**DOCUMENTATIE TEMA 1**

**CALCULATOR POLINOAME**

**Nume prenume: Dragomir Mihai-Robert**

**Grupa: 302210**

**Profesor Laborator Assist Mitrea Dan**

Contents

[1. Cerinte Functionale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv Principal: 3](#_Toc476131447)

[2.2. Obective Secundare: 4](#_Toc476131448)

[3. Analiza Problemei 4](#_Toc476131449)

[4. Proiectare 5](#_Toc476131450)

[4.1. Structuri de date 5](#_Toc476131451)

[4.2. Diagrama de clase 5](#_Toc476131452)

[4.3. Algoritmi 7](#_Toc476131453)

[5. Implementare 8](#_Toc476131454)

[6. Testare 9](#_Toc476131455)

[7. Concluzii si Dezvoltari Ulterioare 9](#_Toc476131456)

[8. Bibliografie 10](#_Toc476131457)

# Cerinte Functionale

Dezvoltati un calculator de polinoame care sa permita urmatoarele functionalitati:

* Permite utilizatorului inserarea unui polinom de la tastatura sub forma 4X^3+2X^2-6X+1;
* Permite utilizatorului sa selecteze operatia matematica dorita, care sa actionize asupra inputului transmis de acesta;
* Permite adunarea celor doua polinoame date de utilizator drept input;
* Permite scaderea celor doua polinoame date de utilizator drept input;
* Permite inmultirea celor doua polinoame date de utilizator drept input;
* Permite impartirea celor doua polinoame date de utilizator drept input, interpretand deimpartitul drept polinomul de grad mai mare, pentru a evita efectuarea de operatii cu efect minimal, precum impartirea unui polinom de grad mai mic, la un polinom de grad mai mare;
* Permite derivarea celor unui polinom dat de utilizator drept input.
* Permite efectuarea operatiei de integrare nedefinita a unui polinom dat de utilizator drept input, afisand in rezultat si constanta care rezulta in mod inerent in urma procesului de integrare;
* Permite utilizatorului consultarea unui manual de instructiuni, ce are scop informativ, in vederea familiarizarii acestuia atat cu modul in care trebuie introduce datele de intrare de la tastatura, cat si cu modul general de interactiune cu interfata grafica a calculatorului de polinoame;
* Atentioneaza utilizatorul in cazul in care au exitat erori in procesarea datelor de intrare, sugerand consultarea manualului de instructiuni in prealabil introducerii unui viitor input;
* Atentioneaza utilizatorul in cazul in care au exitat erori in performarea unei operatii solicitate, spre exemplu o incercare indezirabila de impartire a unui polinom la 0.

# Obiective

## Obiectiv Principal:

Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila, cu coeficienti intregi, care beneficiaza de o interfata grafica dedicata, prin intermediul careia un utilizator poate introduce polinoame de la tastatura (in conformitate cu un format prestabilit), poate selecta operatia matematica pe care calculatorul de polinoame sa o performeze asupra datelor de input si poate sa vizualizeze rezultatul acesteia.

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenario | Prezinta o reprezentare grafica a modului in care utilizatorul interactioneaza cu interfata grafica a sistemului de calcul, urmarind anumite spete si scenarii de functionare si utilizare. | 3 |
| Alegerea structurilor de date | Ofera explicatii cu privire la optiunea pentru anumite structuri de date in implementarea solutiei. | 4 |
| Impartirea pe clase | Urmareste din puct de vedere schematic modul de interactiune si asamblare al claselor in vederea functionarii corecte a calculatorului de polinoame. | 4 |
| Dezvoltarea algoritmilor | Aduce adnotari imperios necesare asupra modului in care sunt traduse in practica, sub forma algoritmica, anumite metode auxiliare sau fundamentale in implementarea sistemului de calcul. | 4 |
| Implementarea solutiei | Realizeaza o introspectie asupra functionalitatii si a necesitatii fiecarei clase constituente. | 5 |
| Testare | Presupune testarea corectidunii rezultatului provenit in urma aplicarii diferitelor operatii asupra unor polinoame date drept input. | 6 |

# Analiza Problemei

Diagrama de use case case:

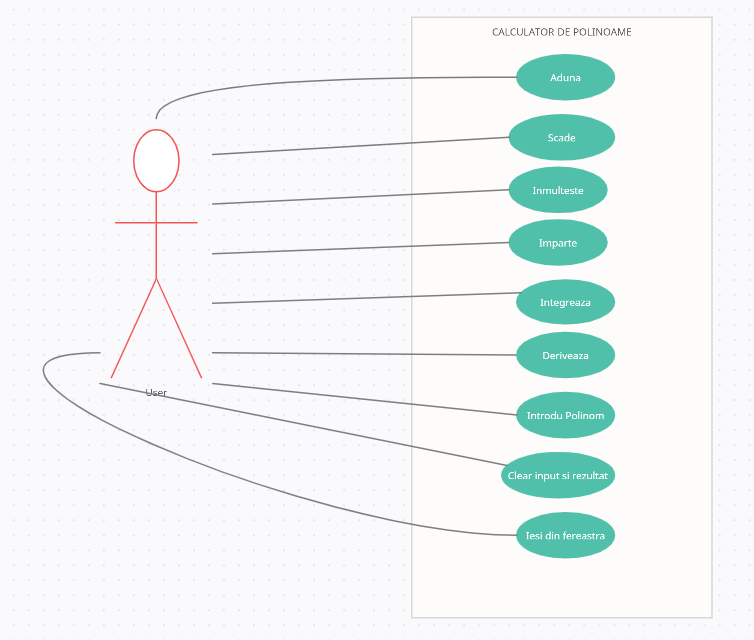


Diagrama use case aferenta operatiei de adunare:

Use Case: aduna polinoame

Actor principal: user

Principalul scenariu

1. Utilizatorul insereaza 2 polinoame in interfata grafica;

2. Utilizatorul selecteaza operatia de adunare;

3. Calculatorul de polinoame calculeaza suma celor 2 polinoame si afiseaza rezultatul;

Scenariu alternativ

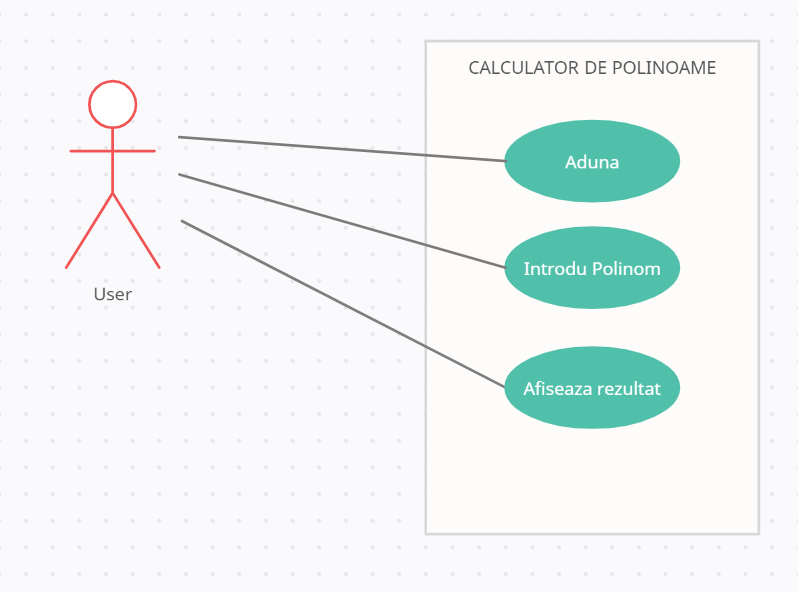
Polinoame incorecte

-

Utilizatorul introduce in mod incorect polinoamele.

-

Se realizeaza redirectionarea catre primul pas.



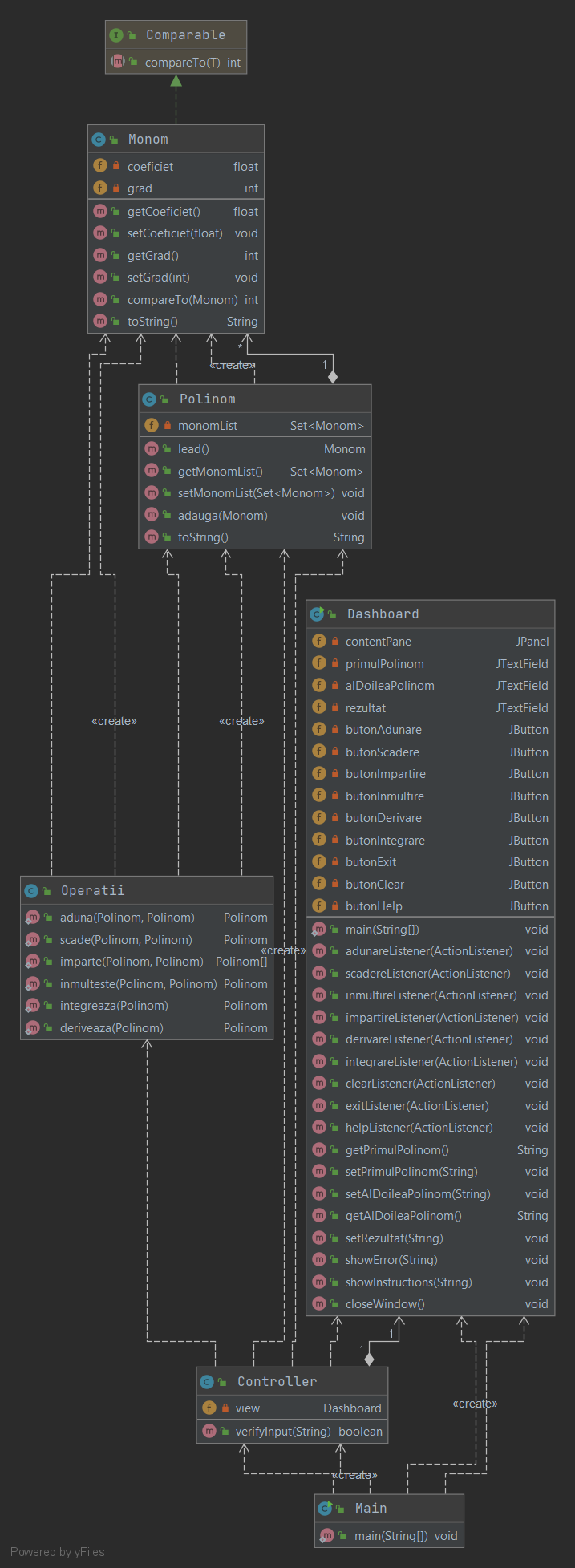
# Proiectare

## Structuri de date

Structura de data fundamentala in vederea implementarii solutiei finale o constituie TreeSet-ul, drept atribut singular in clasa “Polinom”, care are elemente constituente, obiecte de tip “Monom”. Optiunea pentru TreeSet, in detrimental unui Array pentru coeficienti, isi are sorgintea in redundanta stocarii datelor pe care ar fi implicat-o o structura de tip Array. Astfel, daca extrapolam notiunea de matrice rara la nivel de array-uri, am putea referi un astfel de Array de Monoame drept un “array rar”, care pentru anumite input-uri ar fi abundant inutil de zero-uri, cee ace ar fi determinat in fapt o irosire si o utilizare deficitara a memoriei. De asemenea, optiunea pentru TreeSet in detrimental unui ArrayList, este justificata prin caracteristicile intrinseci ale primei structure mentionate. Astfel, desi deficitar in implementarea anumitor metode, unde ArrayList-ul ar putea fi mai benefic, ordonarea implicita a elementelor in momentul adaugarii intr-un TreeSet, l-a facut pe acesta din urma mai dezirabil.

## Diagrama de clase

Diagrama UML din randurile ce urmeaza descrie modul in care clasele utilizate in implementarea sistemului de calcul interactioneaza intre ele.



## Algoritmi

Intr-o prima instanta, sistemul de calcul polinomial trebuie sa aiba capacitatea de a interpreta corect datele introduse de la tastatura de utilizator si de a realiza transformarea acestora intr-un obiect de tip Polinom. Aici rolul fundamental il joaca clasa Regex, care, prin utilizarea unui sablon, bazat pe ideea de “expresii uzuale”, realizeaza, in cadrul metodei “covertInputToPolynom”, o impartire a datelor de intrare in grupuri de monoame. Ulterior, pentru determinarea coeficientului si a puterii, astfel incat sa poata fi creat un obiect de tip Monom si adaugat in structura de polinom, fiecare grup in parte este prelucrat in metoda “convertInputToMonom”. Aceasta analizeaza toate spetele de reprezentare a unui Monom, tratand fiecare caz special (ex: coeficientul = 1 sau puterea =1) si returnand obiectul Monom caracterizat prin coeficient si putere.

Monomul, astfel obtinut, trebuie adaugat in lista de monoame a polinomului. Algoritmul care implementeaza aceasta operatie este transpus la nivel de cod in metoda “adauga” din clasa Polinom. Astfel, este creata o structura noua, auxiliara de TreeSet.

Daca in lista de monoame a polinomului exista deja un monom de acelasi grad cu cel care urmeaza sa fie adaugat, atunci sunt parcursi termenii rand pe rand si adaugati in lista auxiliara de monoame. In momentul in care este gasit totusi monomul de grad egal cu cel ce urmeaza sa fie adaugat, in lista auxiliara este adaugat un monom ce are drept coeficient suma coeficientilor celor doua monoame mai sus mentionate. Daca suma respectiva este egala cu 0, lista auxiliara nu va mai contine niciun monom de grad egal cu gradul polinomului care urma sa fie adaugat, deoarece acesta, adunat cu monomul de acelasi grad existent deja in structura se anuleaza reciproc. Finalmente, in aceasta speta, lista de monoame a polinomului devine lista auxiliara creata.

In cealalta speta, in care in lista de monoame a polinomului nu mai exista alt monom de acelasi grad cu cel al monomului care trebuie adaugat, acesta din urma este adaugat ca atare in setul de monoame, uzand de metoda predefinita de “add” a structurii de Set.

Operatia de adunare a doua monoame descrisa la nivel de limbaj Java in clasa Operatii, cu ajutorul metodei “aduna”, ce primeste drept parametri doua polinoame, implica un algoritm bazat pe modul de adaugare al monoamelor descries anterior. Astfel, sunt parcurse iterative monoamele din lista celui de-al doilea polinom si inserate pe baza metodei adauga intr-o lista auxiliara echivalenta cu lista de monoame primului polinom. Lista auxiliara, denumita in cod, in mod intuitiv “rezultat” va fi in final returnata.

Operatia de scadere implica o implementare asemanatoare cu cea a operatiei de adunare, doar ca de aceasta data, monoamele din lista de polinoame ale celui de al doilea polinom sunt adaugate in rezultat, drept obiecte noi de tip Monom, care pastreaza puterea initiala si au coeficientul egal cu coeficientul monomului initial cu semn schimbat. Astfel, la nivel ideatic, am putea spune ca adunam pe rand la polinomul initial, fiecare monom din al doilea polinom, cu semn schimbat.

Operatia de inmultire, constituie in fapt inmultirea fiecarui monom din primul polinom, cu fiecare monom din cel de-al doilea polinom, rezultand un monom care are drept coeficient produsul coeficientilor celor doua monoame anterior mentionate si drept putere, suma puterilor acestora. Fiecare astfel de monom obtinut este inserat in lista de monoame a polinmulului care reprezinta rezultantul, tot prin intermediul metodei “.adauga()”.

Operatia de impartire verifica initial care dintre cele doua polinoame are gradul mai mare in vederea stabilirii deimpartitului (P) si a impartitorului (Q). Tinand cont ca in cadrul polinoamelor monoamele sunt ordonate in ordine descrescatoare gradului, se imparte initial primul monom din P la primul monom din Q si se obtine astfel primul element al catului si se adauga in lista acestuia din urma. Apoi catul este inmultit cu Q, iar din P se scade rezultatul inmultirii, rezultand astfel restul impartirii la acest moment. Apoi se repeta pasii considerand la iteratia urmatoare P-ul( deimpartitul) drept restul de la iteratia anterioara, pana cand gradul restului devine mai mic decat decat gradul lui Q(gradul impartitorului), sau pana cand restul ajunge egal cu 0.

Operatia de integrare presupune parcurgerea fiecarui monom din lista polinomului dat drept input si adaugarea intr-o lista de monoame de rezultat a monomului de coeficient egal cu coeficientul initial impartit la valoarea puterii initiale adunate cu unu, respective de putere egala cu valoarea puterii initale la care se adauga unu.

Operatia de derivare presupune, de asemenea, parcurgerea fiecarui monom din lista polinomului, dar, de aceasta data, in lista de monoame de rezultat este adaugat un monom care are coeficientul egal cu coeficientul anterior inmultit cu puterea initiala si puterea, egala cu puterea initiala din care se scade valoarea unu. Daca monomul care urmeaza sa fie derivate este un monom de grad 0, adica o constanta, acesta nu mai este prelucrat pentru a fi adaugat in lista finala, intrucat prin derivare, valoarea sa va fi indubitabil 0.

# Implementare

Basorelieful architectural MVC (Model – View - Controller) se contureaza prin modelul logic epistemic de impartire a claselor in pachete. Astfel, clasele Monom, Polinom, Operatii si Regex se subscriu pachetului model, care manipuleaza toate datele si operatiunile logice si de utilizare de informatie. Clasa Dashboard se subscribe pachetului view, care contureaza reprezentarea grafica a sistemului de calcul si exprima forma finala a datelor. Clasa Conroller face parte din pachetul cu acelasi nume, gestionand comunicatia dintre model si view si controleaza accesul la aplicatie. Astfel, acesta se transpune din punct de vedere logic, in postura unui canal de comunicare.

***Clasa Monom*** are drept atribute un intreg reprezentat de grad/putere si un numar real, de tip float, care constituie coeficientul. In momentul insantierii unui monom pe care se doresc prelucrari ulterioare, constructorul seteaza atat gradul cat si coeficientul. Constructorul implicit, reiterat la nivelul clasei, isi are utilitatea in vederea declararii de obiecte de tip Monom vide, utilizat drept auxiliare in metode externe. La nivelul clasei am suprascris metoda toString() pentru a putea avea o reprezentare dezirabila la nivel de String a unui monom, utila in reprezentarea unui polinom. Totodata, am suprascris metoda compareTo(), pentru a avea un resort care sa reactioneze in vederea sortarii monoamelor in ordine descrescatoare gradelor, stabilind astfel relatia de ordine dorita.

***Clasa Polinom*** are drept atribut singular un set de monoame, alegere justificata la nivelul capitolului 4, Sectiunea 4.1. Am suprascris si la nivelul acestei clase metoda toString() pentru o afisare convenabila a unui polinom, bazata pe modul de afisare al unui monom. Metoda adauga(), omiprezenta si fundamentala in metode si operatii externe, permite adaugare unui monom intr-un polinom, care din punct de vedere prologar se constituie drept un set de monoame. Modul de functionare al acesteia este descris in captitolul 4, sectiunea 4.3. Metoda lead() returneaza primul Monom din lista de polinoame, avand semnificatia de monom de grad maxim.

***Clasa Operatii*** descrie prin metode statice, utilizate pentru ca apelul lor sa nu implice instantierea unui obiect de tip Operatii, modul de functionare al operatiilor pretinse: Adunare, Scadere, Inmultire, Impartire, Integrare, Derivare, returnand obiecte de tip Polinom. Functionalitatea fiecareia dintre acestea este explicate pe larg in capitolul 4, Sectiunea 4.3.

***Clasa Regex*** este indispensabila in vederea prelucrarii input-ului, astfel incat acesta sa poata fi modelat sub forma unui obiect de tip Polinom, oferind un sablon general caruia orice obiect de acest gen ar trebui sa se subscribe. Atat metodele principale, cat si functionalitatea lor, sunt descries pe larg in capitolul 4, Sectiunea 4.3.

***Clasa Dashboard*** inglobeaza principalele elemente de interfata grafica si descrie in cadrul constructorului sau modul de constructie al interfetei cu care interactioneaza utilizatorul. Atributul contentPane reprezinta panoul principal asupra caruia a fost aplicat un FlowLayout, in cadrul caruia sunt dispuse panourile secundare. In ceea ce priveste metodele utilizate, au fost implementate functionalitati ce permit adaugarea de ascultatori pe butoane, respective metode de getter si setter asupra input-urilor si a TextField-ului unde este afisat rezultatul operatiei. Totodata au fost create metode al caror rol rezida in deschiderea unor ferestre de dialog cu utilizatorul.

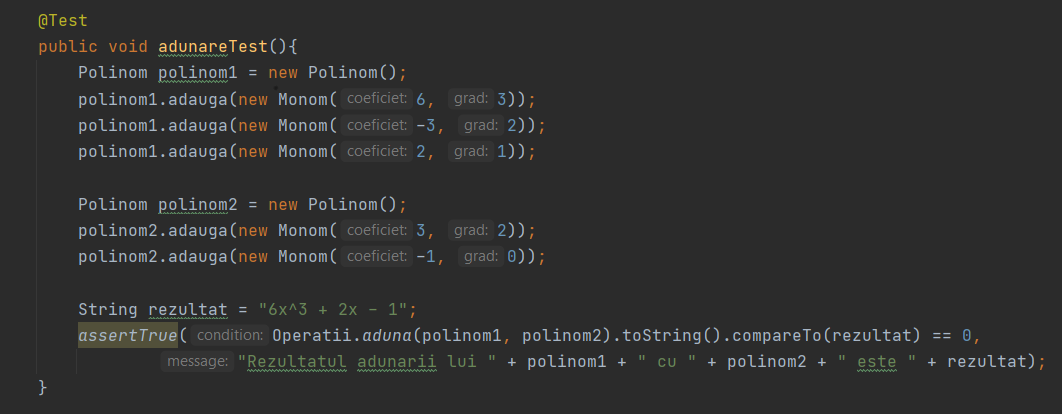
***Clasa Controller*** se manifesta drept utilitar care intermediaza modul de interactiune dintre view si model. La nivelul sau au fost adaugati Listener-i pe cele 9 butoane si s-a descries modul de actiune prorpiu in cazul fiecarei actiuni care se rasfrange asupra unui anumit buton. Tot la nivelul clasei a fost creata o metoda ( verifyInput() ) care verifica daca input-ul transmis de utilizator corespunde modului predefinit de inserare a unui polinom.

***Clasa Main*** instantiaza un obiect de tip Dashboard pe care il seteaza drept vizibil, respective un obiect de tip Controller, facand astfel posibila functionalitatea sistemului de calcul polinomial proiectat.

# Testare

In clasa OperatiiTest am optat spre testarea prin JUnit a tuturor operatiilor pe care sistemul de calcul ar trebui sa fie capabil sa le performeze din punct de vedere functional. In urma rularii, cele 6 teste, corespunzatoare celor 6 operatii au trecut cu success.

Exemplu de test din cadrul clasei OperatiiTest:



# Concluzii si Dezvoltari Ulterioare

Finalmente, sistemul de calcul polinomial proiectat isi dovedeste utilitatea prin interfata intuitiva si prin acuratetea executiei celor 6 operatii matematice asupra polinoamelor introduce de la tastatura. Acesta poate constitui o solutie si implicit un substituent viabil pentru realizarea calculelor pe hartie, care presupun o durata mai indelungata de timp, iar in cadrul procesului pot interfera erori umane de calcul care ar vicia rezultatul final.

Ca dezvoltari ulterioare ce pot fi aduse calculatorului de polinoame, ar putea fi capacitatea acestuia de calcul a radacinilor unui polinom transmis drept input, folosind un pattern din punct de vedere algorithmic bazat pe metoda lui horner. De asemena, calculatorul ar putea sa opereze cu mai multe variabile, sau la nivel minimal cu polinoame de o singura variabila al carei nume sa fie la latitudinea utilizatorului. Tot in scop matematic, dupa dezvoltarea unei metode de determinare a radacinilor, aceasta functionalitate ar putea constitui baza spre extinderea catre determinare unor solutii comune intre 2 polinoame.

# Bibliografie

1. <http://tynerblain.com/blog/2007/04/09/sample-use-case-example/>
2. <https://regex101.com/>
3. <https://www.w3schools.com/java/java_regex.asp>
4. <https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html>
5. <https://www.inflectra.com/ideas/topic/use-cases.aspx>
6. <https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression>
7. <https://ro.wikipedia.org/wiki/Model-view-controller>
8. <https://www.yiiframework.com/doc/guide/1.1/ro/basics.mvc>
9. <http://coned.utcluj.ro/~marcel99/PT2021/Model%20Documentatie/>
10. <https://stackoverflow.com/questions/5283444/convert-array-of-strings-into-a-string-in-java/5283753>