**Calculator de polinoame**

Documentatie

Cîrciu Mihnea Teodor

Grupa 30227

**Cuprins**

1. Obiectiv
2. Analiza problemei
3. Implementare
   1. Diagrama UML
   2. Clase
   3. Metode si impementare
   4. GUI
4. Testare
5. Concluzii
6. Bibliografie
7. **Obiectiv**

Obiectivul acestei aplicatii este de a implementa un calculator care sa efectueze asupra unui sau asupra a doua polinoame urmatoarele operatii: adunare, scadere, inmultire, impartire, integrare si derivare.

Introducerea datelor si rezultatele primite de utilizator se vor efectua printr-o interfata grafica.

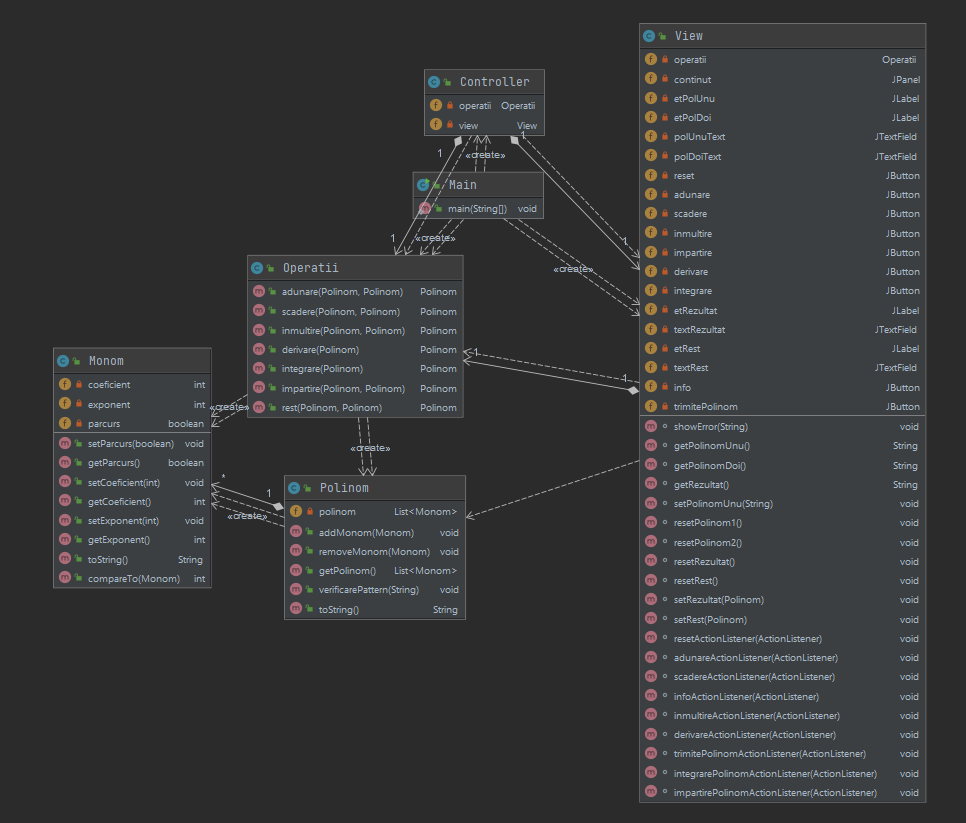
Coeficientii polinoamelor sunt considerati numere intregi, de asemenea si coeficientii rezultatelor.

1. **Analiza problemei**

Un polinom este o expresie matematica formata dintr-un sir de unul sau mai multe monoame intre care se efectueaza adunare sau scadere. Un monom este reprezentat prin: semn, coeficient, variabila si exponent. Astfel in acest proiect am implementat polinomul ca fiind un sir de monoame, fiecare monom fiind exprimat prin coeficient ( la care se adauga semnul ) si exponent ( variabila fiind implicit „x” ). Utilizatorul introduce polinoamele in ordine descrescatoare a exponentilor iar monoamele rezultatelor operatiilor sunt, de asemenea, ordonate automat de calculator in ordine descrescatoare. Fiecare operatie este reprezentata in interfata grafica printr-un buton iar rezultatele sunt scrise in casete text. De asemenea interfata dispune si de un buton de resetare a casetelor text, adica resetarea primului polinom, celui de-al doilea, al rezultatului si, in caz de impartire, al restului. Intern, aplicatia are metode prin care: polinomul introdus este impartit in monoame, se sorteaza, se adauga sau elimina monoame din polinom, se ofera informatii legate de proiect ( programator ), trimiterea rezultatului unei operatii ca operand pentru o operatie ulterioara.

1. **Implementare**
   1. **Diagrama UML**

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. Pentru reprezentarea acesteia am folosit generatorul implicit de diagrame UML din Intellij.



* 1. **Clase**

Clasa „Monom” este elementul structural de baza in alcatuirea polinoamelor aplicatiei. Aceasta prezinta trei variabile instanta: „coeficient” si „exponent” de tip intreg si „parcurs” de tip boolean. Primele doua sunt sugestive si reprezinta exact coeficientul si exponentul monomului iar cea de-a treia reprezinta starea de „parcurs” care ne ajuta in momentul operatiilor de adunare si scadere pentru a trece prin ele o singura data, fiind initializat cu valoarea „false”. Aceasta clasa implementeaza interfata „Comparable” pentru a ordona ulterior monoamele in polinom.

Clasa „Polinom” este cea care se ocupa de crearea polinoamelor si a aranjarii acestora ca structura. Prezinta o variabila instanta reprezentata de o lista de monoame („List<Monom>”) „polinom” pentru a putea lucra mai usor pe structura.

Clasa „Operatii” este cea care se ocupa de cele sase operatii (metode) pe polinoame: adunare, scadere, inmultire, impartire, integrare si derivare. Aceasta nu prezinta nicio variabila instanta.

Clasa „View” reprezinta clasa care construieste interfata grafica dintre utilizator si aplicatie si extinde clasa „JFrame”. Aceasta contine mai multe variabile instanta: o variabila de tipul „Operatie”, una de tipul „JPanel” care prezinta continutul interfetei, patru variabile de tip „JLabel”, patru de tip „JTextField” si noua variabile de tip „JButton”. Fiecare are o denumire sugestiva („continut”, „etRezultat”-eticheta, „polUnuText”, „info”) si utilizare unica.

Clasa „Controller” este cea care coordoneaza fiecare legatura dintre interfata si lucru intern al aplicatiei. Prezinta doua variabile instanta: una de tip „Operatii” si una de tip „View” care sa faca legatura dintre sub-clasele ce implementeaza „ActionListener” si metodele din „Operatii” si „View”. Aceasta clasa prezinta actiunile care urmeaza sa fie facute la fiecare apasare de buton al interfetei.

Clasa „Main” reprezinta clasa care porneste aplicatia si deschide interfata grafica.

* 1. **Metode si implementare**

Metodele clasei „Monom”:

Metodele de „get” si „set” sunt folosite pentru a returna sau a modifica valorile variabilelor instanta ale clasei:

Public void setExponent( int exponent ){ Public int getExponent() {

this.exponent=exponent; return this.exponent;

} }

Public void setCoeficient( int coeficient ) { Public int getCoeficient(){

this.coeficient=coeficient; return this.coeficient;

} }

Public void setParcurs ( boolean parcurs ) { Public boolean getParcurs(){

this.parcurs=parcurs; return this.parcurs;

} }

In aceasta clasa mai exista inca doua metode: cea de „compareTo( Monom o )” care compara exponentii a doua monoame pentru a le ordona descrescator in cadrul polinomului, si cea de „toString (Monom o)” care returneaza sub forma de String un monom. Exista mai multe cazuri de scriere sub forma de String a unui monom, in functie de semn, de coeficient si de exponent. De exemplu: in cazul in care coeficientul monomului este 0, nu are rost sa scriem in polinom „+0x^2” ci doar „”, un alt caz este cel care exponentul monomului este 1 si nu are rost sa scriem monomul ca „2x^1” ci doar „2x”. In total sunt 13 cazuri de luat in considerare pentru o scriere cat mai „prietenoasa” si usor de inteles a monomului.

Metodele clasei „Polinom”:

O prima metoda a clasei „Polinom” este „getter-ul” care returneaza lista de monoame, adica componenta „polinom” a obiectului:

Public List<Monom> getPolinom (){

return this.polinom;

}

Sunt existente si doua metode specifice: adaugarea si stergerea unui monom dintr-un polinom:

Public void addMonom (Monom monom){ Public void removeMonom (Monom monom) {

this.polinom.add(monom); this.polinom.remove(monom);

} }

-care adauga la finalul listei un monom -care sterge din polinom un anume monom

O metoda importanta este si cea „toString ()” care ordoneaza descrescator monoamele polinomului obiectului asupra caruia este apelata metoda iar apoi se creeaza un String gol in care se concateneaza fiecare monom ordonat descrescator cu metoda „toString” din clasa „Monom”. Pentru corectitudine, in cazul in care monomul de exponent maxim al polinomul este pozitiv, in String-ul final va fi ignorat semnul „+”:

„+3x^3+2x^2-5” -> „3x^3+2x^2-5”

Probabil una dintre cele mai importante metode ale acestei clase este metoda „verificarePattern (String input)”. Aici este folosit un Regex pentru a separa fiecare monom dintr-un String in 4 „grupuri” astfel: semn, coeficient, variabila si exponent. De exemplu pentru String-ul „x^2-8” avem doua iteratii (monoame) in care sunt gasite grupuri: primul este < „+” „1” „x” si „^2” > iar al doilea este < „-” „8” „” si „” >. Exista si aici anumite cazuri speciale cum ar fi: daca grupul 4 (exponent) este „” atunci exponentul monomului este setat cu 0, sau daca grupul 2 (coeficient) este „” atunci coeficientul monomului este setat cu 0. In cele din urma monomului i se seteaza coeficientul si exponentul, iar apoi este adaugat la lista „polinom”.

Metodele clasei „Operatii”:

Metoda „public Polinom adunare” primeste ca parametrii doua polinoame ( „Polinom polinom1” si „Polinom polinom2” ) si efectueaza operatia de adunare asupra acestora astfel:

Este creeat un obiect nou de tip „Polinom” ( „rezultat” ). Se parcurge fiecare monom al „polinom1” intr-un for each, apoi fiecare monom al „polinom2” tot intr-un for each, se verifica daca valorile exponentilor sunt egale si daca nu sunt deja parcursi (componentele „parcurs” ale monoamelor sunt „false”) atunci unui „monRezultat” ii atribuim valoarea exponentilor monoamelor adunate pentru „exponent” si adunam cei doi coeficienti in „coeficient”-ul monomul rezultat. In cele din urma adaugam monomul rezultat la polinomul rezultat. Un ultim pas este sa adaugam monoamele care au ramas neparcurse din cele doua polinoame la rezultat, dupa care „rezultat” este returnat.

Metoda „public Polinom scadere” primeste ca parametrii doua polinoame ( „Polinom polinom1” si „Polinom polinom2” ) si efectueaza operatia de scadere asupra acestora astfel:

Este creeat un obiect nou de tip „Polinom” ( „rezultat” ). Se initializeaza fiecare monom din cadrul polinomului „polinom1” cu „false” (un aspect care se va arata util in cadrul operatiei de impartire). Se parcurge fiecare monom al „polinom1” intr-un for each, apoi fiecare monom al „polinom2” tot intr-un for each, se verifica daca valorile exponentilor sunt egale si daca nu sunt deja parcursi (componentele „parcurs” ale monoamelor sunt „false”) atunci unui „monRezultat” ii atribuim valoarea exponentilor monoamelor scazute pentru „exponent” si scadem cei doi coeficienti in „coeficient”-ul monomul rezultat. In cele din urma adaugam monomul rezultat la polinomul rezultat. Un ultim pas este sa adaugam monoamele care au ramas neparcurse din cele doua polinoame la rezultat ( monoamele primului polinom cu semnul „+” iar monoamele celui de-al doilea polinom cu semnul „-”), dupa care „rezultat” este returnat.

Metoda „public Polinom inmultire” primeste ca parametrii doua polinoame ( „Polinom polinom1” si „Polinom polinom2” ) si efectueaza operatia de inmultire asupra acestora astfel:

Este creeat un obiect nou de tip „Polinom” ( „rezultat” ). Se parcurge fiecare monom al primului polinom intr-un for each, pentru al doilea polinom de asemenea si se inmultesc fiecare „coeficient” ai monoamelor polinomului 1 cu fiecare „coeficient” ai monoamelor polinomului 2 si de asemenea se aduna fiecare „exponent” ai monoamelor polinomului 1 cu fiecare „exponent” ai monoamelor polinomului 2. Se adauga apoi monomul rezultat la polinomul „rezultat” iar in final acesta este returnat.

Metoda „public Polinom derivare” primeste un singur parametru, un polinom ( tip „Polinom” ) si efectueaza operatia de derivare asupra acestuia astfel:

Este creeat un obiect nou de tip „Polinom” („rezultat”). Se parcurge intr-un for each fiecare monom al polinomului trimis ca parametru, la fiecare iteratie se creeaza un monom nou ( „Monom monRezultat” ). Se seteaza ca „coeficient” al noului monom produsul dintre coeficientul si exponentul < coeficient \* exponent > monomului din iteratia actuala a polinomului, apoi se seteaza ca „exponent”, < exponentul – 1 > din monomul iteratiei. Monomul rezultat este adaugat la polinomul „rezultat” si in final este returnat.

Metoda „public Polinom integrare” primeste un singur parametru, un polinom ( tip „Polinom” ) si efectueaza operatia de integrare asupra acestuia astfel:

Este creeat un obiect nou de tip „Polinom” („rezultat”). Se parcurge intr-un for each fiecare monom al polinomului trimis ca parametru, la fiecare iteratie se creeaza un monom nou ( „Monom monRezultat” ). Se seteaza exponentul noului monom ca < exponentul + 1 > al monomului iteratiei actuale, apoi se seteaza coeficientul noului polinom ca < coeficient / ( exponent + 1 ) > al monomului iteratiei actuale. Fiind toti coeficienti intregi, in cazul unei impartiri la un impartitor mai mare decat deimpartitul, se atribuie 0 coeficientului si exponentului, anulandu-se. Monomul este adaugat in polinomul „rezultat” iar in cele din urma „rezultatul” este returnat.

Metoda „public Polinom impartire” primeste doi parametrii, un polinom1 ( tip „Polinom” ) si un polinom2 ( tip „Polinom” ) si efectueaza operatia de impartire asupra acestora astfel:

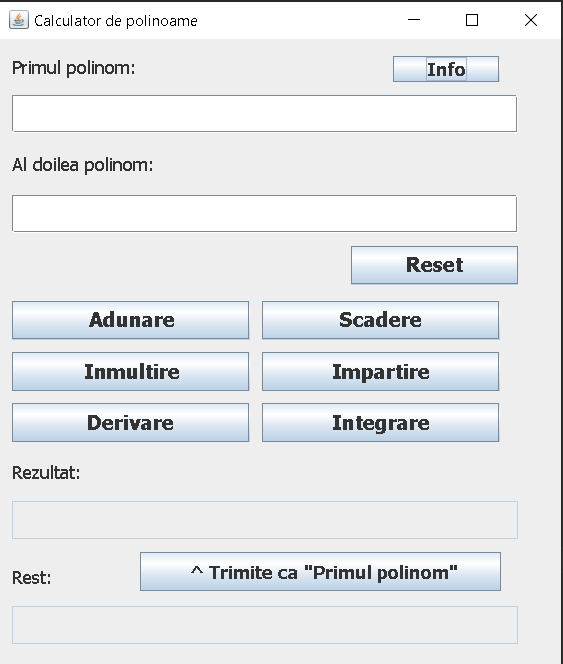
Este creeat un obiect nou de tip „Polinom” („rezultat”). Sunt comparati exponentii maximi ai celor doua polinoame si in cazul in care exponentul impartitorului este mai mare decat cel al deimpartitului, se returneaza un polinom nul ( format doar dintr-un monom cu coeficient si exponent de valoare 0 ). In cazul favorabil, cand impartirea are loc, se creeaza un „Polinom” ( „polinomAux” ) care ne va ajuta sa actualizam deimpartitul dupa fiecare iteratie din impartire. Intr-un while se verifica daca exponentul maxim al lui polinom1 este mai mare sau egal decat exponentul maxim al lui polinom2. In interiorul acestuia este creeat un nou „Monom” ( „monRezultat” ) caruia ii setam exponentul ca fiind diferenta dintre cei doi exponenti maximi iar coeficientul ca fiind catul impartirii dintre coeficientul celui mai mare monom din polinom1 si coeficientul celui mai mare monom din polinom2. Fiind toti coeficienti intregi, in cazul unei impartiri la un coeficient mai mare decat coeficientul deimpartitului , se atribuie 0 coeficientului si exponentului, anulandu-se. Se adauga apoi monomul rezultat la polinomul „rezultat”, se adauga si la polinomul „polinomAux”. Asa cum se procedeaza si in algoritmul original de impartire al polinoamelor, se efectueaza inmultirea ( rezultatul o sa-l numim aici „PARTIAL” intre polinom2 ( impartitor ) si actualul monom rezultat ( „polinomAux” – pentru ca o sa trebuiasca sa fie actualizat la fiecare iteratie cu „monRezultat” si nu putem sa pierdem informatia din polinomul „rezultat” ), apoi din polinom1 (deimpartit) scadem PARTIAL. Curatam „polinomAux” pentru ca in iteratia urmatoare o sa avem nevoie de un alt monom rezultat. Polinomul polinom1 este modificat si se trece la iteratia urmatoarea pana cand exponentul monomului maxim al polinom1 (deimpartit) devin mai mic decat exponentul monomului maxim al polinom2 (impartitor), astfel devenind restul impartirii. Polinomul „rezultat” este la finalul iteratiilor, returnat.

Metoda „public Polinom rest” primeste doi parametrii, un polinom1 ( tip „Polinom” ) si un polinom2 ( tip „Polinom” ) si se obtine restul impartirii astfel:

Intrucat in „impartire” este returnat doar catul, in aceasta metoda se va returna restul. Este creeat un „Polinom rezultat” si un „Polinom aux” asupra caruia voi efectua operatiile de impartire dupa care de inmultire astfel: se imparte polinom1 la polinom2, obtinandu-se catul, apoi se inmulteste catul cu polinom2 iar in ultimul pas se face scaderea intre polinom1 si aux pentru a obtine restul. Metoda aceasta copiaza regula „D : I = C + Rest”.

* 1. **GUI**

Interfata grafica a aplicatiei reprezinta conexiunea dintre utilizator si program. Aceasta afiseaza elemente grafice pentru ca utilizatorul sa foloseasca programul mai usor, fiecare buton fiind sugestiv. Interfata este simpla si usor de manevrat.

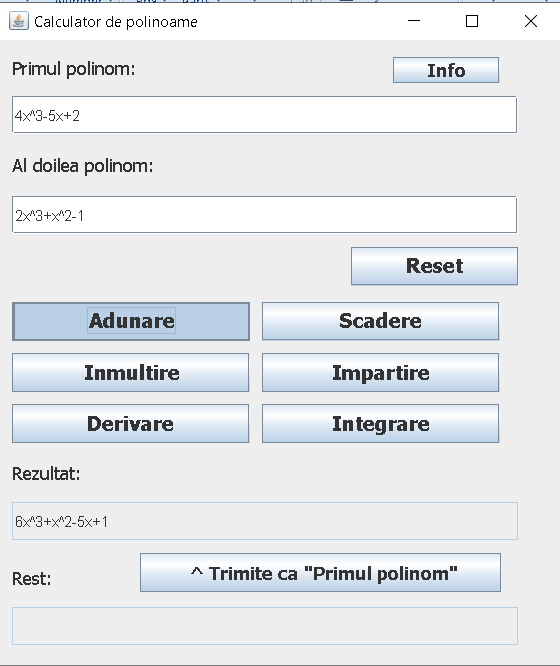


Clasa „View” exinde clasa „JFrame” pentru a crea interfata. Aceasta prezinta un JPanel „continut” la care adaugam restul componentelor interfetei. Contine 2 casete text ( JTextField ), alaturi de doua etichete (JLabel) cu nume sugestive ( in partea de sus a interfetei ), modificabile si in care se pot introduce cele doua polinoame. Interfata dispune de asemenea si de 9 butoane: 6 pentru operatii („Adunare”, „Scadere”, „Inmultire”, „Impartire”, „Derivare” si „Integrare” ), unul pentru „Reset” (curata toate casetele text), unul pentru „Info” ( afiseaza informatii despre programatorul aplicatiei) si unul care sa trimita „Rezultat”-ul ca „Prim polinom”. Importanta celor doua casete text din partea de jos a interfetei este sugerata de etichetele text sugestive: in prima este scris „Rezultat”-ul operatiei iar in a doua este scris „Rest”-ul in cazul operatiei de impartire. Cele doua casete text nu sunt modificabile.

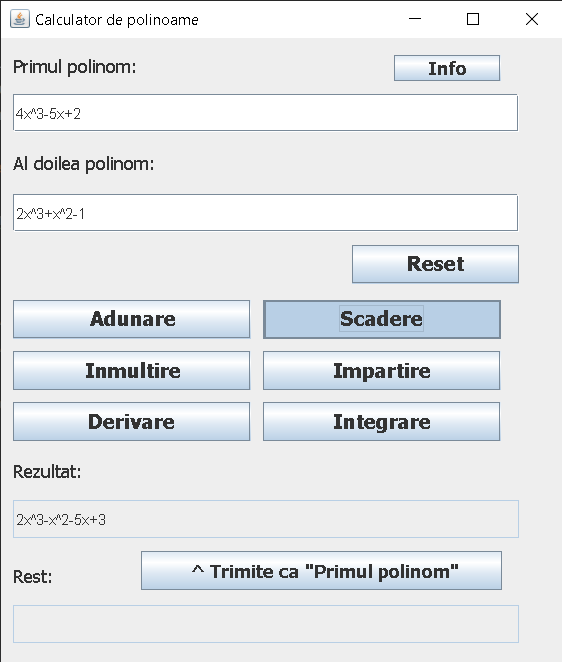
1. **Testare**

In acest capitol vor fi prezentate exemple concrete cu operatii pe polinoame din cadrul aplicatiei prin capturi de ecran din aplicatie:

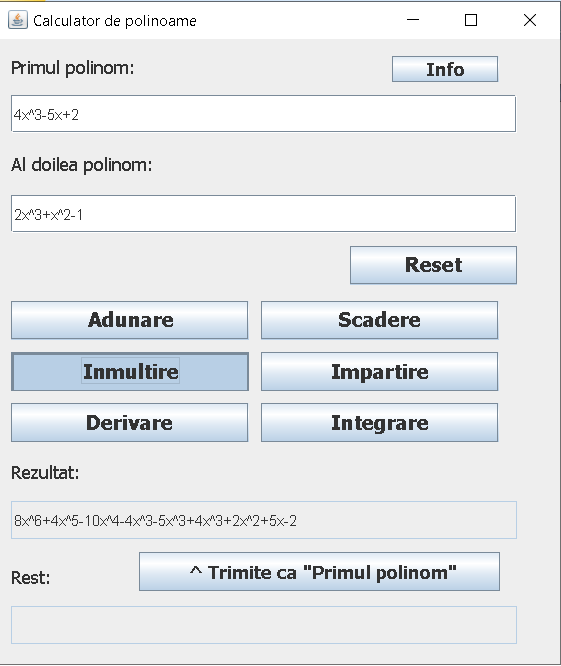
Adunare:



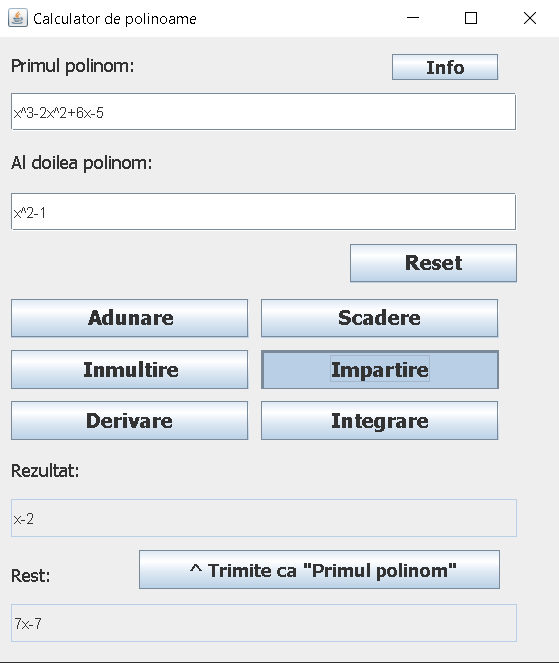
Scadere:



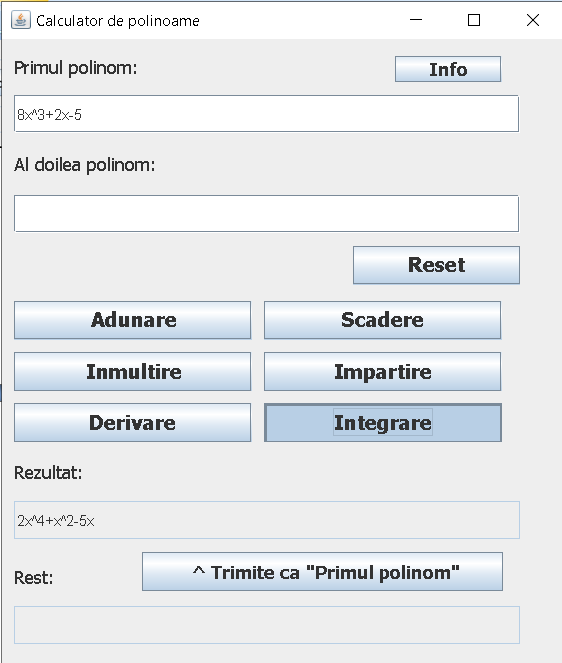
Inmultire:



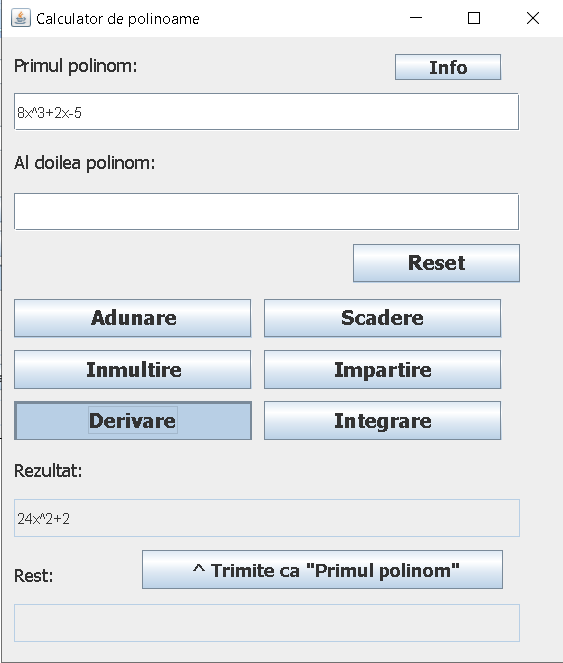
Impartire:



Integrare:



Derivare:



1. **Concluzii**

In concluzie, acest proiect m-a ajutat sa aprofundez notiunile de Programare Orientata pe Obiect, implicit Java, ceea ce asteptam inca din liceu sa invat. Programul poate beneficia ulterior de optimizare privind operatia de inmultire, intrucat monoamele de acelasi exponent nu sunt adunate in final, de asemenea privind si anumite aspecte legate de memorarea monoamelor, deoarece mai exista in anumite cazuri monoame nule ramase in polinoame care nu sunt afisate. Consider ca cele mai grele aspecte ale proiectului au fost: ultilizarea Regex-ului si operatia de impartire.

1. **Bibliografie**

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>

<https://ro.wikipedia.org/wiki/Pagina_principal%C4%83>