

**TEHNICI DE PROGRAMARE**

**Calculator de Polinoame**

**Proiectat și implementat de către:**

**Kasler Mădălina**

**Grupa 30221**

Contents

[1.Cerințe funcționale 3](#_Toc66826217)

[2.Obiectivul temei 3](#_Toc66826218)

[3.Utilizarea aplicației 4](#_Toc66826219)

[4.Analiza Problemei 4](#_Toc66826220)

[5.Proiectare 5](#_Toc66826221)

[6.Implementare 8](#_Toc66826222)

[7.Rezultate 8](#_Toc66826223)

[8.Concluzii 12](#_Toc66826224)

# 1.Cerințe funcționale

Pentru această temă, am avut de gândit, proiectat și implementat un calculator pentru polinoame care sunt alcătuite din monoame care au coeficienți de tip întreg. De asemenea, am avut de proiectat și o interfață grafică pentru ca utilizatorul să poată folosi aplicația mea. Astfel, utilizatorul va introduce două polinoame în cele două câmpuri, urmând să primească răspuns la operația aleasă de ei. Operația poate să fie aleasă dintre următoarele:

* Adunarea a două polinoame
* Scăderea a două polinoame
* Înmulțirea a două plinoame
* Împărțirea a două polinoame
* Derivarea unui polinom
* Integrarea unui polinom

Considerații de implementare:

• Folosirea limbajului de programare Java

• Folosirea Java Swing pentru implementarea interfeței grafice

• Regex – pentru verificarea validității polinoamelor

• Folosirea listelor în locul vectorilor

• Folosirea foreach în locul for(int i = 0…)

• Implementarea claselor trebuie să fie cu maximum 300 de linii de cod(cu excepția claselor UI)

• Implmentarea metodelor trebuie să aibă maximul 30 de linii de cod

• Folosirea JUnit pentru testarea aplicației

# 2.Obiectivul temei

Principalul obiectiv al acestei teme a fost crearea unei aplicații prin care utilizatorul va putea opera anumite operatii pe 1 sau 2 polinoame introduse de la tastatură, doar prin intermediul unei intefețe grafice. Fiecare polinom va fi alcătuit din mai multe monoane, adica un termen care va avea un coeficient și un grad. Atat gradul, cât și coeficientul fiecărui monom din polinom va trebui să fie un număr pozitiv și intreg.

Alte obiective ale temei au reprezentat pașii mici parcurși pentru a ajunge la rezultatul final pe care îl prezint în această documentație.

1. Folosirea unui model de tip MVC (Model-View-Controller) pentru ca toate clasele să fie organizate ierarhic și corect, pentru ca mai apoi să pot proiecta o GUI prin care să se desfășure interacțiunea utilizatorului cu aplicatia.
2. Gândirea și dezvoltarea unol algoritmi simpli și eficienți pentru a efectua operațiile prezentate mai sus.
3. Proiectarea și implementarea soluției finale de prezentat prin legarea claseslor care se ocupa de operațiile pe polinoame, care realizează interfața grafica și restul.
4. Testarea unitara se vor crea mai multe scenarii a operațiilor pe polinoame, prin care se va verifica faptul că algoritmul din spatele operației funcționează corespunzător.

# 3.Utilizarea aplicației

Utilizarea aplicației este foarte simplă și poate să fie făcută de către orice utilizator. Inițial, utilizatorul trebuie să introducă de la tastatură în cele două câmpuri text (sub cele doua label-uri care indică Primul Polinom și Al Doilea Polinom) cele 2 polinoame cu care va dori să facă operațiile. Cele doua polinoame vor trebui să fie sub forma coeficientx^grad+/-coeficientx^grad+/-…... După ce a introdus cele 2 polinoame, următorul pas pe care va trebui să îl faca va fi acela de a-și alege operația pe care dorește să o facă, folosind butoanele din partea stânga care conțin numele fiecărei operații: adunare, scădere, împărțire, înmulțire, derivare sau integrare. Dacă utilizatorul va dori să selecteze operația de integrare sau de derivare, operații care se vor aplica doar asupra unui singur polinom, acesta ar trebui să știe că operația se va aplica doar asupra primului polinom. După apăsarea butonului corespunzator operației, aplicația va lua în considerare doar termenii introduși conform formatului menționat mai sus, ignorând textul introdus de utilizator ce nu corespunde cu formatul specificat. În cazul în care în unul din câmpurile text nu este identificat niciun termen introdus corect sau nu există nimic introdus se va deschide o nouă fereastră care va atenționa utilizatorul ca datele introduse nu respectă formatul și operația selectată nu se poate executa. În cazul în care datele sunt introduse corect, programul va genera rezultatul dorit și îl va afișa în ultimul chenar de text(sub label-ul „ Rezultat: ”). Polinoamele introduse inițial rămân în câmpurile text, astfel fiind posibilă efectuarea unei alte operații pe aceleași două polinoame.

# 4.Analiza Problemei

Utilizarea interfeței presupune introducerea de catre utilizator în două JTextField-uri polinoamele care urmează să fie supuse efectuării unor operații(adunare, scădere, înmulțire, împarțire, derivare sau integrare). Alegerea operației ce urmează a fi efectuată asupra celor două polinoame se poate alege prin intermediul celor 7 JButton,iar rezultatul va fi afișat într-un al treilea JTextField, iar restul cazul operației de împărțire se va afișa în ultimul JTextField.

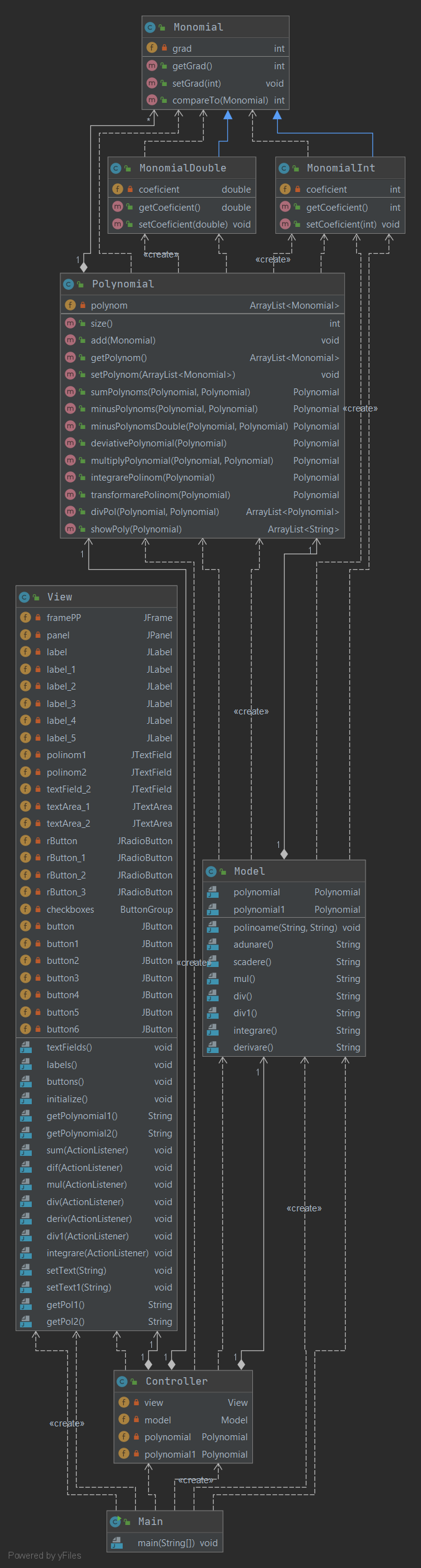
Deoarece este posibil să se introducă date greșite(String-uri care nu reprezintă un polinom), se va afișa un mesaj pe ecran care va avertiza utilizatorul că datele introduce de el sunt invalide(nu respectă formatul). Dacă polinomul introdus este dezordonat(gradele variabilei nu sunt în ordine descrescătoare, acesta va fi considerat corespunzător, deoarece polinomul va fi aranjat ulterior de către program, înaintea afișării rezultatului așteptat). În cazul derivării sau al integrării, rezultatul afișat va fi cel pentru primul polinom introdus de către utilizator.

# 5.Proiectare

Pentru a putea implementa și proiecta produsul final, am ales să folosesc 2 clase principale Polynomial si Monomial. Clasa Monomial va fi moștenită de alte 2 clase: MonomialInt și MonomialDouble. Folosesc aceste 2 clase pentru că, coeficientul unui Monom poate sa fie Double sau Int. Clasa Monomial va mai avea atributul grad, care va fi integer, iar MonomialInt va avea un coeficient de tipul integer, iar MonomialDouble va avea unul de tip double. Clasa principala Polynomial va conține un arraylist de monoame (adică de obiecte din clasa Monomial), iar aceasta va fi structura de date principală folosită, deoarece mai multe monoame formează un polinom. Pe lângă acestea voi mai avea cele 3 clase Model-View-Controller care alcătuiesc intefața grafică.

Diagrama de clase

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele și specificații software. Diagrama de clase UML este folosită pentru reprezentarea vizuală a claselor și a interdependențelor, taxionomiei și a relațiilor de multiplicitate dintre ele. Diagramele de clasă sunt folosite și pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte și a legăturilor concrete dintre acestea.



Algoritmi

Adunare polinoame

Algoritmul de adunare polinoame presupune crearea unui nou Monom care va avea ca și coeficient suma dintre coeficienții monoamelor din cele 2 polinoame unde gradul este coumn. Adunarea a două polinoame este implementată prin parcurgerea primului polinom și parcurgerea celui de-al doilea polinom, pentru fiecare monom din primul. În acest fel, în momentul în care pentru un monom din primul polinom se găsește un monom în cel de-al doilea polinom cu același grad ca și el, coeficienții celor două monoame se vor aduna și vor duce la crearea unui nou monom, care va fi adăugat în polinomul care reprezintă rezultatul.

Scădere polinoame

Algoritmul de adunare polinoame presupune crearea unui nou Monom care va avea ca și coeficient diferența dintre coeficienții monoamelor din cele 2 polinoame unde gradul este coumn. De asemenea, acest algoritm este aproape identic cu cel de adunare, atâta doar că se vor scădea coeficienții și termenii care nu au fost modificați în urma scăderii vor fi inmulțiti cu (-1).

Înmulțire polinoame

Algoritmul de înmulțire de polinoame presupune înmulțirea fiecărui monom din primul polinom cu fiecare monom din al doilea polinom și apoi se va face suma unde gradul polinomului rezultat este egal. Aici se va face parcurgera simultană a celor 2 polinoame, iar mai apoi se va parcurge polinomul rezultat, pentru a se regrupa termenii. Cele două monoame din cele două polinoame vor fi șterse.

Împărțire polinoame

Algoritmul de împărțire de polinoame presupune împărțirea a fiecărui monom din primul polinom, cu fiecare monom din al doilea, iar apoi se va face diferența dintre ele. Algoritmul merge până in momentul în care gradul împărțitorului devine mai mic ca și cel al deîmpărțitului. In acel moment, tot ce rămâne în împărțitor va fi considerat rest.

Derivare polinom

Derivarea monomului presupune ca primul coeficient să fie produsul dintre coeficientul monomului ce trebuie a fi derivat și gradul acestuia, noul monom va avea gradul egal cu gradul curent minus 1. Algoritmul de derivare presupune o simplă parcurgere a polinomului ce trebuie derivat în care fiecărui termen i se aplică metoda de derivare concepută în clasa monom.

Integrare polinom

La integrarea monomului coeficientul nou va fi coeficentul vechi împărțit la gradul actual minus 1, iar coeficientul nou este cel actual incrementat cu valoarea 1. Aici se va aplica același procedeu ca și la derivarea polinomului.

# 6.Implementare

Clasa Monomial – variabile instanță: un întreg care va fi exponentul monomului. Această clasă va avea ca și metode getteri si setteri, dar și o metodă prin care pot compara monoamele pentru a putea sorta monoamele într-un polinom ca să fie mai ușor. Această clasă extinde clasa Comparable.

Clasa MonomialInt care va extinde clasa Monomial – variabile instanță: un întreg care va fi coeficientul monomului. Această clasă va avea ca și metode getteri si setter pentru a putea face cast pe obiectele de tip MonomialInt pentru a le introduce in ArrayList-ul polynom care contine obiecte de tip Monomial.

Clasa MonomialDouble care va extinde clasa Monomial – variabile instanță: un double care va fi coeficientul monomului. Această clasă va avea ca și metode getteri si setter pentru a putea face cast pe obiectele de tip MonomialDouble pentru a le introduce in ArrayList-ul polynom care contine obiecte de tip Monomial.

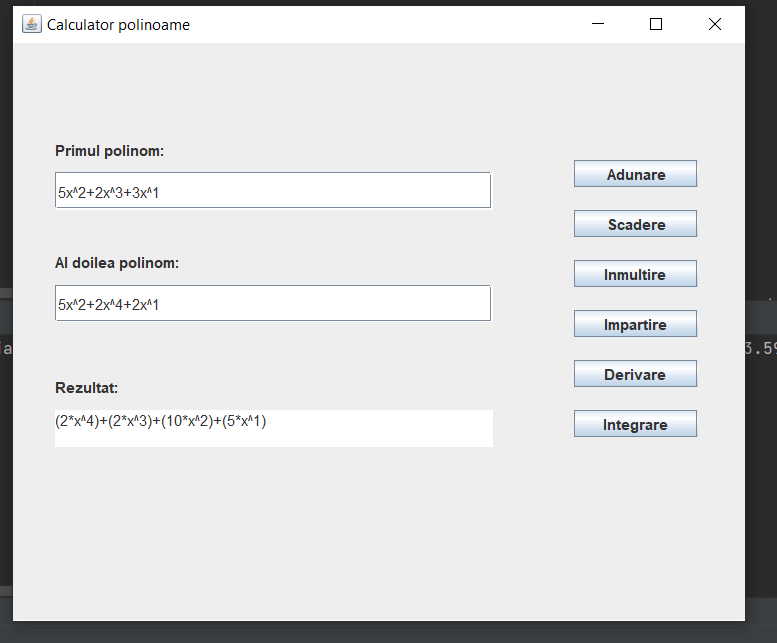
Clasa Polynomial are ca variabile instanția un singur ArrayList de monoame din care o să fie format polinomul. Aceasta clasă are toate operațiile implementate pentru proiect, toate metodele care descriu operațiile, precum și niște metode ajutătoare pentru a se afișa rezultatul.

Clasa View – variabile instanță: frame-ul, un panel,1 JTextArea, 2 JtextField-uri, 3 Jlabel-uri și 6 Jbutton-uri. Cele 3 TextField-uri sunt folosite fie pentru introducerea datelor în interfața grafică de către utilizator, cât și pentru afișarea rezultatului așteptat în urma realzării operației selectate. Butoanele sunt folosite pentru selectarea operației pe care utilizatorul dorește să o faca. După ce utilizatorul va introduce cele 2 polinoame dorite (sau doar unul singur dacă utilizatorul dorește să facă integrare sau derivare) va selecta butonul corespunzător aplicația va afișa in ultimul textArea

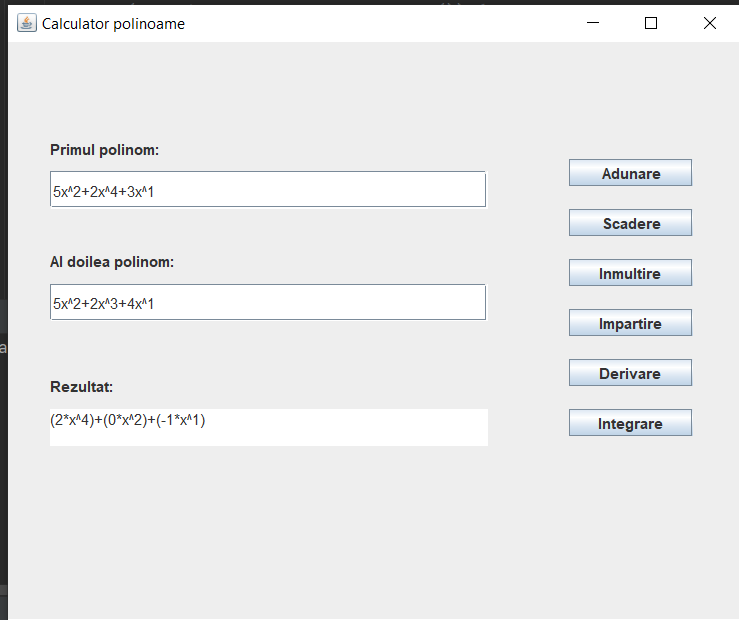
Clasa Controller – variabile instanță: un model și un view si 2 obiecte de tip Polinom; 2 metedde implementate în această clasă: aceea care controlează ceea ce se întamplă cand are loc un eveniment asupra unuia dintre butoane. și metoda care va prelua cele 2 șiruri de polinoame de la tastatura si le va transofrma in 2 obiecte de tip polinom, alcatuite din mai multe monoame, pentru a se putea realiza mai apoi operațiile pe polinoame.

# 7.Rezultate

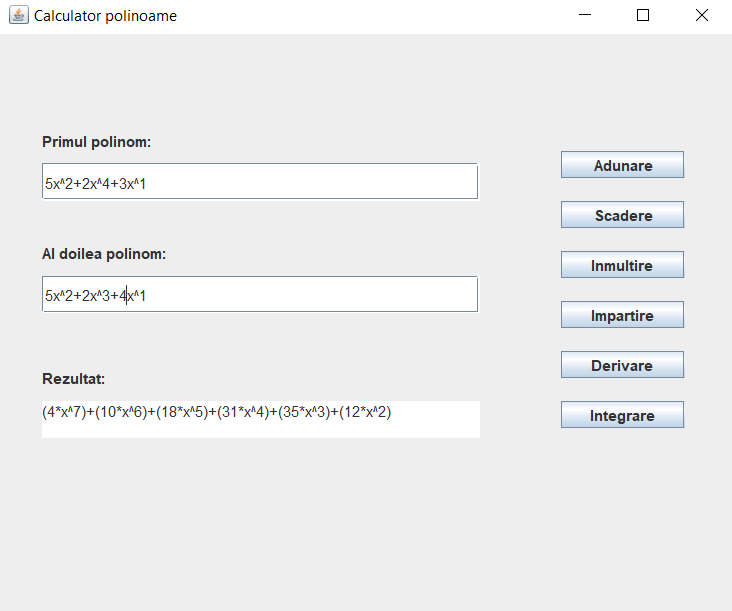
Adunare



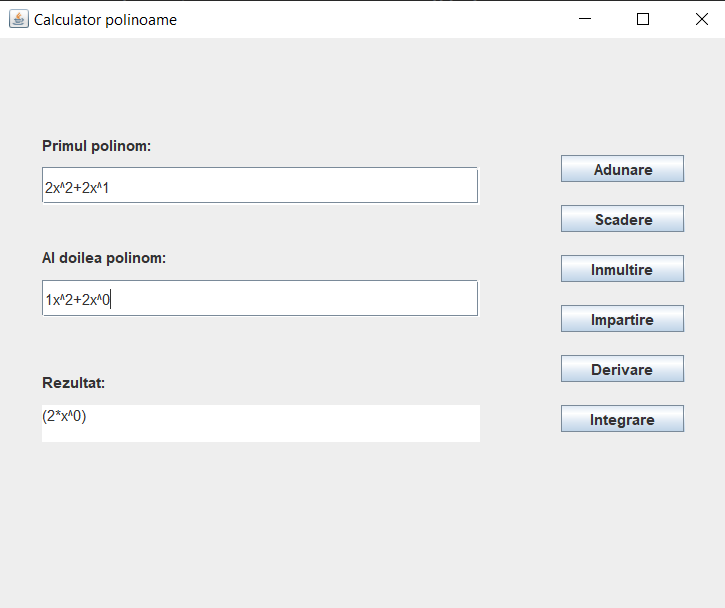
Scădere



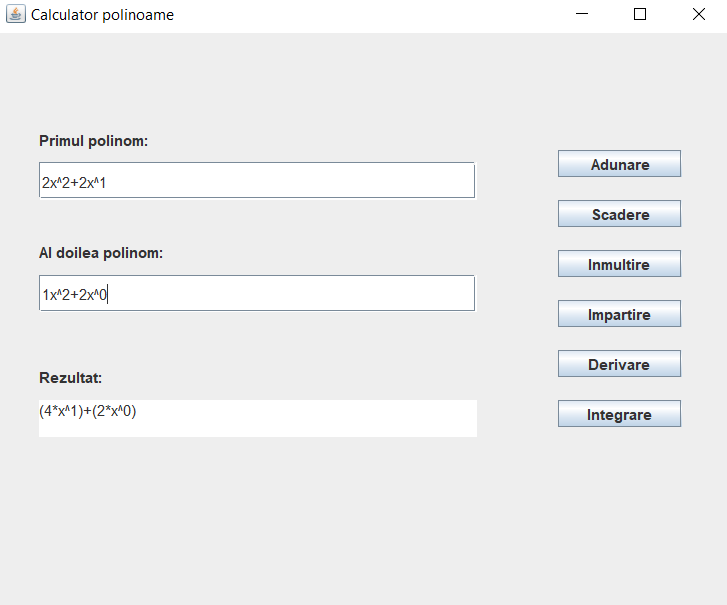
Înmulțire



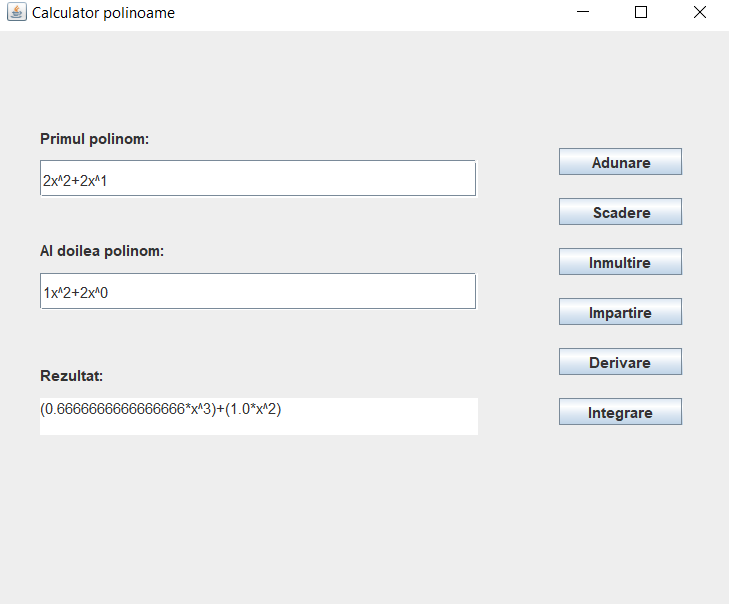
Impărțire



Derivare



Integrare



# 8.Concluzii

După cum am spus la început, pentru a realiza acest proiect am folosit multe concepte învățate la materia Programare Orientată pe Obiect din semestrul anterior și de asemenea am fixat cunoștințe legate de modeul Model View Controller, deci consider că această tema a fost foarte utilă.

Ca și dezvoltări ulterioare aș incerca sa modific modul în care cele 2 polinoame vor fi introduse, pentru a putea exista mai multe modalități în care cele 2 polinoame pot fi introduse, pentru a nu se respecta o structură standard ca și acum. Acum este necesar sa nu am operatorul \* între coeficientul monomului și variabila x, iar operatorul ^ trebuie să existe între x și gradul polinomului pentru a se putea realiza operațiile dorite. Aș incerca să fac să nu mai existe această barieră în introducerea polinomului, pentru că astfel va fi mult mai simplu pentru orice utilizator să introducă cele 2 polinoame. Aș mai adăuga câmpuri care să deschidă un nou panel în care sa pot selecta pe câte polinoame doresc să fac operațiile existente, sau pe care polinom. De exemplu, poate aș dori să fac adunare pe 3 polinoame, și mai apoi să il derivez rezultatul primit. Aș face si un buton care să imi deschidă un meniu de tip HELP, cu exemple pe operații pentru a putea înțelege mai bine cum funcționează algoritmii din spatele programului, astfel acest calculator ar putea fi folosit ca un mecanism de învățare. Astfel fiecare operație ar putea fi individualizată, iar frameurile respective ar putea conține mai multe informații despre modul în care se realizează operația respective pe polinoame, despre modul în care ar trebui introduse datele astfel încât acestea să fie validate de către aplicație, iar rezultatul să fie introdus correct.