

**CALCULATOR POLINOMIAL**

Moldovan Liana

Grupa 30223

An universitar 2020-2021

**CUPRINS**

1. Obiectivul temei
2. Analizarea temei
3. Modul de proiectare al temei
4. Implementarea calculatorului
5. Concluzii si inbunatatiri
6. Testarea cu JUnit
7. Bibliografie

**1.Obiectivul temei**

Obiectivul temei este realizarea unui calculator polinomial cu o interfata user friendly, usor de utilizat, care sa realizeze operatiile de adunare, scadere, inmultire, impartire, derivare si integrare. Pentru utilizarea functionalitatii operatiilor de integrare, respectiv derivare, user-ul trebuie sa introduca un polinom (in primul JtextField), in conformitate cu schema generala a unui polinom. Pentru celelalte operatii, este necesar ca utilizatorul sa introduca doua polinoame, iar rezultatul va fi afisat in TextField-ul corespunzator rezultatului. In cazul in care nu se introduc parametrii necesari, pe interfata va aparea un mesaj de eroare.

**2.Analizarea temei**

Polinoamele și operațiile cu acestea sunt unele dintre cele mai comune structuri matematice din Computer Science. Motivul principal este simplitatea algebrică a operațiilor – vorbim doar de adunări și înmulțiri (care la rândul lor sunt adunări repetate). Avantajul este că polinoamele pot aproxima oricât de bine orice funcție continuă și derivabilă, deci pot “interpola” funcții mult mai complexe. Un calculator pentru calcularea diferitelor operatii cu polinoame trebuie implementat tinand cont de toate impedimentele care ar putea aparea. Operatiile de derivare si impartire pot genera un rezultat cu coeficient double, lucru de care trebuie tinut cont. In acest sens, in implementare vom avea nevoie de doua clase, MonomInt pentru coeficientii intregi si MonomDouble pentru coeficientii reali, care vor mosteni clasa de baza Monom, aceasta continand gradul monomului. Ca si structuri de date, am folosit ArrayList-uri pentru stocarea monoamelor unui polinom. Monoamele sunt stocate in liste de la stanga la dreapta, fiecare pozitie din lista corespunde cu gradul monomului respectiv.

In interfata grafica este necesara introducerea polinoamelor in forma generala :



Textul primit de la user sub forma unei expresii regulate(string) si incercam sa facem matching pe patterns, extragand astfel si parsand separat fiecare Monom in parte.Fiecare Monom este extras depinzand de semn, daca are sau nu coefienti, daca are sau nu “x^”. Pentru fiecare Monom incercam sa parsam separat coeficientul si gradul. Conform pattern-ului, nu se admit spatii in interiorul polinomului, coeficienti cu virgula, puteri negative ale monoamelor, alte variabile in polinom decat ‚x’, simbolul ‚^’ neinsotit de un numar intreg, intre doua monoame se admite un singur semn, dar primul monom poate avea semn + sau – in fata. Un monom poate sa nu aiba variabila ‚x’, caz in care este de grad 0, sau sa nu aiba un numar inaintea variabilei ‚x’, caz in care coeficientul sau ar fi 1 sau -1, in functie de semnul monomului, sau sa nu aiba simbolul ‚^’ dupa ‚x’, caz in care gradul sau ar fi 1. Extragerea fiecarui monom din sirul de caractere am realizat-o cu ajutorul pattern-ului ([+-]?[^-+]+).

Tema am implementat-o folosindu-ma de modelul arhitectural MVC(Model View Controller) . Fiecare dintre aceste componente sunt construite pentru a gestiona aspectele specifice de dezvoltare ale unei aplicații.

**3.Modul de proiectare al temei**

In proiectarea temei am folosit cunostintele matematice si am incercat sa elaborez proiectul cat mai usor de inteles.

Clasele folosite in realizarea modelul MVC:

**Model**-Această parte a aplicatiei manipulează operațiunile logice și de utilizare de informație (trimisă dinainte de către rangul său superior) pentru a rezulta de o formă ușor de înțeles. Clasele Model din aceasta aplicatie au grija de propria lor logica de implementare pentru operatii si alte lucruri necesare, astfel facilitand controllerul cu functiile necesare si oferindu I doar rezultatul.

Clasa de baza Monom(abstracta) are atributul grad, iar aceasta este mostenita de Clasele MonoInt si MonomDouble si contine antetul metodelor care urmeaza sa fie implementate de clasele mostenitoare.

MonomInt- coeficientul este intreg, se implementeaza toate metodele clasei de baza, operatiile pe monoame care ulterior se vor folosi pentru calculul diferitelor operatii pe Polinoame. Operatiile de impartire si integrare vor returna MonomDouble.

MonomDouble- coeficient numar real, necesar pentru buna functionalitate a operatiilor de impartire si integrare, se implementeaza toate metodele clasei de baza, operatiile pe monoame care ulterior se vor folosi pentru calculul diferitelor operatii pe Polinoame.

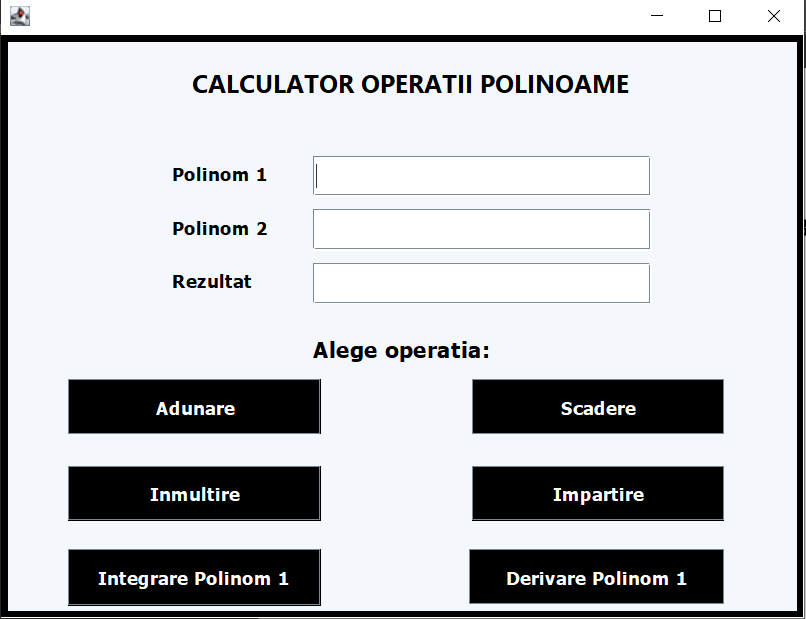
Polinom- are ca si atribut un ArrayList de tipul Monom, unde fiecare Monom este stocate de la stanga la dreapta, fiecare fiind pe pozitia gradului sau.

3x^4 este la pozitia 4, 2x^0 este la pozitia 0 samd. Exista mai multe modalitati de a instantia un polinom, printre care singura care este folosita pentru a instantia polinoamele primite de la user este cea in care primeste un string sub forma de ReGex. Toate celelalte metode de a instantia (contructorii) sunt folosite internal in momentul in care efectuam operatii pe polinoame.

RezultatImpartire- are ca atribute catul si restul de tip Polinom, este folosita pentru operatia de impartire polinoame si este utila pentru a putea returna atat catul, cat si restul in cadrul metodei de impartire din clasa Polinom.

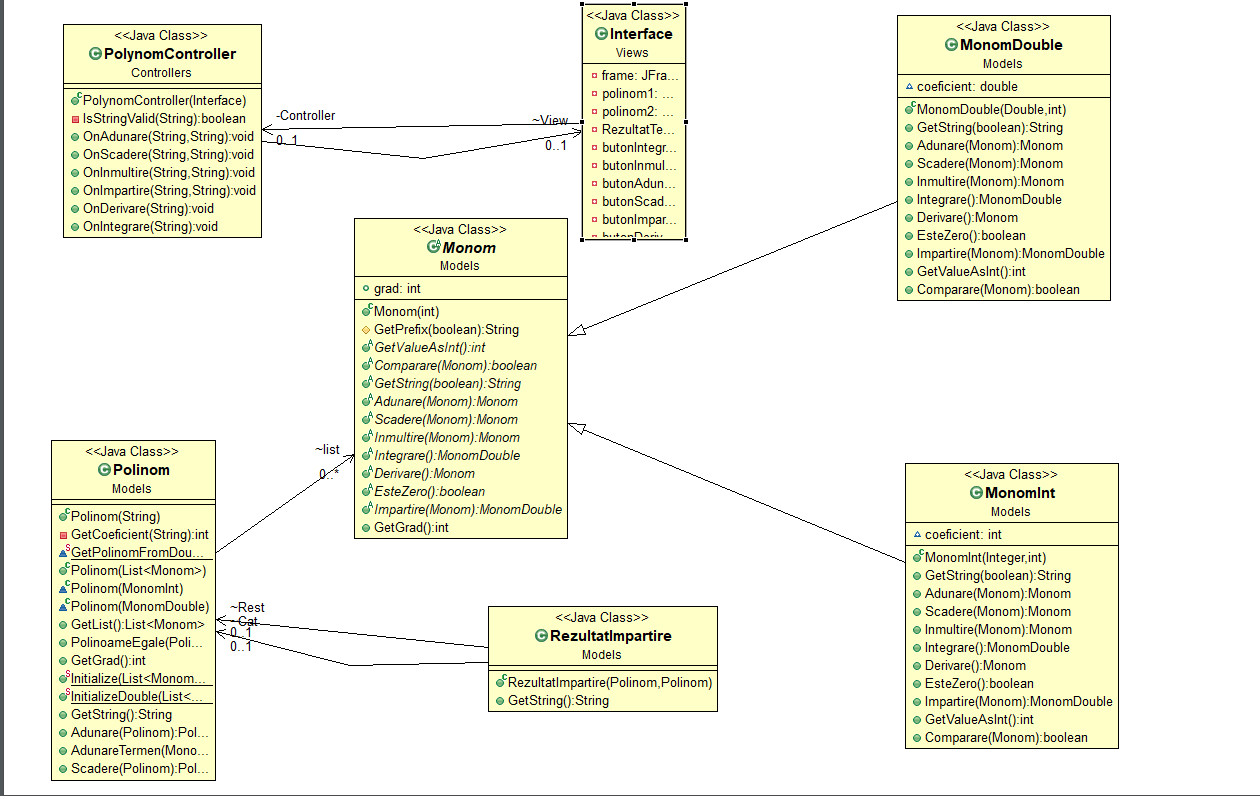
**View-**Acestui membru al familiei îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controlor.

Clasa Interfacecreeaza interfata calculatorului de polinoame, acesta putand fi mai usor de utilizat de catre utilizator. In interfata am doua textField-uri in care utilizatorul trebuie sa scrie polinoamele(sau polinomul in cazul integrarii si derivarii) asupra carora doreste sa efectueze operatiile, un TextField unde se va scrie rezultatul operatiei si un numar de sase butoane, fiecare avand functia specifica numelui. Ea are rost doar pentru vizualizare, toate comenzile ei fiind redirectionate catre controller, care la randul lui ii spune interfetei ce sa faca.



**Controller-** Acest element face conexiunea dintre view si model.

Clasa PolynomController-ea ofera call back-uri pentru interfata, este responsabila de logica necesara, face operatiile cerute cu polioanemele, trateaza cazurile de eroare, dupa care transmite interfetei ce sa afiseze.



In clasa UML se pot observa clasele si relatiile dintre acestea.

**4. Implementarea calculatorului**

Clasele MonomInt si MonomDouble implementeaza toate operatiile necesare pe monoame, pentru a putea fi implementate aceleasi metode pe Polinoame mult mai usor.(Polinom=grupare de mai multe monoame).

In clasa Polinom am implementat metodele pentru operatiile pe polinoame, cat si metode care sa ma ajute la testarea cu Junit si alte metode

1.Adunarea

Pentru adunare am nevoie de doua polinoame, la care le-am comparat gradele pentru a stii de la care sa incep parcugerea. Pentru parcurgere am folosit un foreach in care adun ambele polinoame date de catre utilizator. Cand indicele i din for ajunge la o pozitie mai mare decat gradul celui de-al doilea polinom, ramasitele din primul polinom se aduna cu un Monom de acelasi grad, insa cu coeficient 0 deoarece in adunarea a doua polinoame regula de adunare este de a aduna monomul din primul polinom cu monomul din al doilea polinom care au acelasi grad, adunandu-se doar coeficientul fiecarui monom. Rezultatul fiecarui calcul se insereaza intr-un ArrayList de Monoame avand numele sugestiv suma. La terminarea adunarii celor doua polinoame rezultatul stocat in acel ArrayList este folosit in creearea unui obiect de tip Polinom, iar apoi functia returneaza acest obiect ca si rezultat.

2.Scaderea

Scaderea intre doua polinoame se face folosind functia de scadere din clasa Polinom. Ea primeste ca si parametru un alt Polinom, dupa care afla care dintre cele doua polinoame are gradul cel mai mare, dupa care continua sa inceapa scaderea fiecarui monoam din lista polinoamelor intre ele, folosind functia de scadere din clasa Monom care este implementata in clasele MonomInt si MonomDouble. Scaderea ia in considerare in momentul in care un polinom are gradul mai mare decat celalalt si initializeaza termenul scazut sau scazator cu un Monom nou de gradul necesar. Un nou Polinom este creat din lista rezultatelor si apoi returnat.

3.Inmultirea

Inmultirea se realizeaza cu 2 foreach uri, ordinea in care se parcurg cele doua polinoame nu contand, deoarece rezultatul va fi acelasi. Salvam rezultatul fiecarei inmultiri de Monoame intr-o variabila de tip Monom, calculam gradul rezultat si dorim sa inseram rezultatul in listRezultat. Insa, listRezultat, pe acea pozitie poate avea un rezultat anterior, pe care trebuie sa-l adunat rezultatului curent, pentru a nu exista pierderi. Pentru metoda de inmultire se folosetse si metoda implementata anterior mai sus si anume metoda de adunare pentru a aduna la rezultat monomul rezultat dupa inmultire. La finalul metodei se returneaza un Polinom construit pe baza Listei listRezultat unde se afla rezultatul inmultirii celor doua polinoame, rezultat care ulterior va fi afisat in campul corespunzator din interfata.

4.Integrarea

Pentru integrare vom avea nevoie doar de un singur Polinom, introdus de utilizator in campul aferent primuluo polinom in interfata grafica. Pentru integrarea acestui polinom, in metoda de integrare folosesc un ArrayList de tipul clasei Monom in care se stocheaza rezultatul final al integrarii. Pentru a face integrarea folosesc un foreach in care initial initializez ArrayList-ul folosit pentru rezultat, iar apoi intr-un alt foreach intergrez fiecare MOnom al polinomul dat pentru integrare. La finalul metodei declar un obiect de tipul Polinom pe care il initializez cu ArrayList-ul in care am calculate rezultatul, iar apoi returnez rezultatul integrarii.

5. Derivarea

Pentru derivare vom avea nevoie doar de un singur Polinom, introdus de catre utilizator in campul aferent primuluo polinom in interfata grafica. In aceasta metoda initializez chiar la inceput un ArrayList de tipul clasei Monom in care voi stoca rezultatul derivarii. Pentru implementare folosesc un foreach, in care se ia fiecare Monom pe rand si se deriveaza. La finalul metodei rezultatul salvat in ArrayList va initializa un obiect de tipul Polinom care se va returna ca fiind rezultatul derivarii.

6.Impartirea

Se realizeaza creeand o copie a polinomului impartit, urmand ca mai departe sa se realizeze pe acea copie toate operatiile de scadere, la final ramanand doar restul in acel polinom copiat. In cazul in care impartirea se incearca a se face cu grade incompatibile( primul polinom avand grad mai mic decat al doilea) se va semnala acest lucru in interfata in campul “rezultat”.

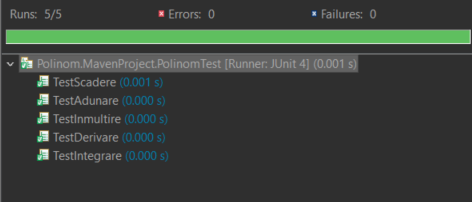
Catul se salveaza intr-un nou Polinom care are membrii doar de tipul MonomDouble si polinomul salvat in copie este impartit cu primul termen care este salvat la inceput in variabila TermenImpartitor urmand ca rezultatul sa fie salvat in Polinomul CatOperatie de tip double, dupa care sa fie inmultit cu tot impartitorul, urmand ca sa fie scazut din copia polinomului impartit. Lucrul acesta se repeta pana cand gradul polinomului impartit este mai mic decat gradul polinomului impartitor.

**Alte metode:**

1. public boolean PolinoameEgale(Polinom a)-folosesc aceasta metoda pentru testarea cu Junit. Functia verifica daca polinomul this este egal cu polinomul transmis ca si parametru.
2. public String GetString()-pentru a transforma polinomul in text am creat o functie GetString care converteste polinomul intr-un string folosindu-se de metoda abstracta GetString din clasa Monom care este implementata in fiecare monom, respectiv MonomInt si MonomDouble. Metoda GetString() din Monom primeste ca parametru un boolean care ii spune daca sa includa sau nu semnul inainte, ea fiind folosita in cazul in care Monomul este primul de la stanga, pentru a nu mai fi pus acel “+” la inceput.
3. Metodele Initialize sunt folosite pentru a initializa lista Polinomului, punand Monoame cu coeficient 0 la gradele care lipsesc din Polinomul dat.
4. public int GetGrad()- extrage gradul polinomului parcurgand lista invers si luand prima pozitie valida;
5. private int GetCoeficient(String S)- returneaza coeficientul fiecarui termen in parte, el tratand cazurile in care coeficientul este 1 si lipseste sau este -1 si lipseste impreuna cu toate celelalte posibilitati prevazute.

**5.Testarea cu Junit**

Testarea operatiilor din clasa Polinom am facut-o cu Junit, pentru a verifica corectitudinea rezultatului lor. Au fost facute teste atat pentru adunare,scadere cat si pentru inmultire, derivare, integrare, fiecare dintre teste a fost efectuat folosing assert-ul specific assertTrue, luand in considerare rezultatul functiei PolinoameEgale, care a fost create pentru aceste teste, care verifica daca un polinom primit este egal cu polinomul this si acestei functii ii se da ca parametru un polinom creat manual, care se presupune ca este rezultatul corect. Testul returneaza true daca polinoamele coincid.



**6.Concluzii si imbunatatiri**

Pentru elaborarea temei mi-am folosit in mare parte cunostintele dobandite la cursul de POO iar unde a fost nevoie de aprofundare sau intelegerea unor concepte noi am folosit diferite site-uri de programare.

In ceea ce priveste imbunatatirile, consider ca o tastatura in cadrul interfetei ar fi utila, de asemenea un design mai interesant ar acesteia ar capta mai mult atentia utilizatorului. De asemenea , ideea de a converti dintr-o bază polinomială într-o alta pare foarte atractivă, necesitând algoritmi pentru manipularea matricilor de transformare a bazelor. Mai mult, am putea trasa graficele pentru polinoame.

**7.Bibliografia**

<https://regexr.com/>

<https://www.w3schools.com/java/java_regex.asp>

https://www.tutorialspoint.com/design\_pattern/mvc\_pattern.htm