Calculator de polinoame

Documentatie

Munteanu Cezar-Lucian

Grupa 30226

Profesor Coordonator: Dorin Vasile Moldovan

Cuprins

1. Cerinte Functionale………………………………………………………………………3
2. Obiective…………………………………………………………………………………3

2.1. Obiectiv Principal……………………………………………………………………3

2.2. Obiective secundare…………………………………………………………………3

3. Analiza problemei………………………………………………………………………….4

4. Proiectare…………………………………………………………………………………..5

4.1. Structuri de date……………………………………………………………………...5

4.2. Diagrame de clasa……………………………………………………………………5

4.3. Algoritmi……………………………………………………………………………..7

5. Implamentare……………………………………………………………………………….8

6. GUI………………………………………………………………………………………..21

7. Testare……………………………………………………………………………………..23

8. Concluzii si posibilitati de dezvoltare……………………………………………………..23

1. Cerinte Functionale

Se cere dezvoltarea unui calculator de polinoame care sa permita urmatoarele operatii matematice: adunare, scadere, inmultire, impartire, derivare si integrare (dintre care avem si unele optionale) . Acesta poate citi de la tastatura un polinom sub forma: 3x^2 + 2x + 1 prin intermediul unei interfete grafice.

1. Obiective
   1. Obiectiv principal:

Obiectivul principal este de a proiecta si implementa un calculator polynomial cu interfata grafica dedicate prin care utilizatorul poate introduce polinoame, selecta operatia matematica care urmeaza sa fie efectuata si totodata vizualiza rezultatul acestei operatii.

* 1. Obiective secundare:

Pentru a putea ajunge la un obiectiv final, problema trebuie sa fie impartita in probleme mai mici, respective obiective mai mici. Asadar avem urmatoarele obiective secundare:

* **Dezvoltarea de use case-uri si scenarii**

Reprezinta scenariile in care programul merge, respective cazurile in care merge si modul in care trebuie utilizat programul pentru ca utilizatorul sa nu aiba un rezultat nedorit.

* **Alegerea structurilor de date**

Pentru a realiza operatiile avem matematice intr-un mod cat mai efficient este nevoie de anumite structure de date. Ca de exemplu pentru a putea tine minte un polinom, acesta va fi declarant ca un ArrayList de monoame.

* **Dezvoltarea algoritmilor**

Pentru a putea realiza operatiile matematice cerute de accest calculator, este necesar sa concepem niste algoritmi transpusi dupa algoritmii matematici clasici.

* **Testare**

Ca sa fim siguri ca calculatorul nostrum merge, trebuie sa ne verificam codul, asa ca , vom verifica rezultatele obtinute in urma operatiilor effectuate pe polinoame.

1. Analiza problemei

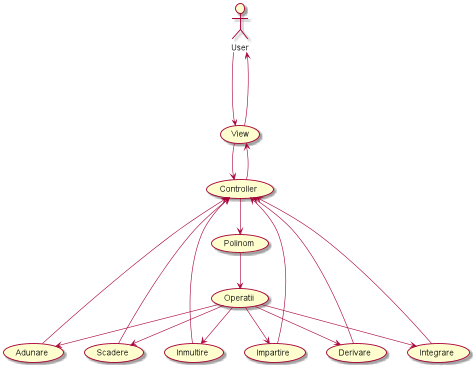


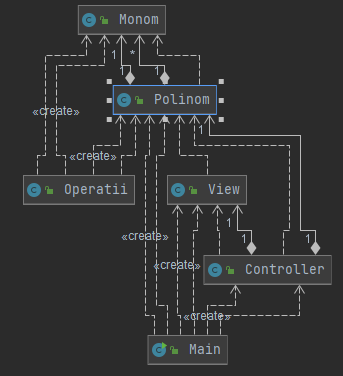
Diagrama USE-CASE este cea mai simpla reprezentare a interactiunii dintre utilizarotor cu sistemul/programul, astfel intelegand procesul de interactiune mult mai usor. In cazul problemei noastre, USE-CASE-ul este unul relative simplu, deoarece avem doar un utilizator care poate interactiona doar cu interfata grafica . Prin urmare, utilizatorul va folosi interfata grafica pentru a introduce cele doua polinoame si va alege butonul cu operatia dorita. Odata ce acesta apasa butonul si va alege ce operatie doreste sa faca, programul va extrage input-ul din interfata si il va prelucra astfel urmand prin intermediul view-ului sa afiseze tot pe interfata rezultatul operatiei cerute.

1. Proiectare
   1. Structuri de date

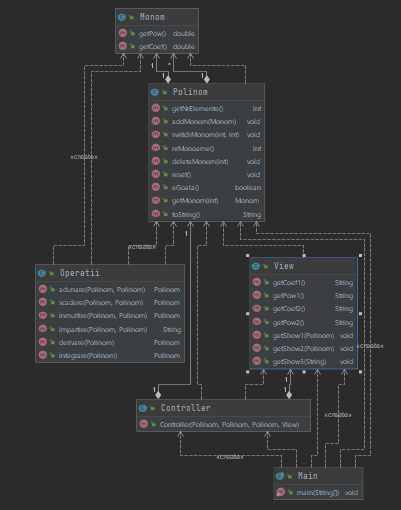
Pentru a putea realiza un model Polinom, am ales sa lucrez cu o structura de date de tip ArrayList. Aceasta va fi folosita pentru a putea inlantui o lista de Monoame, care ulterior va define modelul nostrum Polinom, intr-un mod cat mai efficient din punct de vedere al memoriei, si al timpului de executare. Am ales aceasta structura de date pentru ca ofera un set de instructiuni deja predefinite care ne ajuta la o scriere cat mai usoara a codului cat si ca acesta sa fie mult mai usor de inteles prin instructiunile sale predefinite extrem de sugestive dupa nume cum ar fi: add() – adauga element, List.clear() – sterge elementele din lista, List.isEmpty() – verifica daca lista e goala etc..

* 1. Diagrama de clase

Diagrama de clase (UML) este o structura de diagrama statica care descrie structura unui system afisand clasele, atributele si metodele implementate cat si relatia dintre obiectele proiectului. Mai jos avem reprezentata o diagrama UML simplificata pentru a urmarii mai usor relatiile dintre obiecte . De exemplu putem vedea ca intre clasa Monom si Polinom exista o relatie de compozitie, care reprezinta faptul ca un polinom nu poate exist afara un monom, si ca este alcatuit din cel putin un monom.



Aici avem si o reprezentare mai detaliata a diagramei UML:

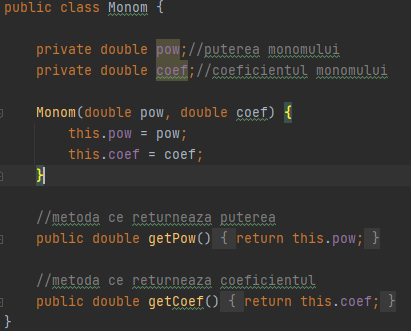


* 1. Algoritmi

Pentru a rezolva cerintele nu a fost nevoie de algoritmi foarte complicati cum ar fi BFS sau DFS, ci a tebuit sa eleborez cate un algoritm pentru fiecare operatie matematica. Acestia au fost realizati urmarind un algoritm matematic classic de rezolvare care a fost verificat pe foaie. Un exemplu il reprezinta algoritmul de impartire a doua polinoame. Practic, pentru a vedea de cate ori impartitorul cuprinde deimpartitul , al scazut exponentii si am impartit coeficientii, pentru inmultirea deimpartitului si catului am declarant un monom care e rezultatul impartirii anterioare pe care l-am inmultit cu polinomul deimpartit. La fel am procedat pentru fiecare algoritm dezvoltat.

1. Implementare

Clasa Monom:



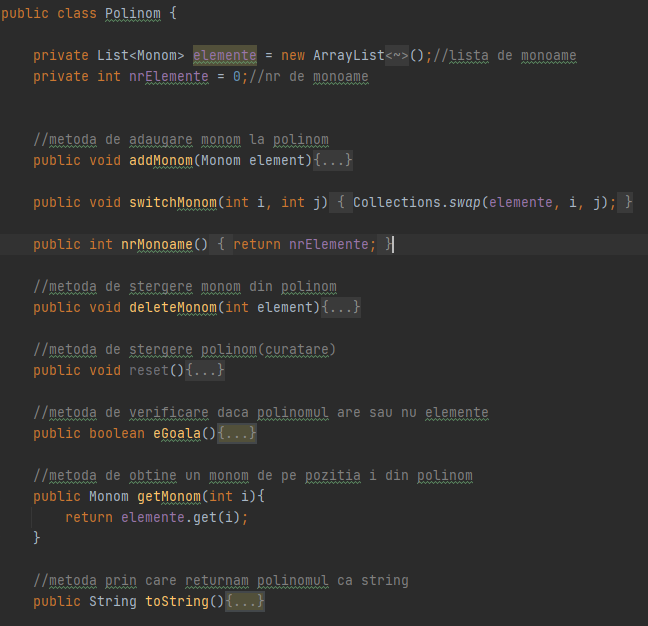
In aceasta clasa este reprezentat modelul unui Monom. Un monom reprezinta un polinom care contine doar un termen de forma a0x^a1. Fiecare instanta va avea ca variabila de clasa un exponent si un coefficient. Un monom va fi instantiate direct din constructor. Aceasta clasa sta la baza clasei Polinom, fiecare polinom fiind compus din mai multi Monomi.

In aceasta clasa avem delcarate 2 metode:

getPow() – aceasta metoda returneaza exponentul monomului declarat ;

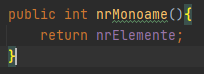
getCoef() – aceasta metoda returneaza coeficientul monomului declarant ;

Clasa Polinom:

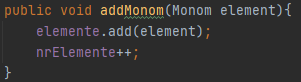


Clasa polinom contine modelul polinom. In aceasta clasa se implementeaza cateva functii ajutatoare pentru operatiile ce o sa le avem de facut in clasa Operatii. Ca variabila de clasa,

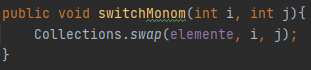
Polinom contine un ArrayList de monoame. Functiile acestei clase sunt:



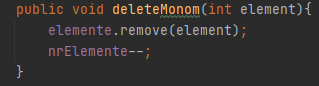
Aceasta functie este una simpla si ne returneaza numarul de elemente ale Polinomului, mai exact numarul de monoame.



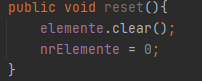
Aceasta functie este la fel de simpla ca cea anterioara si are ca scop adaugarea in polinom a unui element nou (a unui monom nou), totodata crescand numarul de elemente.



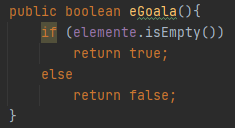
Aceasta functie are rol de interschimbare a doua elemente din Polinom. Pentru a putea realiza acesta functie am folosit libraria:  .



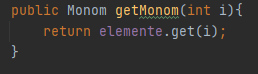
Aceasta functie este o functie de stergere din Polinom. Ea lucreaza pe indici, dandu-se in antet pozitia elementului ce este dorit a fi sters. Pentru a sterge elemetul cerut folosim o functie predefinita pentru ArrayList, mai exact functia remove() , dupa care scadem cu 1 numarul de monoame din polinom.



Functia de reset() este o functie ce are ca rol curatarea Polinomului, transformandu-l intr-un polinom null. Aceasta foloseste la baza functia predefinita de ArrayList, clear(), si totodata facem ca numarul de elemente din polinom sa devina 0 .

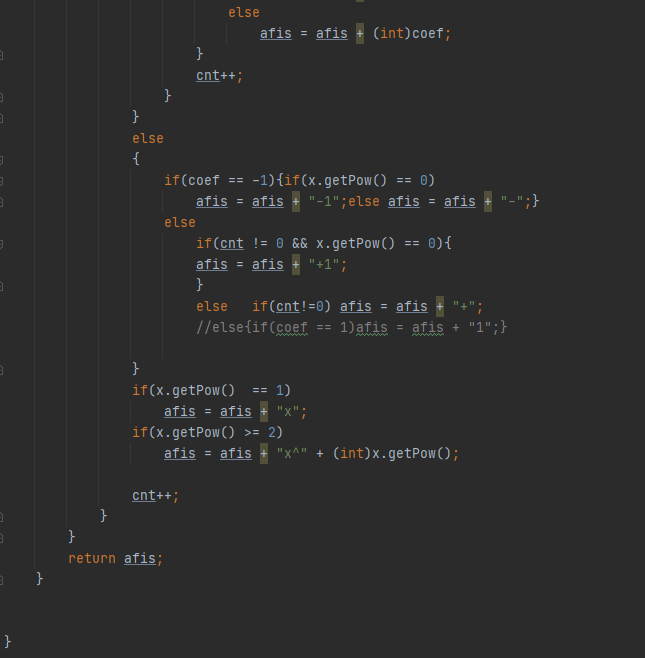


Functia eGoala() va returna o constanta de tip boolean, care ne va spune daca Polinomul nostru este null sau nu. Pentru a realiza aceasta functie am folosit functia predefina din ArrayList, isEmpty().



Functia getMonom() va returna o variabila de tip Monom. Aceasta are ca rol returnarea monomului de pe pozitia i data in antet.

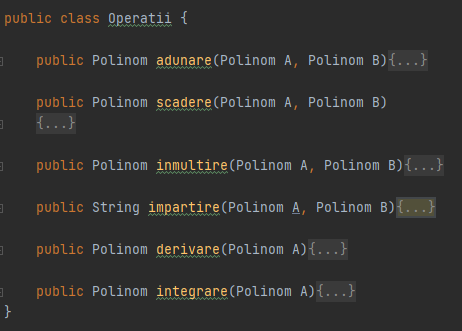




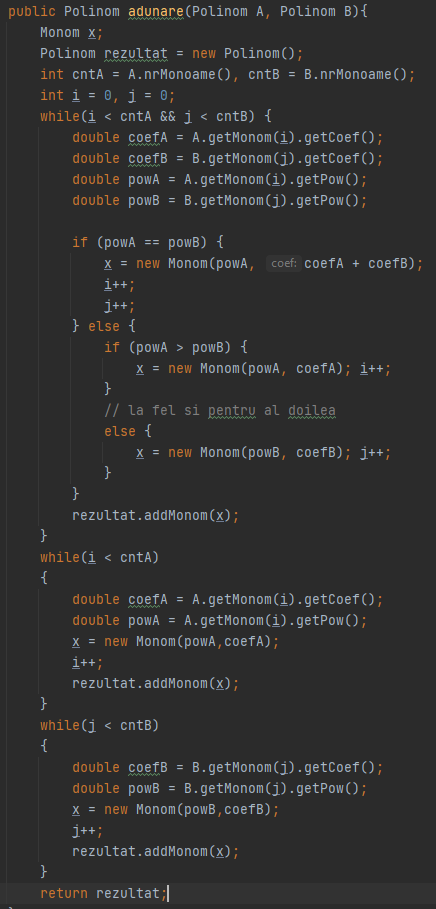
Aceasta functie este una putin mai complicate, si are rolul de a returna un string, mai exact polinomul nostru sub forma de String. Pentru a realiza aceasta am avut nevoie de un String rezultat ( “afis” ) si un counter (cnt) care ne va ajuta la determinarea cazului cand parcurgerea noastra se va afla la primul element . Initial verificam daca Polinomul nostrum este null, daca acesta nu este atunci trecem la parcurgerea sa cu ajutorul unui foreach. Pentru fiecare monom din polinom verificam mai multe conditii cum ar fi daca suntem la primul element (cnt = 0), pentru a determina daca punem sau nu in fata sa semnul necesar, mai verificam daca acesta are coeficent null cee ace inseamna cu nu va mai fi nevoie sa fie afisat, mai verificam daca coeficientul este 1 sau -1 ceea ce inseamna ca daca vrem sa afisam un monom cu astfel de coeficienti el va afisa doar “x”-ul si semnul lui corespunzator, iar daca elementele sunt diferite de ceea ce am precizat mai sus, atunci inseamna ca ele nu sunt cuprinse in acele cazuri deci vor fi adaugate la Stringul rezultat corespunzator.

Pentru coeficientii de tip double am folosit o functie predefinita pentru a arata doar 2 zecimale, mai exact, functia String.format().

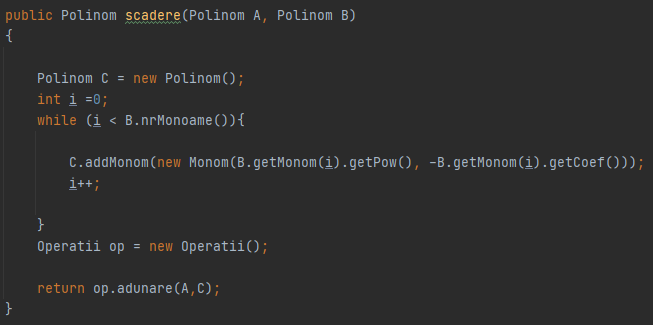
Clasa Operatii:



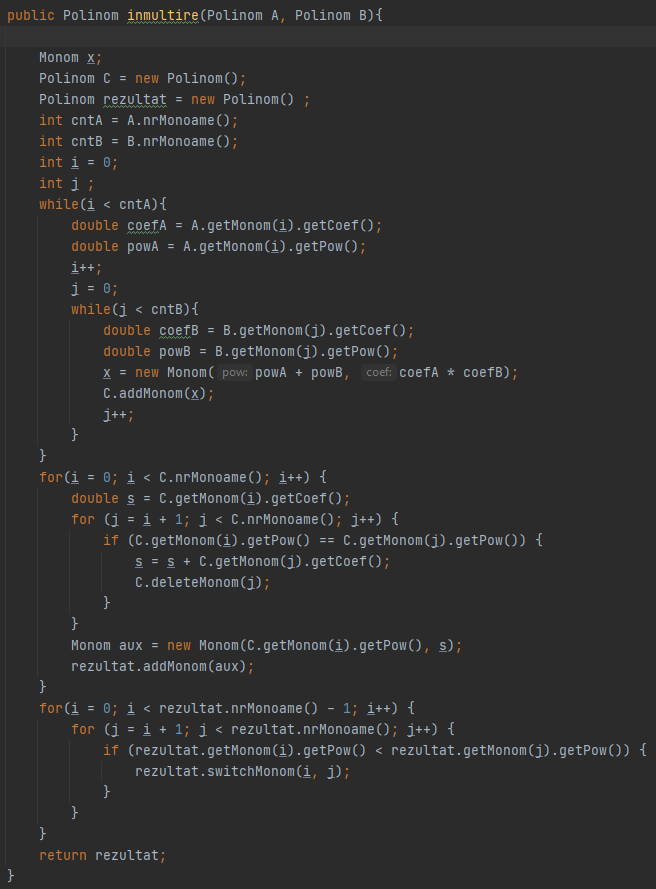
Aceasta este cea mai importanta clasa, mai exact cea in care sunt scrise toate operatiile matematice care le va face calculatorul nostru. Aceasta clasa are toate functiile cerute incepand de la adunare pana la integrare. Pentru inceput vom incepe cu:



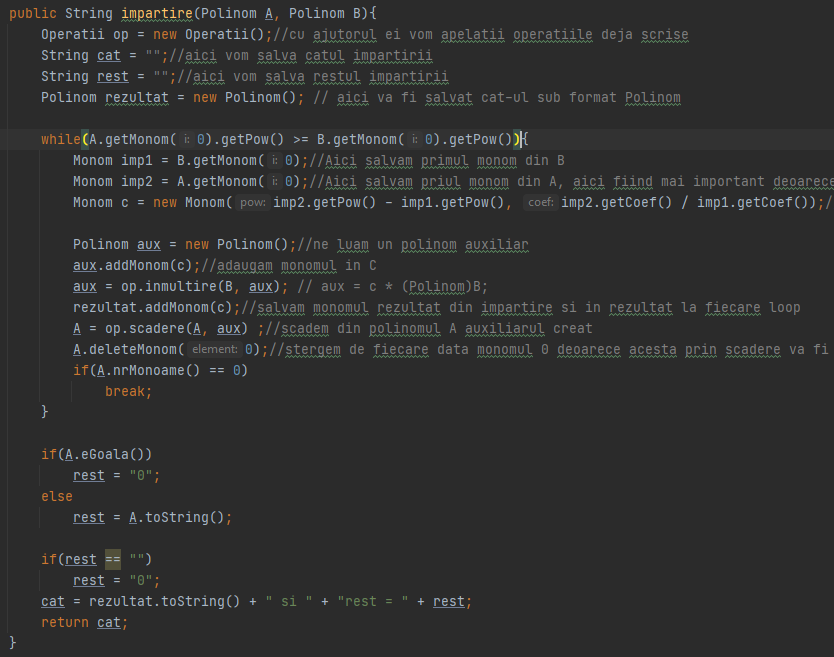
Functia de adunare, care returneaza un Polinom “rezultat”. Aceasta se bazeaza pe algoritmul matematic classic, ea verificand fiecare monom si adunand monomii ce au acelasi exponent. Pentru a realiza aceasta am avut nevoie de un polinom “rezultat” in care sa salvam suma celor 2 polinoame date in antet, pentru a parcurge cele doua polinoamne date si construirea celui care trebuie returnat, ne-am folosit de trei loop-uri de tip while primul adaugand suma monoamelor cu coeficienti egali din cele 2 polinoame, in timp ce al doilea si al treilea while adaugand in polinomul rezultat, monoamele ce nu s-au gasit in celalalt polinom .



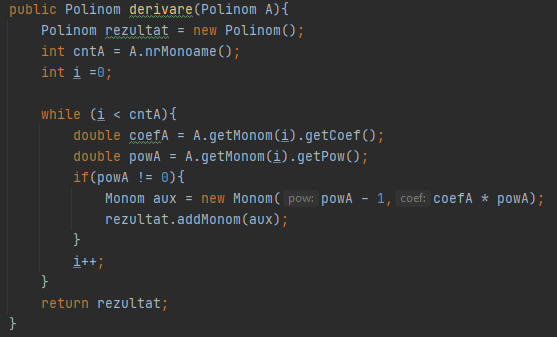
Functia de scadere va returna tot un polinom rezultat. Aceasta se bazeaza pe algoritmul precedent, mai exact pe cel de adunare. Pentru realizarea acestui aloritm ne-am folosit de formula A – B = A + ( - B ) . Acest lucru ne-a ajutat sa economisim foarte multe linii de cod sit imp, noi avand deja functia de adunare. Cu ajutorul unui loop de tip while am creat un nou polinom C care va fi = -B, si care va fi adunat la primul polinom, astfel rezultand diferenta dintre cele 2 polinoame date.



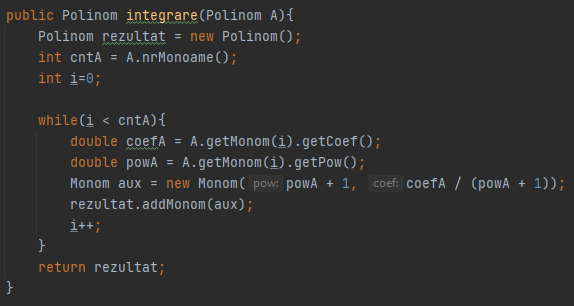
Algoritmul de inmultire. Acesta va returna tot un Polinom rezultat. El se bazeaza pe algoritmul matematic classic, mai exact cel prin care inmultim fiecare monom din primul Polinom cu monoamele din al doilea Polinom. Inmultirea se efectueaza prin adunarea exponentilor si imultirea coeficientilor. Pentru a realiza inmultirea de fiecare cu fiecare vom folosi un loop in loop, mai exact un while in while, care ne permite sa inmultim moanoamele intre ele. Pentru ca Polinomul nostrum rezultat dupa acel while in while poate contine termini ce au acelasi exponent, am ales sa ma folosesc de un loop de tip for pentru a putea sa le adun intre ele, iar la final am mai folosit tot un for pentru a putea ordona monoamele crescator dupa operatiile effectuate.



Algoritmul de impartire. Acesta va returna un String si este bazat pe teoria matematica clasica D(deimpartit) / I(impartitor) = cat + rest . Pentru a adapta acest algoritm matematic in cod, m-am folosit de un loop while care atat timp cat exponentul primului monom din polinom A este mai mare decat exponentul primului monom din exponentul B, vom putea efectua algoritmul. Mai exact, ne vom lua un monom auxiliar in care salvam catul impartirii pentru a-l inmulti mai departe cu deimpartitul si a efectua algoritmul classic de impartire al polinoamelor . Deoarece primul monom din polinomul A va deveni null dupa fiecare pas din loop, acesta va fi sters pentru ca loop-ul sa continue si sa aplice algoritmul in continuare. Rezultatul va fi returnat ca String si va fi compus din cat si rest.



Functia de derivare. Este o functie simpla, se bazeaza pe algoritmul matematic classic. Pentru aceasta functie derivam prin intermediul unui while fiecare monom din polinom, mai exact, inmultim coeficientul actual cu valoarea exponentului si exponentul il scadem cu 1. Functia va returna un Polinom.



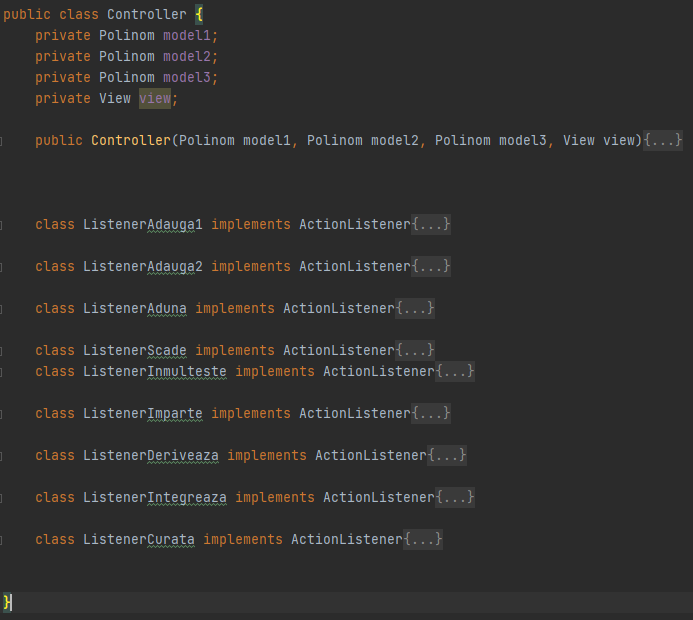
Functia de intergrare. La fel de simpla ca cea de derivare. Se bazeaza si ea la randul ei pe algoritmul matematic classic. Tot cu ajutorului unui while parcurgem polinomul monom cu monom si il integram prin impartirea coeficientului cu valoarea exponentului – 1 si incrementarea exponentului.

Clasa View:



Aceasta clasa realizeaza interfata grafica a programului, fara a executa nicio operatie. Aici sunt declarate toate butoanele, text field-urile si text area-urile . Totodata aceasta contine si Linstener-ele pentru butoane, cat si functiile pentru elementele declarate.

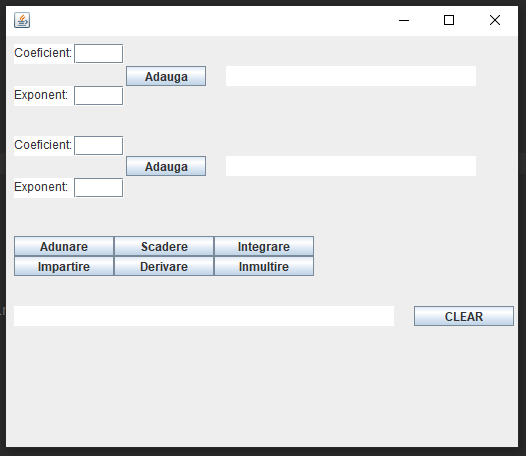
Clasa Controller:



Aceasta functie realizeaza controlul interfetei, evident componenta controller din MVC. Aceasta aduna informatiile date de catre utilizator cat si ce actiune vrea sa efectueze asupra lor. Totodata, prin intermediul acestuia afisam si pe interfata. Aceasta detine controlul asupra operatiilor effectuate de butoanele pentru fiecare operatii cat si pentru butonul de curatre “CLEAR”. Totodata aceasta clasa face conexiunea dintre tex fielduri si operatii cat si afisarea rezultatului in text area-ul final.

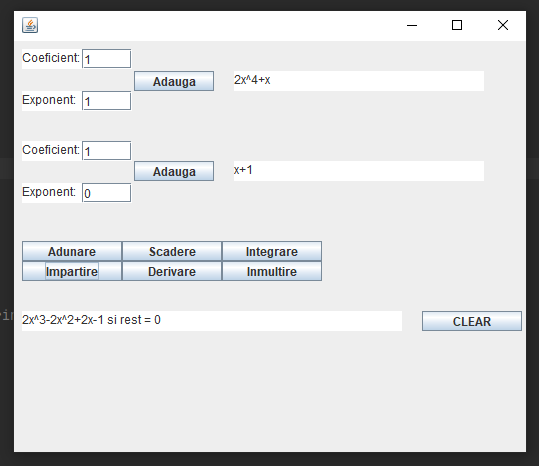
1. GUI ( Graphic User Interface)

Pentru realizarea interfetei grafice am folosit pachetul Java Swing. Am folosit un Frame in care am introdus un singur panel. In acest panel am introdus text fiel-urile, butoanele si text area-urile classic cu ajutorul coordonatelor dupa bunul meu plac.

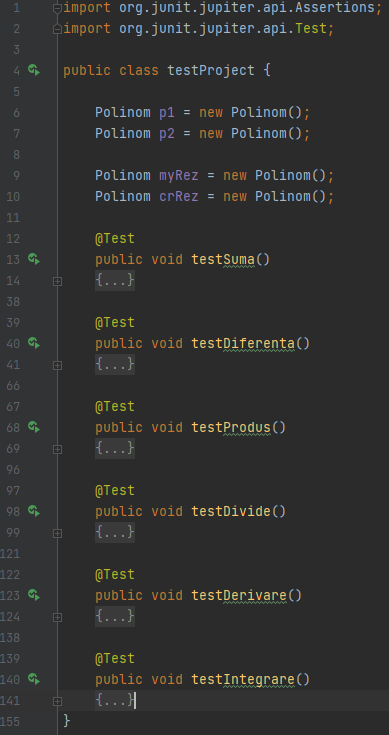


Dupa cum putem observa, aici se gasesc text field-uri pentru introducerea coeficientilor si exponentilor pentru monoamele din polinomul unu cat si pentru cele din polinomul doi, in dreapta fiecarui buton de adauga se gaseste un text area unde se va afisa polinomul construit de noi prin adaugarea fiecarui monom in timp real. Putem observa butoanele pentru fiecare operatie ( 6 la numar ) , text area-ul pentru polinomul rezultat cat si butonul de “CLEAR” .

Asa arata interfata dupa o operatie de impartire efectuata:



1. Testare



Pentru testarea programului am folosit pachetul Junit si pentru fiecare operatie matematica a polinomului am folosit un assertEquals() pentru a verifica daca rezultatul operatiei facute de calculator este egala cu rezultatul operatiei facut de mine matematic pe foaie.

1. Concluzii si posibilitati de dezvoltare

In concluzie acest proiect m-a ajutat din mai multe puncta de vedere. Unul dintre ele ar fi ca am invatat mult mai bine cum se creaza o interfata in stilul MVC si pe ce principii merge cat si cum se teasteaza programele cu ajutorul JUnit. Cel mai important lucru invatat din aceasta dupa parerea mea este ca acum stiu cum sa abordez un program dupa o interfata grafica.

Ca posibilitati de dezvoltare ar fi ca programul as poata sa citeasca direct dintr-ul text field polinomul fara a mai adauga un monom pe rand, cat si unii algoritmi care pot fi imbunatatiti cum ar fi cel de sortare de la operatia de imultire.

1. Bibliografie

* <https://users.utcluj.ro/~igiosan/teaching_poo.html>
* <https://stackoverflow.com/questions/8819842/best-way-to-format-a-double-value-to-2-decimal-places/8819889>
* <https://www.youtube.com/watch?v=0e96sdCVv8s&ab_channel=videomeditatii>
* <https://www.youtube.com/watch?v=Kmgo00avvEw&t=2013s&ab_channel=BroCode>