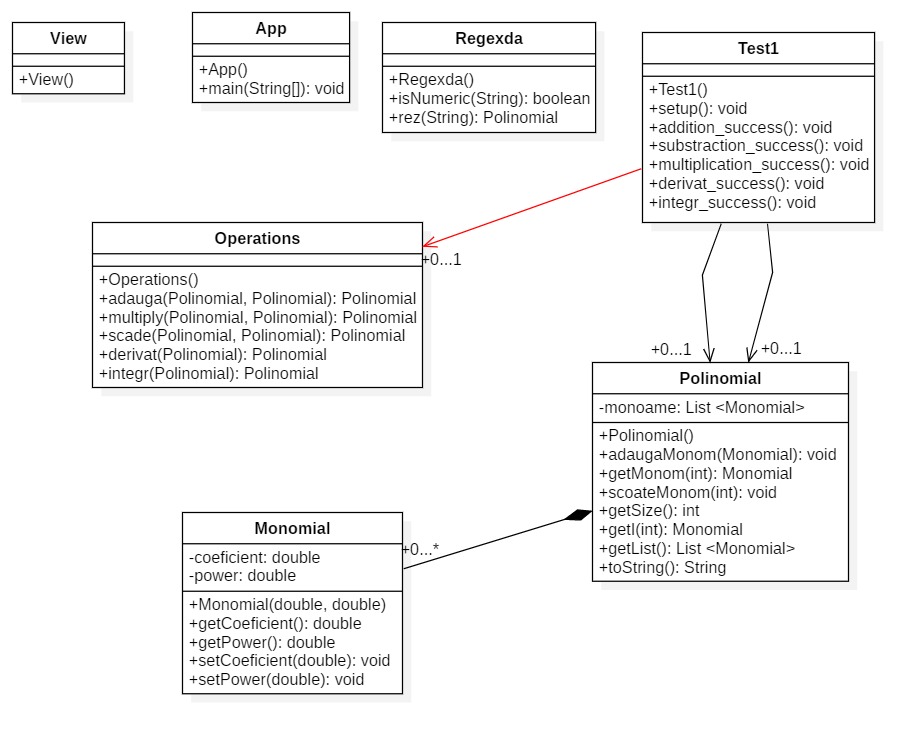
1. Utilizatorul introduce corespunzător polinomul in câmpul Polinom 1, de exemplu: „5x^3+ 2x^2+1x^0”.
2. Utilizatorul apasă pe butonul corespunzător operației pe care dorește sa o efectueze.
3. Sistemul afișează rezultatul obținut in urma efectuării operației in câmpul Result, aflat deasupra butoanelor operațiilor.



1. **Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfețe, relații, packages, algoritmi, interfață utilizator)**

Unified Modeling Language sau UML, pe scurt, este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.

**Diagrama UML:**



(c)

(b)

(a)

Am încercat sa modularizez cerința cat de mult posibil, astfel rezultând clasele din diagrama de mai sus: Monomial, Polinomial, Operations, Regexda, App, View si Test1. Clasele au fost proiectate intuitiv, pentru a fi ușor de citit si de înțeles.

**Relații:**

* Clasa Operations folosește o colecție de Polinoame, deci exista o relație de agregare
* Clasa Polinom folosește o colecție de Monoame, deci exista o relație de compoziție
* Clasa Test1 instantiaza un obiecte de tip Monom si Polinom, deci apare o relație de asociere
* Clasa View instantiaza polinoame, astfel apare o alta relație de asociere

1. **Implementare**

**Interfață utilizator (GUI) – Clasa View**

Interfața grafică (în engleză: Graphical User Interface sau GUI) este o interfață cu utilizatorul bazată pe un sistem de afișaj ce utilizează elemente grafice. Interfața grafică este numit sistemul de afișaj grafic-vizual pe un ecran, situat funcțional între utilizator și dispozitivele electronice.

Interfața utilizator a fost implementata utilizând librăria Java Swing, care conține elemente grafice. Am creat mai întâi un frame, după care am utilizat cate un panel separat pentru label-uri, pentru descrierea datelor de intrare, textfield-uri si butoane corespunzătoare fiecărei operații, panel-uri ce au fost adăugate ulterior in frame.

Frame = “rama” in care se adauga toate elementele de care avem nevoie pentru buna functionare a programului. E o fereastra care, dupa utlizare, apasand butonul X din dreapta sus, se poate inchide (Exit\_on\_close).

Fiecare buton are propriul action listener, care creează cate o instanță pentru regex, operația aferenta butonului, si polinoamele extrase din textfield-uri. Aranjarea in frame am realizat-o cu ajutorul BoxLayout. Acest layout se bazează pe aranjarea pe orizontala sau verticala a elementelor grafice. Am optat pentru aranjare pe verticala, întrucât am creat cate un panel separat pentru fiecare componenta.

**Clasa Regex**

O expresie regulată, regex sa (numită uneori expresie rațională) este, în informatica teoretică⁠ și în teoria limbajelor formale, un șir de caractere⁠ care definesc un șablon⁠ de căutare. De obicei, acest șablon este apoi utilizat de către algoritmii de căutare pe șiruri⁠ pentru operațiuni de „căutare” sau „căutare și înlocuire” operațiuni pe șiruri de caractere⁠.

Clasa Regex este folosita pentru a extrage din textfield-uri cifrele corespunzătoare coeficientului si exponentului fiecărui monom din polinomul introdus. Astfel, metoda isNumeric, folosește o expresie regulata pentru a verifica daca, caracterul extras din string este sau nu un număr. Se creează un matcher si un pattern. Prima data este creat pattern-ul din expresia regulata, după care se creează un matcher din pattern-ul deja existent. Matcher-ul caută in string apariții care coincid cu expresia regulata, prin intermediul unei bucle while. Când este găsit un grup ce coincide cu expresia regulata, se validează pârțile care ar trebui sa fie numerice, cu ajutorul funcției isNumeric, după care se asignează, cu un cast prealabil, unor variabile. Astfel, se creează un monom nou cu valorile găsite, care este ulterior adăugat intr-un polinom nou creat, la începutul metodei.

**Clasa Monomial**

Clasa Monomial este nucleul acestui proiect, deoarece, prin intermediul ei se creează fiecare monom ce este introdus de la tastatura. Acesta reprezintă structura unui monom de sine stătător, având ca atribute coeficientul si puterea, ambele de tip double, pentru a fi mai ușoară realizarea algoritmilor. Poseda doar constructor cu parametri, întrucât unul fără parametri nu ar avea utilitate. S-au generat getteri si setteri pentru coeficient si putere, aceștia fiind utilizați in realizarea operațiilor.

**Clasa Polinomial**

Clasa Polinomial reprezintă structura unui polinom, reprezentata simbolic printr-o lista (ArrayList) de monoame, întrucât acest tip de lista este mai eficienta decât un vector tradițional. Constructorul nu are parametri. Exista metode de adăugare si scoatere monom, de returnare a unui anumit monom, de returnare a listei de monoame si a dimensiunii acesteia. De asemenea, metoda de afișare a listei de monoame, toString, afișează lista sub forma unui polinom, de ex: „ + 5.0x^3.0 + 2.0x^2.0 + 1.0x^0.0”.

**Clasa Operations**

In aceasta clasa am definit cate o metoda pentru fiecare operație: adunare, scădere, înmulțire, derivare si integrare.

Algoritmul de adunare, parcurge cele 2 liste ale polinoamelor create din datele de intrare. Am folosit o bucla while, unde se verifica puterile fiecărui monom, iar unde puterile sunt egale, se aduna coeficienții, se adaugă monomul într-o noua lista si se cresc iteratorii. Unde puterile nu sunt egale, se verifica care dintre puteri este mai mare, se creste iteratorul respectiv si se adaugă monomul in lista de rezultate. Pentru a evita cazuri nefavorabile, de exemplu: „+ 4.0x^4.0 + 4.0x^4.0”, care pot apărea in rezultat, am creat un polinom nou, diferit de polinomul rezultat, iar la finalul buclei while, am apelat metoda de adunare pe noul polinom si polinomul rezultat, după care am golit lista de monoame a polinomului rezultat. Astfel, la fiecare iterație a buclei while, lista polinomul rezultat va fi goala, iar la final se vor aduna rezultatul iterației precedente si cel al iterației actuale, lucru ce va elimina cazul nefavorabil prezentat mai sus.

Algoritmul de scadere este asemanator cu cel de adunare. Ulterior, pentru monoamele din Polinomul 2 care nu au avut pereche ca exponent, sunt introduse si ele in lista. Diferenta fata de adunare, este ca aici, la parcursul polinomului 2, cel care se scade, monoamele fara pereche primesc acelasi coeficient, dar cu semn negativ:

(0 - b.getCoeficient()).

Algoritmul de înmulțire este realizat cu 2 bucle while, astfel inmultindu-se fiecare monom cu fiecare, rezultatul fiind un nou monom, având coeficientul egal cu produsul coeficienților celor doua monoame, iar puterea egala cu suma puterilor celor doua monoame. Acesta este adăugat ulterior unei noi liste dintr-un nou Polinom. De asemenea, la sfârșitul buclei while am procedat la fel ca in cazul algoritmului de adunare sau scădere, pentru a aduna termenii cu puteri egale din rezultat.

Algoritmii de derivare si integrare sunt foarte asemănători, ambii fiind realizați cu o bucla de tipul for each. Ambele metode au doar cate un parametru, intrucat, atat derivarea, cat si integrarea, se realizeaza asupra unui singur polinom. Se salvează in cate o variabila, puterea si coeficientul monomului curent, după care se creează un nou monom, avand coeficientul si puterea modificati corespunzator, care este adăugat într-o noua lista a unui nou Polinom. De asemenea, la sfârșitul buclei while am procedat la fel ca in cazul algoritmului de adunare sau scădere, pentru a aduna termenii cu puteri egale din rezultat.

**Clasa App**

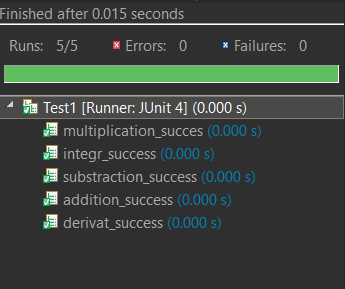
Clasa App conține Main-ul proiectului, si este clasa in care spunem programului ce sa execute. Aceasta creează o instanță a interfeței grafice. Testele sunt realizate separat in clasa Test1.

**Clasa Test1**

Clasa Test1 conține testele corespunzătoare fiecărei operații. Acestea au fost realizate prin intermediul Junit si assertEquals, pentru testarea rezultatului fiecărei operații. Am testat toString-ul fiecărui rezultat cu rezultatul corect, scris de mana.

1. **Rezultate**

Testarea si rezultatele au fost implementate cu ajutorul librăriei JUnit. Am instantiat o metoda @BeforeClass, care se va executa mereu înainte de testarea propriu zisa, ce conține declarații de monoame, polinoame si operații. Se creează doua polinoame noi ce vor fi testate, iar metodele de test sunt precedate de @Test, lucru ce indica faptul ca sunt metode de testare a operațiilor. Se simulează operațiile implementate pe cele 2 polinoame si se verifica cu assetEquals daca toString-ul rezultatului coincide cu ceea ce ar trebui sa se afișeze.



Interfata grafica este destul de simpla si usor de utilizat, chiar si pentru cineva care ia contact pentru prima data cu aceasta aplicatie.

1. **Concluzii**

In concluzie, sunt de părere ca acest proiect m-a ajutat sa îmi aprofundez cunoștințele in tot ce înseamnă limbajul Java, implementarea paradigmelor OOP, crearea unui program cu o interfață grafica si dezvoltarea unui algoritm eficient, care sa satisfacă obiectivul unui proiect.

De asemenea, m-a ajutat sa îmi reamintesc tehnicile de programare învățate semestrul trecut si sa îmi modelez gândirea astfel încât sa pot dezvolta un proiect, numai după o cerință data. Totodata, am inteles mult mai bine cum functioneaza colectiile si metodele de accesare si modificare a elemetelor acestora. De asemenea, am invatat ce sunt expresiile regulate, la ce folosesc si modul de utilizare intr-o aplicatie de tip Java.

In ceea ce privește dezvoltarea ulterioara a programului:

* se poate implementa un buton care sa reseteze calculatorul, pentru introducerea unor noi polinoame.
* de asemenea, se poate adăuga un buton de informații, care sa prezinte utilizatorului cum ar trebui introdus polinomul si sa descrie pe scurt funcționarea programului.
* Se poate imbunatati performanta algoritmilor
* Se poate implementa algoritmul de gasire al radacinilor unui polinom
* Implementarea calculului cu polinoame de mai multe variabile, sau chiar variabile complexe

1. Bibliografie

* <https://stackoverflow.com/>
* Cursul de TP si POO
* http://docs.oracle.com/