**DOCUMENTATIE**

**CALCULATOR POLINOMIAL**

NUME STUDENT: Botis George

# CUPRINS

1. Obiectiv………..……………........................................................................................3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare……………………………3
3. Proiectare………………………………………………………………………………4
4. Implementare…………………………………………………………………………..5
5. Concluzie………………………………………………………………………………7
6. **Obiectiv**

Obiectivul proiectului este de a concepe si a implementa un calculator de polinoame cu interfata proprie si usor de folosit. Acest obiectiv poate fi impartit in mai multe mini-obiective care constituie produsul final. Acestea sunt:

1. Analiza problemei si identificarea materialelor necesare
2. Conceperea calculatorului de polinoame
3. Implementarea calculatorului de polinoame
4. Testarea calculatorului de polinoame

-Analiza problemei si identificarea materialelor necesare:

Problema in sine consta in implementarea operatiilor de baza intre polinoame si anume: adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea. Pentru implementarea calculatorului in java s-a folosit spatiul de dezvoltare Intellij.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Scopul programului este de a efectua operatiile simple matematice intre 2 polinoame. Interfata trebuie sa fie simpla, intuitiva si usor de inteles si sa afiseze corect rezultatul la urmatoarele operatii:

1. Adunarea intre 2 polinoame Pr(x) = P1(x) + P2(x)
2. Diferenta dintre 2 polinoame Pr(x) = P1(x) – P2(x)
3. Produsul dintre 2 polinoame Pr(x) = P1(x) \* P2(x)
4. Raportul dintre 2 polinoame Pr(x) = P1(x) / P2(x)
5. Derivarea unui polinom Pr(x) = P1(x)’
6. Integrala unui polinom Pr(x) =

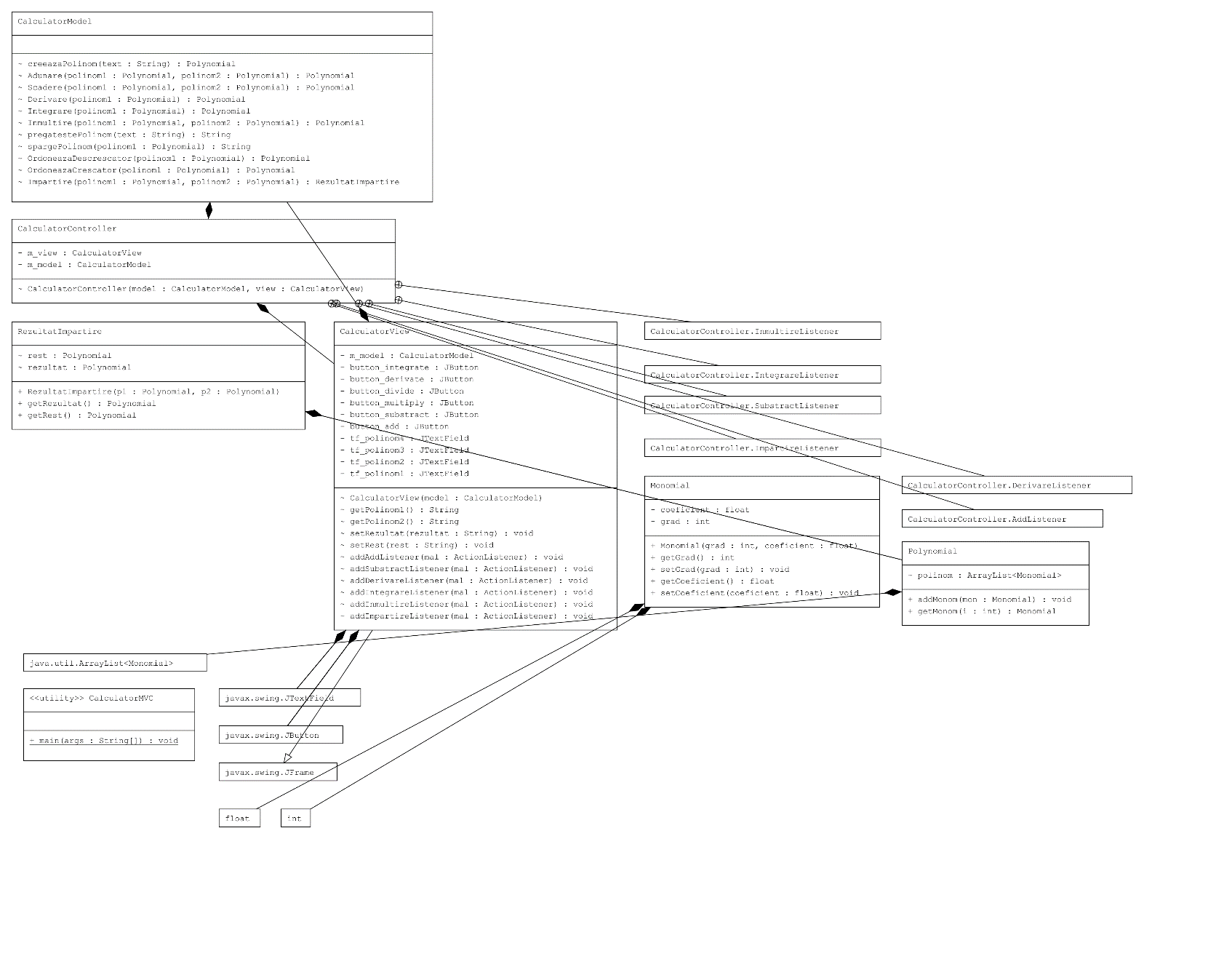
Un caz de utilizare poate fi acela in care utilizatorul introduce corect cate un polinom in cele 2 textfield-uri si apasa pe unul din cele 6 butoane.

Un caz de utilizare in care programul nu o sa functioneze este aceala in care daca utilizatorul introduce gresit polinomul (nu de forma a\*x^b+c\*x^d+…) sau in care introduce cu totul altceva si apasa pe un buton nu o sa produca nici un rezultat.

Un alt caz de utilizare este acela in care utilizatorul introduce doar un polinom si doreste sa efectueze o adunare, scadere impartire sau inmultire dar n-o sa apara nici un rezultat

1. **Proiectare**

Aplicatia contine cele 3 clase descrise anterior si foloseste arhitectura MVC in care se adauga si metodele cu care se efectueaza operatiile.



1. **Implementare**

Proiectul contine 7 clase si mai multe metode.

**Monomial –** Aceasta clasa contine 2 campuri, grad de tip int si coeficient de tip float, care reprezinta elementele principale ale unui monom in matematica. De asemenea, clasa contine settere si gettere si un constructor care seteaza atat gradul cat si coeficientul monomului.

**Polynomial –** Aceasta clasa contine o lista de monoame care reprezinta in mod matematic polinomul. Lista poate primi sau pierde termini folosind setterul si getterul atasati clasei. De asemenea, clasa nu are un constructor definit ceea ce inseamna ca se foloseste constructorul default fara termeni, creeand un polinom fara termeni cand este definit.

**RezultatImpartire –** Aceasta clasa contine 2 polinoame**,** *rest* si *rezultat*. Singurul motiv pentru existenta acestei clase este pentru a transmite mai usor rezultatul impartirii dintre 2 polinoame in program, intrucat este singura operatie care are nevoie de 2 campuri (una pentru rezultat si cealalata pentru rest). Clasa contine de asemenea 2 gettere si un constructor care contine cele 2 polinoame, rest si rezultat.

Celelalte 4 clase implementeaza interfata programului si sunt dupa cum urmeaza:

**CalculatorMVC –** Intrucat programul foloseste arhitectura MVC, aceasta clasa este practic inima programului. Aceasta face legatura intre model, view si controller si le pune impreuna pornind astfel programul.

**CalculatorModel –** Aceasta clasa este clasa cu cel mai mult text din tot proiectul. Aici se afla toate metodele importante care efectueaza operatiile calculatorului de polinoame. Metodele din model sunt dupa cum urmeaza:

*creeazaPolinom (String text)* – metoda primeste un string care reprezinta forma text a polinomului introdus intr-una dintre cele 2 casute pentru introdus de polinoame din interfata calculatorului si returneaza un polinom. Metoda desparte textul in mai multe mini-stringuri despartite de semnul +. Apoi ia fiecare mini-string si cauta pozitia necunoscutei x sau X si apoi, implicand faptul ca mereu in stanga ei se afla \* si in dreapta ei se afla ^ avem astfel coeficientul si gradul.

*adunare(Polynomial polinom1, Polynomial polinom2)* – metoda primeste 2 polinoame si returneaza un polinom ai carui elemente reprezinta suma dintre cele 2 polinoame primite initial. Metoda introduce mai intai in rezultat primul polinom apoi verifica pe rand fiecare element din al doilea polinom daca acesta exista in primul polinom, caz in care o sa schimbe coeficientul monomului cu gradul elementului gasit cu suma coeficientilor din cele 2 polinoame, in caz contrar pur si simplu adauga un nou monom in rezultat.

*scadere(Polynomial polinom1, Polynomial polinom2)* – metoda primeste 2 polinoame si returneaza un polinom ai carui elemente reprezinta diferenta dintre cele 2 polinoame primite initial. Metoda este similara ca implementare cu metoda Adunare.

*derivare(Polynomial polinom1)* – metoda primeste un polinom si returneaza un alt polinom care reprezinta polinomul initial insa derivat. Felul prin care efectueaza acest lucru este prin derivarea fiecarui monom in parte al polinomului folosind regula derivarii.

*integrare(Polynomial polinom1)* – metoda primeste un polinom si returneaza un alt polinom care reprezinta polinomul initial insa integrat. Felul prin care efectueaza acest lucru este prin integrarea fiecarui monom in parte, adunarea intre integrale fiind una dintre proprietatiile integralei.

*inmultire(Polynomial polinom1, Polynomial polinom2)* – metoda primeste 2 polinoame si returneaza un polinom care reprezinta produsul celor 2 polinoame primite initial. Aceasta metoda o sa ia pe rand fiecare termen din primul polinom si o sa il inmulteasca cu cel de-al doilea polinom pastrand rezultatul intr-un polinom temporar. Fiecare polinom temporar este apoi adaugat la rezultat simuland astfel deschiderea de paranteze element cu element.

*pregatestePolinom(String text)* – metoda primeste un string care reprezinta polinomul initial introdus de utilizator intr-una din cele 2 casute pentru polinoame. Ce face de fapt metoda este in felul urmator: Intrucat functia pentru creearea polinoamelor foloseste o comanda care nu poate sa desparta termeni folosind decat un singur caracter (+), aceasta metoda rezolva aceasta problema in cazul in care apare – in loc de +. Intrucat urmatoarea relatie matematica este adevarata: a-b = a+(-b), metoda pur si simplu adauga un + inaintea fiecarui -, ceea ce inseamna ca – va merge la coeficientul monomului si + va ramane drept limitatorul intre monoame. Exista cazul special in care polinomul incepe cu un -. Metoda de descifrare nu recunoaste polinoamele care incep cu + asadar trebuie venit cu o alta solutie. Solutia aleasa a fost de a adauga 0\*x^0+- la inceputul polinomului, ceea ce e echivalent cu a adauga un 0 la inceput, care nu are nici un efect asupra rezultatului dar ajuta la separarea in monoame.

*spargePolinom(Polynomial polinom1) –* aceasta metoda primeste un polinom si va returna un string care reprezinta exact polinomul primit insa sub forma de text.

*ordoneazaDescrescator(Polynomial polinom1) –* aceasta metoda primeste un polinom si va returna un alt polinom care contine aceleasi monoame ca si polinomul initial insa ordonate in ordine descrescatoare. Modul in care realizeaza acest lucru este in felul urmator: o sa creeze un polinom nou in care se stocheaza polinomul ordonat. O sa caute monomul cu cel mai mare grad si il adauga in polinomul final, apoi il sterge din cel vechi. O sa repete acest proces pana cand polinomul initial este gol.

*ordoneazaCrescator(Polynomial polinom1) –* aceasta metoda primeste un polinom si va returna un alt polinom care contine aceleasi monoame ca si cel initial insa ordonate in ordine crescatoare. Modul in care se realizeaza acest lucru este similar cu metoda de mai sus, insa o sa se caute si o sa se insereze mereu monomul cu gradul cel mai mic pana cand polinomul initial este gol.

*Impartire(Polynomial polinom1, Polynomial polinom2) –* aceasta metoda primeste 2 polinoame si returneaza o variabila de tipul RezultatImpartire in care se afla rezultatul si restul impartirii primului polinom la cel de-al doilea. Logica din spatele metodei este urmatoarea: O sa ordoneze in mod descrescator ambele polinoame si o creeze un monom care are ca grad diferenta dintre gradul cel mai mare din primul si cel de-al doilea polinom iar coeficientul ca rezultatul impartirii coeficientului celui mai mare monom din primul la cel de-al doilea. Acest monom este introdus intr-un polinom temporar care este adaugat la rezultat apoi este inmultit cu cel de-al doilea polinom si scazut din polinomul initial, simuland astfel logica din spatele raportului dintre polinoame in matematica. In momentul in care gradul celui mai mare monom din primul polinom este mai mic decat gradul celui mai mare monom din cel de-al doilea polinom, ce mai ramane din primul polinom este transferat in rest si este trimis in variabila de tip RezultatImpartire

**CalculatorView** **–** Aceasta clasa implementeaza partea din interfata cu care interactioneaza utilizatorul.

|  |  |
| --- | --- |
|  | Dupa cum se vede, aceasta interfata contine 4 textfield-uri, primele 2 fiind destinate utilizatorului pentru a introduce polinoamele cu care doreste sa efectueze operatii, celelalte 2 fiind locurile in care apar rezultatul respectivul restul in cazul impartirii.  Interfata dispune de asemenea de 6 butoane care sunt in ordine, butonul de adunare, scadere, inmultire, impartire, derivare si integare a polinoamelor.  Felul in care aceste butoane si textfielduri sunt organizate este in JPanel-uri. Exista doua JPanel-uri principale, si anume unul pentru butoane si unul pentru textfield-uri. JPanel-ul pentru butoane este impartit in alte 3 JPanel-uri mai mici dispuse sub forma unui tabel cu 3 linii si 2 coloane, in care se afle cele 6 butoane. |

**CalculatorController –** Aceasta clasa face legatura intre view si model, aplicand metodele din model si afisand/primind polinoame din view. Clasa contine mai multe subclase de tip ActionListener care au loc atunci cand un buton este apasat si o sa execute un set de instructiuni care corespund cu operatia aleasa.

# Concluzii

In aceasta tema am aprofundat folosirea arhitecturii MVC si am capatat mai multa experienta si capacitatea de a lucra la o viteza mai mare cand vine vorba de java. Posibile dezvoltari ulterioare a programului includ:

-Posibilitatea de a repeta o operatie folosind rezultatul fara a fi nevoie ca utilizatorul sa il copieze si sa il bage intr-unul din textfield-urile pentru adaugat de polinoame

-O metoda care verifica daca textul bagat pentru polinom este corect si care arunca o eroare in caz contrar

-Un buton pentru clear, care sa stearga tot din cele 4 textfield-uri