# 

Documentație Tema1

Calculator de Polinoame

Nume: Câmpean Adelina Ioana

Grupa: 30227

Profesor Laborator: Pop Cristina

Cuprins

1. Cerințe Funcționale
2. Obiectivul temei
3. Informații Matematice
4. Analiza Problemei
   1. Diagrama USE CASE
   2. Diagrama de pachete
   3. Diagrama de clase UML
   4. Clase
5. Concluzii
6. Bibliografie

**CALCULATOR DE POLINOAME**

1. Cerința problemei

Propuneți, proiectați și implementați un sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi.

1. Obiectivul temei

Obiectivul temei este realizarea unui calculator de polinoame, în care introducem de la tastatură două polinoame, de forma (ex. 2x^3+3x^2-5x+4) iar prin apăsarea diferitelor butoane, se va realiza adunarea, scăderea, înmulțirea, împărțirea, derivarea si integrarea celor două polinoame.

1. Informații Matematice

Un polinom, este format dintr-o listă de monoame, iar un monom este alcătuit dintr-un coeficient, o variabilă și respectiv o putere (ex : 2x^2).

Iar un polinom, are structura 3x^3+2x^2-4x+2.

Adunarea polinoamelor se realizează adunând coeficienții monoamelor cu același grad, iar dacă polinoamele au grade diferite, atunci se adaugă în polinomul rezultat, fiecare monom.

Scăderea polinoamelor, funcționează pe același principiu ca și adunarea, se scad coeficienții monoamelor de același grad, iar dacă au grade diferite, se adaugă în polinomul rezultat, in ordinea descrescătoare a gradelor.

Înmulțirea polinoamelor, se realizează înmulțind fiecare element al primului polinom cu fiecare element al celui de-al doilea polinom. Apoi, se adună coeficienții monoamelor cu același grad și se obține polinomul final.

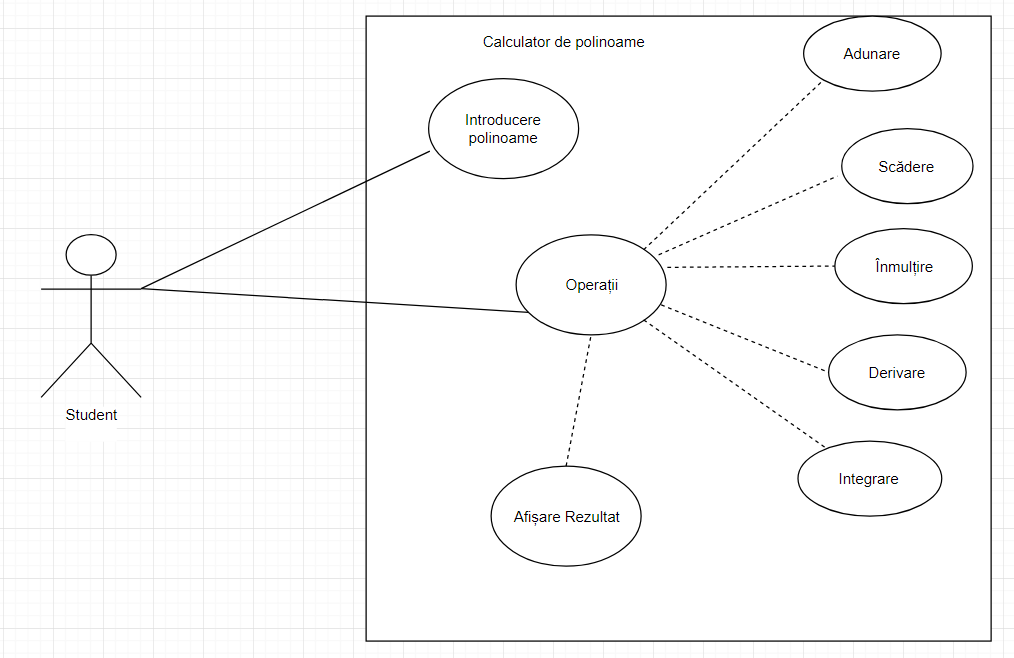
Împărțirea polinoamelor, se realizează ordonând descrescător gradele celor două polinoame, apoi se împarte polinomul mai mare la polinomul de grad mai mic. Se împarte primul termen al primului polinom cu primul termen al celui de-al doilea polinom, astfel se obține primul element al câtului. Se înmulțește rezultatul cu împărțitorul polinomului 2 și acest produs, se scade din deîmpărțitul polinomului 1 (adică se adună acest produs cu semn schimbat la polinomul 1). Astfel se obține primul rest al împărțirii.

Procedura se repetă, utilizând restul, ca deîmpărțit, până în momentul în care restul este mai mic decât împărțitorul.

Derivarea unui polinom, se realizează pe fiecare monom, utilizând formulele de derivare. Se aplica formula: x^n = n\*x^(n-1). Adică, dacă un monom este de forma 2x^3, atunci derivat va fi 2\*3x^2-1, 6x^1.

Integrarea unui polinom, este operația inversă derivării. Se aplică pe fiecare monom, aplicând formula: = 1/(n+1)\*x^n+1 dx. Dacă un monom este de forma 2x^3, atunci integrat va fi 2/4x^4.

1. Analiza problemei
   1. Diagrama USE CASE



Studentul =actor - reprezintă persoana care dorește să efectueze operații pe polinoame

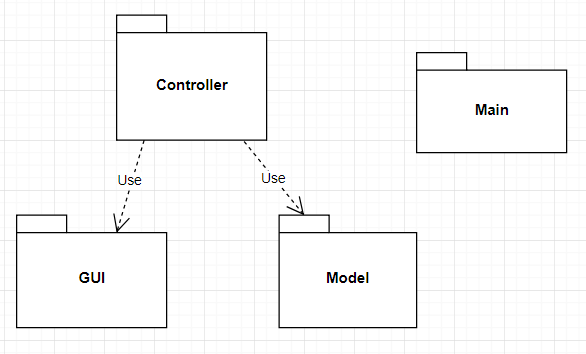
Calcul polinoame = alegerea operației

Afișare rezultat = este legat de calcul, deoarece în funcție de operația aleasă, se afișează

Introducere Polinoame = actorul introduce polinoamele

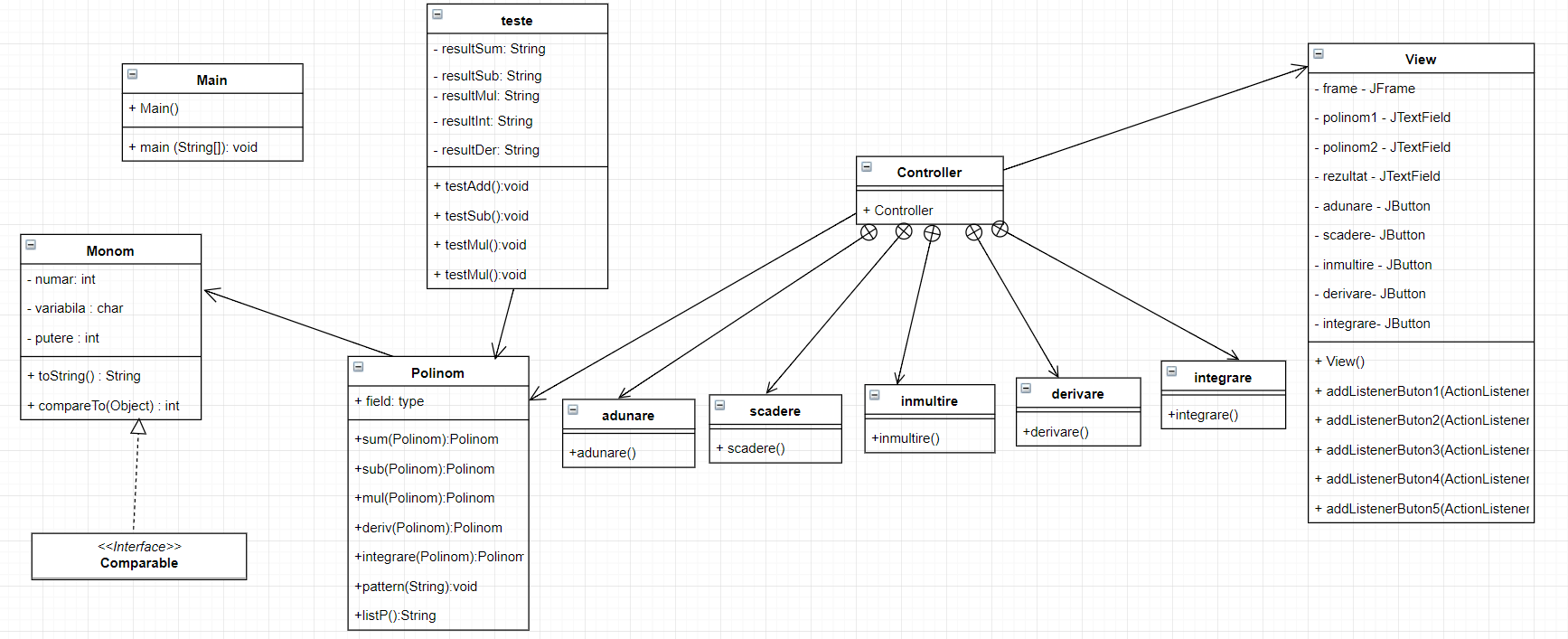
Scenariu de utilizare:

1. Studentul introduce polinomul 1
2. Studentul introduce polinomul 2
3. Din calcul, își alege ce operație dorește să efectueze
4. Afișează rezultatul
   1. Diagrama de pachete



În pachetul controller, se importează clasa View din pachetul GUI, și clasele Monom si Polinom din pachetul Model.

* 1. Diagrama de clase UML

****

––

Diagrama UML este alcătuită din 6 clase: Polinom, Monom, View, Controller, teste si Main.

Clasele Monom, Polinom, Main și teste sunt realizate în interiorul pachetului Model.

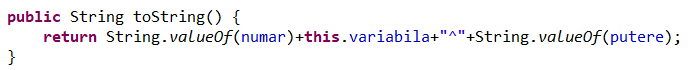
Fiecare clasă, are in componență un constructor, care este folosit pentru crearea unui nou obiect. În momentul în care instanțiem un nou obiect, acela va fi de forma constructorului declarat in clasa respectivă.

Clasa Monom

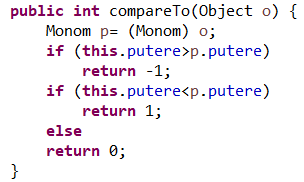
Este realizată pentru a determina structura unui monom, are declarate ca variabile de instanță, un număr, adică coeficientul termenului, o variabilă, operațiile realizându-se pe polinoame cu o singură variabilă și o putere, adică gradul fiecăruia.

Variabilele de instanță sunt private, pentru a nu putea fi modificate din exteriorul programului, deoarece nu vrem ca rezultatul efectuării unei anumite operații pe polinoame, introduse de o persoană să fie diferit din cauza faptului că altcineva a modificat, pe parcurs, polinoamele respective. Pentru a putea fi totuși folosite acele variabile de instanță și în afara clasei, Monom, în care au fost declarate, am realizat gettere și settere. Un getter este o metoda publică care primește ca și parametru variabila declarată private și o returnează. Metoda fiind publică, se poate apela și automat se face referire la valoarea variabilei respective, fără însă a putea fi modificată din exterior. Setter-ul este o metodă publică, de tip void, deoarece ea primește o variabilă ca și parametru și îi modifică valoarea.

Metoda toString() este realizată pentru a afișa rezultatul efectuării operațiilor pe polinoame. Această metodă, este scrisă astfel încât rezultatul afișat să fie de forma: 2x^3.



Tot în cadrul clasei Monom, este implementată și interfața Comparable, pentru a putea sorta polinomul în funcție de grad.



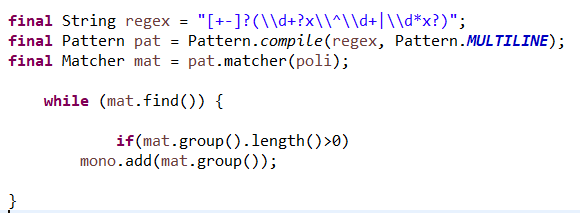
Clasa Polinom

Este caracterizată printr-un ArrayList de monoame. Adică pe fiecare poziție din listă, se află un monom, pe care se vor efectua următoarele operații.

În interiorul acestei clase, am realizat parsarea și operațiile pe polinoame.

În metoda Pattern, a fost realizată împărțirea polinomului în monoame, prin intermediul funcției Regex. Metoda primește ca și parametru, un string, care reprezintă polinomul. Inițial am declarat un String de tip Regex, care reprezenta cum trebuie să arate un polinom. Apoi, Matcher-ul este utilizat pentru a verifica dacă polinomul introdus de noi, este de tipul regex.

Într-un arrayList de string-uri se salveaza fiecare grup de elemente, iar fiecare grup reprezintă un monom.



Pentru a împărți fiecare grup de elemente, într-un monom, am verificat ce tip de monom ar putea fi.

Dacă string-ul respectiv conține si variabilă și “^”, atunci parcurg caracter cu caracter și intr-o variabilă salvez semnul, adica primul caracter din string, dacă caracterul este cifra il salvez intr-un alt string, iar dacă caracterul ajunge la variabila “x”, atunci, salvez valoarea string-ului in care am reținul cifrele, intr-un alt string, pentru a elibera acea variabilă, urmând să se salveze acolo, puterea monomului.

La ieșirea din for-ul pe caractere, verific dacă semnul este -, atunci transform din String în int, cu – în față, dacă nu, fără. Apoi convertesc și puterea în int și adaug valorile în arraylist-ul de monoame.

Dacă string-ul nu îl conține variabila “x”, atunci parcurg din nou, caracter cu caracter, adaug semnul și adaug cifrele. Convertirea este ca înainte, doar că nu va exista putere, deci nu va fi nevoie să mutăm valoarea coeficientului, în alt string.

Dacă string-ul conține variabila “x”, dar nu conține “^”, atunci etapele vor fi aceleași cu cazul trecut, doar că în momentul în care se ajunge la caracterul “x”, se setează puterea ca fiind 1. Apoi se realizează, la fel, conversia.

Adunarea a fost realizată in metoda Sum, care primește ca și parametru un polinom, care reprezintă unul din cele 2 polinoame inițiale. În interiorul metodei, este instanțiat un nou obiect de tip Polinom, în care se va salva polinomul rezultat în urma adunării.

În prima etapă de rezolvare, am parcurs cele două polinoame, prin intermediul a două bucle for each, apoi am verificat dacă gradele anumitor elemente din cele două polinoame, sunt egale, atunci am adăugat în arraylist-ul de monoame al polinomului rezultat, un nou obiect de tip Monom, care primea ca și parametri, un numar, o variabilă și o putere. Pentru coeficient adunam coeficientul termenului cu același grad din primul polinom cu coeficientul termenului cu același grad al celui de-al doilea polinom, variabila și puterea rămâneau neschimbate.

În a doua etapă, pentru a adăuga în rezultat și monoamele care nu au gradele comune, adică nu apar în ambele polinoame, am parcurs tot cu 2 bucle for-each, primul polinom introdus și polinomul rezultat. Am declarat inițial, o variabila val de tip boolean, pe care am folosit-o pentru a verifica gradele și am inițializat-o cu false. Dacă elementele aveau același grad, atunci val lua valoarea false, apoi break, pentru a ieși din buclă, deoarece am tratat deja acel caz în care gradele sunt egale, dacă gradele erau diferite, atunci val ia valoarea true. În afara celui de-al doilea for, am verificat dacă val == true, dacă există în primul polinom, un monom, cu un anumit grad, care nu se găsește în polinomul rezultat, atunci va fi adăugat, în arraylist-ul de monoame al polinomului rezultat. Apoi val ia valoarea false, pentru a putea verifica următoarele monoame.

În a treia etapă, am aplicat același algoritm ca și în etapa doi, doar că a fost aplicat pe cel de-al doilea polinom si polinomul rezultat, pentru a ne asigura că nu au rămas monoame, cu grade diferite, în afara rezultatului.

Metoda returnează tot un Polinom și anume polinomul rezultat, în care au fost salvate monoamele.

Scăderea a fost realizată în metoda Sub, care primește de asemenea ca și parametru un obiect de tip Polinom. În interiorul metodei se instantiază un nou obiect de tip Polinom, în care se va salva polinomul final, rezultat.

Operația de scădere este realizată asemănător cu operația de adunare. În parcurgerea celor 2 polinoame, dacă gradele a două elemente sunt egale, atunci se adaugp în arraylist-ul de monoame al polinomului rezultat, un nou obiect de tip Monom, pentru care, valoarea coeficientului, noului monom, o va lua diferența dintre coeficientul termenului din primul polinom și coeficientul termenului, de același grad, din cel de al doilea polinom.

Verificarea pentru monoamele de grad diferit este la fel ca și pentru operația de adunare.

Înmulțirea a fost realizată în metoda Mul, care primește ca și parametru un obiect de tip Polinom. În interiorul metodei se instantiază un nou obiect de tip Polinom, în care se va salva polinomul final, rezultat.

Înmulțirea propriu zisă a două polinoame, se realizează înmulțind fiecare element din primul polinom, cu fiecare element din cel de-al doilea polinom. Astfel, am folosit 2 bucle for-each pentru ambele polinoame și am înmulțit primul coeficient din al termenului din primul polinom cu coeficientul termenului din cel de-al doilea polinom, iar puterile se aduna. Astfel, în arraylist-ul de monoame rezultat, se va adăuga un obiect nou de tip Monom, în care, pe poziția coeficientului, va fi produsul coeficienților celor două elemente, variabila iar pe poziția puterii, va fi suma puterilor celor două elemente.

Derivarea a fost realizată în metoda Deriv, care primește ca și parametru un obiect de tip Polinom. În interiorul metodei se instantiază un nou obiect de tip Polinom, în care se va salva polinomul final, rezultat.

Derivarea se efectuează pentru un singur polinom. Se parcurge arraylist-ul și există 3 cazuri pe care le-am tratat.

Cazul 1. Dacă puterea monomului respectiv este strict mai mare decât 1, atunci se va adăuga în arraylist-ul de monoame al polinomului rezultat, un nou obiect de tip Monom, în care coeficientul îl reprezintă produsul dintre coeficientul monomului, din polinomul trimis ca și parametru și gradul acestuia, variabila, iar puterea este dată de puterea monomului -1.

Cazul 2. Dacă puterea monomului respectiv este egala cu 1, atunci adăuga în arraylist-ul de monoame al polinomului rezultat, un nou obiect de tip Monom, în care coeficientul rămâne neschimbat, iar puterea este 0. Reprezentând astfel elementul liber, care nu depinde de variabila.

Cazul 3. Dacă puterea monomului este egala cu 0, atunci folosim instrucțiunea “continue;” care are rolul de a trece la instrucțiunea următoare, deoarece dacă puterea este 0, atunci termenul pe care îl derivăm nu depinde de variabilă și automat este 0. Astfel, acest termen, nu va apărea în polinomul rezultat.

Integrarea a fost realizată în metoda Deriv, care primește ca și parametru un obiect de tip Polinom. În interiorul metodei se instantiază un nou obiect de tip Polinom, în care se va salva polinomul final, rezultat.

Integrarea, se realizează tot pe un singur polinom, care se parcurge cu un for each și se disting din nou 2 cazuri.

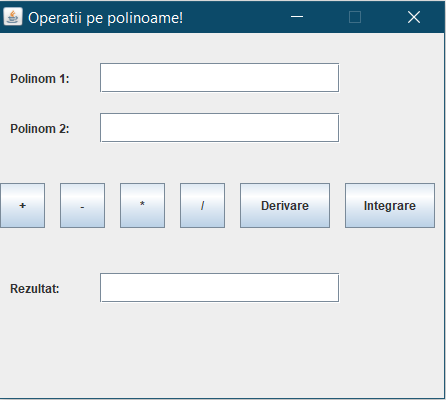
Cazul 1. Dacă puterea este mai mare sau egala ca 1, atunci în arraylist-ul de monoame al polinomului rezultat, se adaugă un nou obiect de tip Monom, în care se seteaza variabila va fiind aceeasi cu cea a polinomului, iar coeficientul și puterea nu se modifică, deoarece este tratat acest caz în pachetul Controller, în clasa integrare.

Cazul 2. Dacă puterea este mai mica ca 1, atunci în arraylist-ul de monoame al polinomului rezultat, se adaugă un nou obiect de tip Monom, în care se seteaza variabila va fiind „x”, deoarece monomul respectiv, nu are variabila, fiind termenul liber, iar coeficientul și puterea nu se modifică, deoarece este tratat acest caz în pachetul Controller, în clasa integrare.

Clasa View

Este o clasa care extinde JFrame, deoarece în interiorul ei se realizează o interfață grafică.

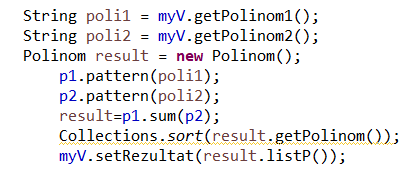
Am declarat un frame, în care am adăugat un panel, câte un textField pentru fiecare polinom, și butoane Jbutton pentru fiecare operație. Am setat pentru frame, fiecare textField si fiecare buton, cate un nume si dimensiunile pe care sa le aiba. Am adăugat pentru fiecare buton câte un ActionListener, care are rolul de a efectua ceva în momentul apăsării butonului.



Pachetul Controller

În Controller, se oferă funcționalitate fiecărui buton, astfel, conține câte o clasa pentru fiecare operație și fiecare implementează interfața ActionListener și fiecare clasă, are metoda void actionPerformed.

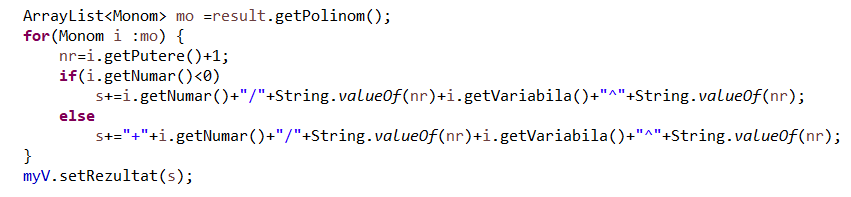
Se declară butoanele și se realizează constructorul.

În clasa Adunare, cele 2 string-uri care provin din cele 2 textField-uri, se transformă în polinoame, prin apelul metodei pattern, apoi se crează un obiect de tip Polinom, in care se salvează suma celor 2 polinoame, prin intermediul interfeței Comparable, sortăm polinomul în funcție de grad, apoi se afișează pe cel de-al treilea textField.

Clasele Scadere și înmulțire sunt exact la fel ca și clasa adunare, singura diferență este că în rezultat se salvează scăderea și respective înmulțirea celor 2 polinoame. În clasa derivare se folosește un singur polinom, în rest este la fel.

În clasa integrare, am parcurs etapele de mai sus, am salvat în polinomul result, rezultatul integraraii, (ne-am asigurat că variabila este pusă corect).

Apoi am declarat un arraylist de monoame, în care am salvat valoarea polinomului result. L-am parcurs, am salvat în nr, valoarea puterii+1 și dacă coeficientul monomului este mai mic ca 0, l-am salvat intr-un string în care am adăugat coeficientul / puterea+1,variabila iar puterea devine putere+1. Iar dacă coeficientul este mai mare ca 0, atunci inainte de a pune coeficientul, adaugă și caracterul „+”.



Clasa Main

Este utilizată pentru apelul metodelor și rularea acestora. În interiorul clasei, am declarat un obiect de tip View, pentru a putea fi vizibilă interfața și un obiect de tip Controller, care are ca și parametri, obiectul de tip view si 2 obiecte de tipul polinom.

Clasa teste

Este folosită pentru testarea operațiilor. Este o clasă de tip JUnit și extinde TestCase. Conține variabile în care se salvează rezultatul pe care ar trebui să îl aibă fiecare operație. Apoi, se realizează câte o metodă, testAdd, testSubs, testMult ,testDeriv, testInteg, în care se instanțiază un obiect de tip polinom, se apelează pattern-ul, apoi în fiecare metodă se apelează metodele din polinom, sum, sub, mul, derivare, integrare iar prin funcția assertEquals, se verifică dacă rezultatul metodelor din polinom este același cu rezultatul declarat ca variabilă de instanță.

În metoda testInteg, se folosește din nou, for-ul din Controller, în care se setează coeficientul și puterea, pentru fiecare monom și se salveaza într-un string, pentru a putea a afișa fracția.

–

1. Concluzii

În concluzie, această aplicație este destinată calculului operațiilor pe polinoame. Calculează adunarea, scăderea, înmulțirea, derivarea și integrarea polinoamelor. Ar putea fi îmbunătățită prin efectuarea împărțirii prin adăugarea mai multor polinoame, sau polinoame cu mai multe variabile.

1. Bibliografie

* Căutare google – diferite informații pentru java, regex și informații despre polinoame.
* Pentru diagrame (use case, uml) – https://www.draw.io