

Universitatea Tehnică Cluj-Napoca

Tema 1

Calculator Polinomial

Student: Stroe Mădălina Ionela

Grupa: 302210

CUPRINS

- ☐ Introducere
- ☐ Obiectivul temei
- ☐ Analiza problemei, modelare, scenarii, cazuri de utilizare
- ☐ Proiectare
- ☐ Implementare
- ☐ Rezultate
- ☐ Concluzii
- ☐ Bibliografie

o Introducere

În matematică, un **monom** reprezintă o expresie algebrică compusă dintr-un singur termen, în care nu figurează nici semnul plus, nici semnul minus, dar în care intervin semnele de înmulțire și împărțire.

Un polinom este construit din mai multe monoame, alcătuite dintr-o constantă, numită coeficient, înmulțită cu una sau mai multe variabile. Fiecare variabilă poate avea un exponent constant întreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile în monom.

Un polinom este o suma de unul sau mai multe monoame.

Exemplu polinom: $-5x^2+13x^1+2x^0$

o Obiectivul temei

Obiectivul principal al acestei teme este de a realiza un calculator care să poată realiza diferite operații pe polinoame, cum ar fi adunarea polinoamelor, scăderea polinoamelor, înmulțirea polinoamelor, derivarea polinoamelor și integrarea polinoamelor, în funcție de ceea ce utilizatorul va alege.

Obiectivele secundare ale acestei teme sunt constituite de aprofundarea conceptelor OOP, precum Încapsularea, Abstractizarea, Polimorfismul, Moștenirea, etc., reamintind, de asemenea, și de câteva concepte matematice.

De asemenea, ca obiectiv secundar putem considera și familiarizarea programatorului cu Model-View-Controller(MVC), un concept foarte răspândit în programare, care are drept scop protecția aplicației/programului, astfel încât, în cazul în care se realizează modificări în aplicație / program, fără să fie afectate alte părți.

o Analiza problemei

Presupunem ca primim de la utilizator 2 polinoame de tipul

$$a_1x^{p_1}+a_2x^{p_2}+\dots+a_nx^{p_n}.$$

Pentru a realiza operatia de *adunare*, se vor introduce explicit atât coeficientul, cât și gradul monomului, fie ele 0, 1 sau altele, in vederea asigurării corectitudinii operației

Pentru a realiza operatia de *scădere*, se vor introduce explicit atât coeficientul, cât și gradul monomului, fie ele 0, 1 sau altele, in vederea asigurării corectitudinii operației

Pentru aceste 2 operații, se va păstra gradul monoamelor, iar coeficienții se vor aduna sau se vor scădea, după caz, afișându-se și semnul corespunzător.

De reținut este faptul că un polinom este o sumă de monoame.

În cazul *înmulțirii*, fiecare monom din primul polinom se va înmulți cu fiecare monom din al doilea polinom, ceea ce duce la înmulțirea coeficienților și creșterea gradului monomului rezultat. Monoamele de același grad rezultate se vor aduna, pentru a obține un polinom cu monoame de grade diferite.

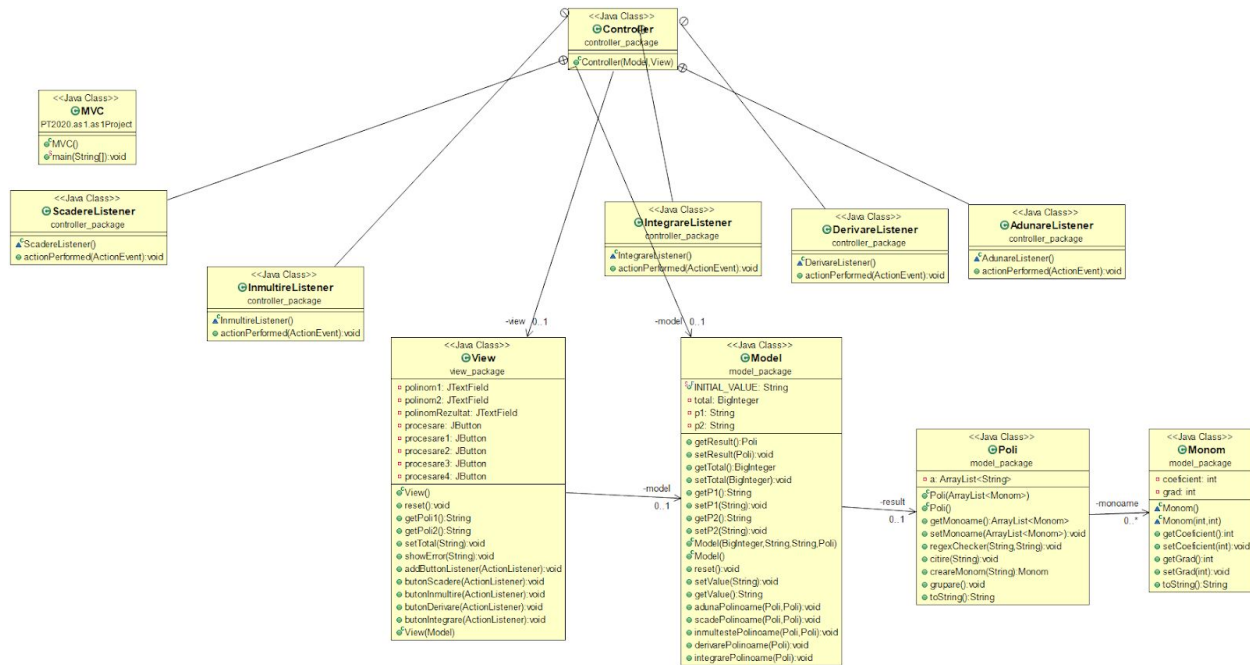
În cazul *derivării*, aceasta se va face după formula matematică: puterea se înmulțește cu coeficientul, rezultatul devine coeficient, iar puterea lui x va deveni p-1.

În cazul *integrării*, coeficientul va deveni coeficient/putere, iar puterea va crește cu 1.

De reținut este că în cazul derivării și integrării vom folosi doar primul polinom.

Toate cele de mai sus, matematic vorbind. În ceea ce privește proiectarea, pentru realizarea unui flux de primire-prelucrare-returnare a informației simplu și accesibil, se va încerca organizarea într-un mod eficient a claselor pe care le vom crea pentru Calculatorul de Polinoame.

o Proiectare



Pentru realizarea acestui proiect, am încercat să respect normele MVC(Model-View-Controller), care presupune următoarele:

Model: Asigura update-ul logic pentru Controlle în cazul în care datele se schimbă.

View: Modul de vizualizare a datelor conținute de Model.

Controller: Păstrează View și Model separate, dar asigură funcționalitatea împreună. Controlează datele ce intră în model și actualizează View atunci când acestea se schimbă.

Am structurat programul în cele trei pachete, fiecare pachet conținând una sau mai multe clase,

Pachetul `view_package` conține Clasa `View`, pachetul `model_package` conține clasele `Model`, `Monom`, `Poli`, pachetul `Controller` conține clasa `Controller`.

Așadar, după cum am spus mai sus, de primirea informației este responsabilă partea de `View` a modelului architectural `Model – View – Controller`.

De partea de prelucrarea a informației se ocupă partea de `Model` a modelului architectural `MCV`.

Pentru punerea în mișcare a celor două componente mai sus menționate, este necesară partea de `Controller`. Într-adevăr, `Model` se ocupă de stocarea și prelucrarea informației, însă fără `Controller` care asigură legătura între `View` și `Model`, nu ar exista un program functional, cel mai probabil.

Design-ul interfeței utilizator a fost dat, de fapt, de nevoia pe care ar avea-o utilizatorul. Să presupunem că datele vor fi preluate de la utilizator cu ajutorul unor căsuțe text și trimise mai departe pentru prelucrare.

Utilizatorul nu are foarte multe opțiuni, însă și cele existente trebuie a fi bine cântărite și gândite pentru implementare.

Pentru fiecare dintre aceste opțiuni, am creat un buton: `Adunare`, `Scădere`, `Înmulțire`, `Derivare` și `Integrare`.

Pentru încercarea de a păstra programul cât mai simplu și ușor accesibil, după realizarea operației selectate, într-un alt `TextField` va fi afișat rezultatul sub forma unui Polinom, după ce a trecut prin prelucrările necesare pentru a îndeplini cerința utilizatorului.

o Implementare

În primul rând, avem clasa **Monom**, cu constructor cu parametrii: Monom(int coeficient, int grad). Avem getter si setter pentru coeficient si grad si o metoda de toString().

Pentru clasa **Poli** avem constructor având ca parametru o listă de monoame. Mai departe avem getter si setter pentru lista de monoame de tip ArrayList si o metodă de toString(). Alături de acestea, pentru separarea polinomului în monom, am creat o metodă regexChecker() cu 2 parametri de tip String: theRegex si str2Check. Primul parametru reprezintă expresia regulate care definește un șablon de căutare. Al doilea parametru este reprezentat de un polinom. Rolul acestei metode este de a separa polinomul în monoame, pe care le va adauga în lista de monoame.

De asemenea, metoda createMonom() care are ca parametru un String, returneaza un monom, după ce setează coeficientul si gradul monomului nou, în functie de stringul dat ca parametru.

Metoda de grupare este folosită pentru a grupa termenii după grad, în cazul înmulțirii,

pentru a nu exista mai mulți termeni de același grad în polinomul rezultat. Presupunem

exemplul următor: $1x^2 + 2x^1 + 3x^0$ și $2x^1 + 1x^0$. Rezultatul ar fi $+ 2x^3$

$+ 1x^2 + 4x^2 + 2x^1 + 6x^1 + 3x^0$. După gruparea termenilor vom avea ca rezultat $+ 2x^3$

$+ 5x^2 + 8x + 3$.

Putem spune despre clasa **Model** că aceasta este cea care oferă implementarea operațiilor matematice: adunare, scădere, înmulțire, derivare, integrare. Conține getter și setter pentru variabilele instanță și constructor cu parametri, iar în constructorul fără parametri avem funcția de reset() ce presupune că în casuța de afișare a polinomului rezultat, va exista o valoare implicită.

Metoda adunaPolinoame(Poli p1, Poli p2), după cum îi spune și denumirea, are rolul de a aduna 2 polinoame, care au același grad și coeficienți explicit introduși de utilizator.

Modul de implementare este următorul: Parcurgem listele de monoame ale fiecarui polinom, având creat deja un Polinom rezultat. Pentru fiecare monom vom aduna coeficienții, atribuind ulterior coeficientul și gradul unui Monom auxiliar.

Acest monom va fi ulterior adăugat într-o listă rezultat pe care o vom seta Polinomului rezultat ce urmează a fi afișat.

Metoda `scadePolinoame(Poli p1, Poli p2)`, scade 2 polinoame, care au același grad și coeficienți explicit introduși de utilizator. Modul de implementare este următorul:

Parcurgem listele de monoame ale fiecarui polinom, având creat deja un Polinom rezultat. Pentru fiecare monom vom scadea coeficienții, atribuind ulterior coeficientul și gradul unui Monom auxiliar.

Acest monom va fi ulterior adăugat într-o listă rezultat pe care o vom seta Polinomului rezultat ce urmează a fi afișat.

În cazul metodei `împartePolinoame(Poli p1, Poli p2)`, sunt parcurse listele de monoame ale fiecărui polinom. Monomul auxiliar va avea gradul egal cu suma gradelor celor 2 monoame initiale, iar coeficientul va fi egal cu produsul coeficientilor celor 2 monoame. Acest monom va fi ulterior adăugat într-o listă rezultat pe care o vom seta Polinomului rezultat ce urmează a fi afișat.

În cazul derivării polinoamelor, vom parcurge lista de monoame a primului polinom. Având un Monom auxiliar creat, care va fi ulterior adăugat în lista Polinomului, îi atribuim acestuia drept coeficient produsul dintre gradul și coeficientul său, puterea scăzând cu o unitate.

Acest monom va fi ulterior adaugat într-o listă rezultat pe care o vom seta Polinomului rezultat ce urmează a fi afișat.

În cazul integrării polinoamelor, vom parcurge lista de monoame a primului

Polinom, având un Monom auxiliar creat, care va fi adăugat ulterior în lista Polinomului rezultat.

Coeficientul va avea valoarea raportului dintre coeficient și $\text{grad}+1$, iar valoarea gradului va devenit $\text{grad} + 1$.

Acest monom va fi ulterior adaugat într-o listă rezultat pe care o vom seta Polinomului rezultat ce urmează a fi afișat.

Clasa **Controller**, după cum poate și numele sugera, a fost creată cu scopul de a controla funcționalitatea programului. Este cea care face legăturile între clasa View despre care vom vorbi ulterior și clasa Model.

Ca variabile de instanță avem `private Model model` și `private View view`.

De reținut este că în acest program, clasa View este cea care se ocupă de interfața grafică, ușurând accesul utilizatorului la program.

Această clasă ajută la implementarea într-un mod mai simplu a event handle-urilor, cu ajutorul Action Listener. Un eveniment are loc, oricând o acțiune este realizată de utilizator. În cazul adunării, utilizatorul va introduce 2 polinoame și va apăsa pe butonul de adunare, ceea ce va genera un action Event. Ce se întâmplă mai departe este că vom citi în 2 variabile de tip String

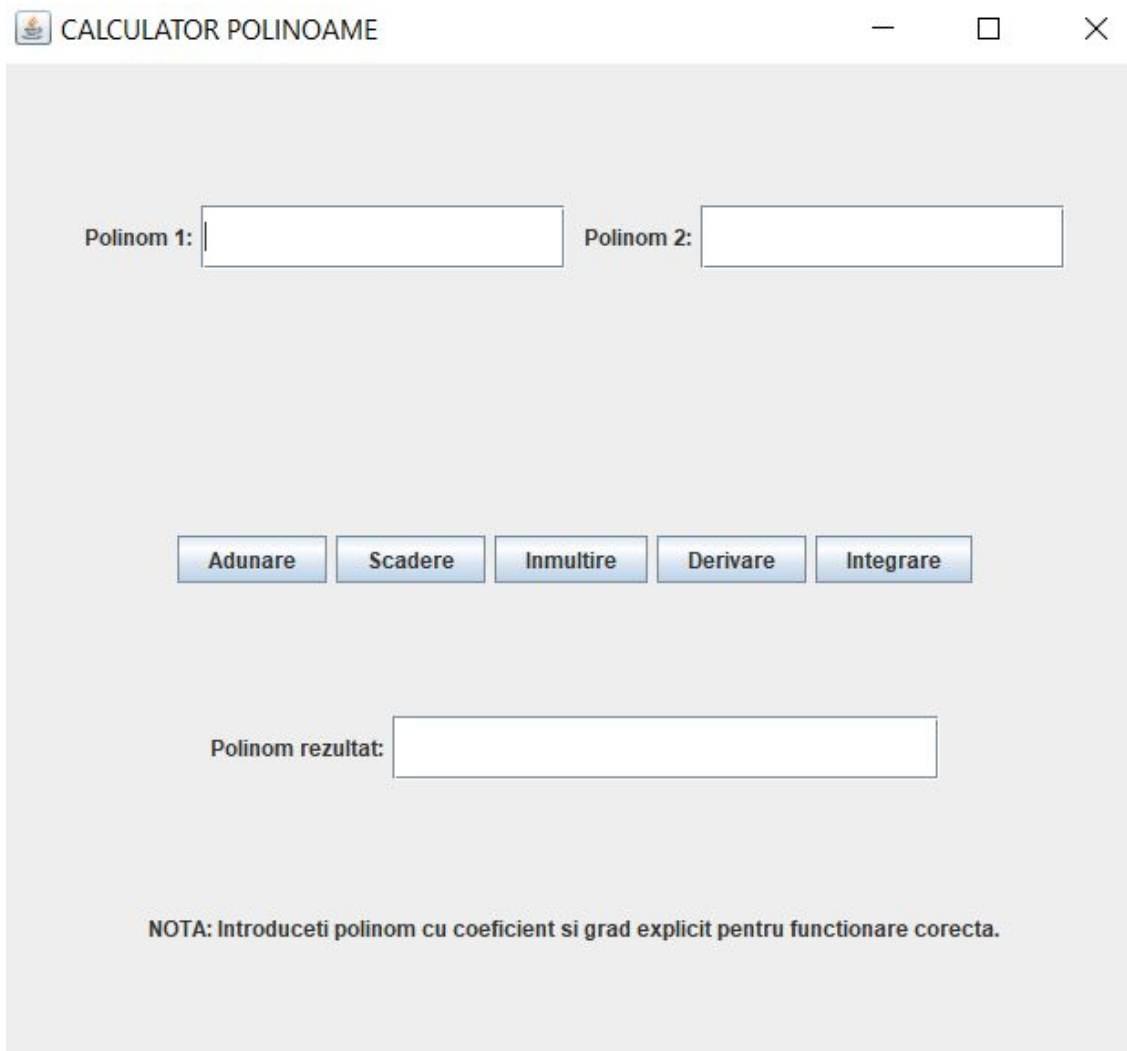
valorile din căsuțele text. Verificăm apoi dacă s-au introdus valori nule, iar în cazul în care nu s-au introdus, vom crea polinoamele cu ajutorul funcției de `regexChecker`.

Urmează apoi să realizăm operația de adunare, al cărui rezultat va fi trimis pentru afisare funcției de `setTotal()` din View, sub formă de String.

Același procedeu este realizat și atunci când sunt apăstate butoanele de scădere și înmulțire.

Deși, putem spune că este asemănător ceea ce se întâmplă când apăsăm butoanele de derivare și integrare, diferența constă în faptul că de această dată ne putem rezuma la scrierea unui singur polinom asupra căruia să fie făcute aceste două operații.

După cum spuneam mai sus, clasa **View** este cea care se ocupă de :



CALCULATOR POLINOAME

Polinom 1: Polinom 2:

Adunare Scadere Inmultire Derivare Integrare

Polinom rezultat:

NOTA: Introduceți polinom cu coeficient și grad explicit pentru funcționare corectă.

Am folosit 3 TextField-uri care ajută la preluarea și afișarea Stringurilor, 5 Butoane care asigură funcționalitatea programului, în funcție de ceea ce dorește utilizatorul să se întâmple cu polinoamele introduse, și 3 Labels, pentru afișarea unor informații suplimentare, cu scopul de a aduce lămuriri.

o Rezultate

Aici vom vorbi despre corectitudinea rezultatelor pentru fiecare operație.

Adunarea a două polinoame introduse cu coeficienți și grade explicite. Rezultatul va fi un polinom cu coeficienți și grade explicite.

The screenshot shows a window titled "CALCULATOR POLINOAME". It has two input fields for polynomials. "Polinom 1:" contains the text $-5x^3+2x^2+0x^1+2x^0$. "Polinom 2:" contains the text $8x^3+5x^2+2x^1+0x^0$. Below these are five buttons: "Adunare", "Scadere", "Inmultire", "Derivare", and "Integrare". The "Adunare" button is highlighted. Below the buttons is a label "Polinom rezultat:" followed by a text box containing $+3x^3 + 7x^2 + 2x + 2$. At the bottom, a note reads: "NOTA: Introduceți polinom cu coeficient și grad explicit pentru funcționare corectă."

Scăderea a două polinoame introduse cu coeficienți și grade explicite. Rezultatul va fi un polinom cu coeficienți și grade explicite.

The screenshot shows the same "CALCULATOR POLINOAME" window. The input fields for "Polinom 1:" and "Polinom 2:" are the same as in the previous image. The "Scadere" button is now highlighted. The "Polinom rezultat:" text box now contains $+13x^3 + 3x^2 + -2x + 2$. The note at the bottom remains the same.

Înmulțirea a două polinoame introduse cu coeficienți și grade explicite. Rezultatul va fi un

polinom cu coeficienți și grade explicite.

CALCULATOR POLINOAME

Polinom 1: Polinom 2:

Adunare Scadere **Inmultire** Derivare Integrare

Polinom rezultat:

NOTA: Introduceți polinom cu coeficient și grad explicit pentru funcționare corectă.

Derivarea primului

polinom:

CALCULATOR POLINOAME

Polinom 1: Polinom 2:

Adunare Scadere Inmultire **Derivare** Integrare

Polinom rezultat:

NOTA: Introduceți polinom cu coeficient și grad explicit pentru funcționare corectă.

Integrarea primului polinom:

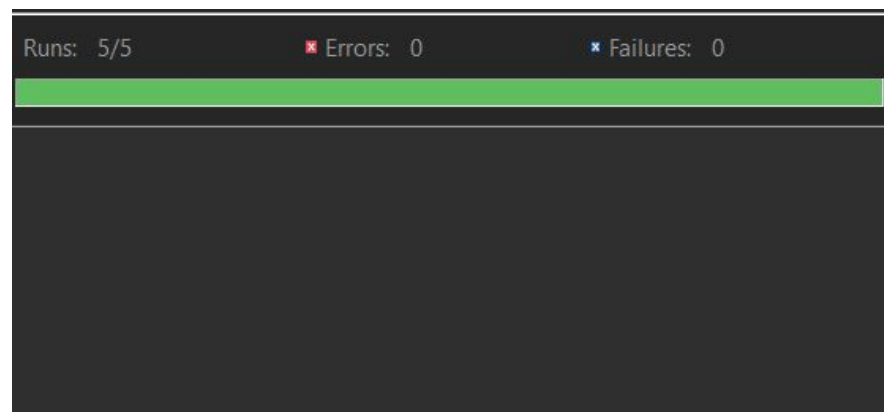
CALCULATOR POLINOAME

Polinom 1: Polinom 2:

Polinom rezultat:

NOTA: Introduceti polinom cu coeficient si grad explicit pentru functionare corecta.

Rezultate Junit:



o Concluzii

În urma realizării acestei teme, am aprofundat conceptele POO. Totodata, am învățat să mă organizez mai bine, poate să lucrez cu lucruri pe care le-am tratat superficial până acum (MVC, pentru că mi se parea greu de înțeles la început). Am înțeles mai bine modul cum folosim componentele în interfața grafică, setarea lor pe planșă, dimensiune, lizibilitate, cum devin ele funcționale.

Ca dezvoltări ulterioare, aş vedea:

- Afișarea cu 2 zecimale în cazul împărțirii coeficientului monomului, modificând coeficientul la tip double.
- Adăugarea operației de înmulțire, precum și a altor operații, eventual, a graficelor celor 2 ecuații polinomiale.
- Personalizarea interfeței utilizator.
- Crearea unei expresii regulate mai permissive
- Detectarea input-urilor în format greșit

o Bibliografie

<https://ro.wikipedia.org/wiki/Polinom>

https://www.youtube.com/watch?v=s_PfopWcMwI&t=52s

<https://www.youtube.com/watch?v=dTVVa2gfht8&t=299s>

<https://www.youtube.com/watch?v=mH1TltI61yU>

<https://stackoverflow.com/questions/18872145/how-do-i-access-the-next-element-in-for-each-loop-in-j>

[ava](#) și altele