**DOCUMENTATIE TEMA 1**

**CALCULATOR POLINOAME**

**Filip Vlad-Ilie**

**Grupa 30222**

**Profesor Laborator: Dorin Moldovan**

Cuprins:

1. Cerinte functionale ………………………………………………………………………… 1

2. Obiective ……………………………………………………………………………………. 1

2.1 Obiectiv principal ………………………………………………………………………...………….. 2

3

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

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

3.1 Use case-uri / scenarii ………………………………………………………………………………. 4

3.2 Diagrama UML ………………………………………………………………………………………..5

3.2 Algoritmi……….………………………………………………………………………………………..5

4.Proiectare ……………………………………………………………………………………………… 6

5.Testare…. ………………………………………………………………………………………………7

6.Concluzii si dezvoltari ulterioare……………………… …………………………………………… 4

7.Bibliografie..…………………………………………………………………………………………… 4

# Cerinte Functionale

( cerintele problemei )

Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila care are in componenta coeficienti intregi.

Operatiile pe care le suporta programul sunt:

* Citirea unui polinom de la tastatura sub forma de String cu formatul a0x^n+a1x^(n-1)+a2x^(n-2)+...+a0, unde si
* Adunarea a doua polinoame P1(x) + P2(x)
* Scaderea a doua polinoame P1(x) - P2(x)
* Inmultirea a doua plinoame P1(x) \* P2(x)
* Derivarea unui polinom P1(x)’
* Integrarea unui polinom ∫ P1(x)

# Obiective

## Obiectiv Principal:

(obiectivul principal al proiectului)

Obiectivul principal al proiectului este de a creea o aplicatie care sa implementeze un sistem de procesare a polinoamelor ( acestea fiind construite din termeni numiti monoame, care sunt alcatuite dintr-o constanta (numita coeficient), inmultita cu una sau mai multe variabile; fiecare variabila poate avea un exponent constant intreg pozitiv).

## Obective Secundare:

( pasii pe care trebuie sa-i urmam pentru a atinge obiectivul principal )

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii | Într-un sistem software un ‘use-case’ este o listă de acțiuni sau pași de eveniment care definesc în mod obișnuit interacțiunile dintre un rol ( cunoscut ca un actor în limbajul unificat de modelare ( UML ) ) și un sistem in atingerea unui obiectiv.  In computere, un scenariu este o naratiune a interactiunilor dintre roluri ( actori in UML ) si sistem. In cazul nostru rolul - un utilizator al aplicatiei. | 3 |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capat obiectivul principal | 4 |
| Impartirea pe clase | Folosirea unui MVC ( Model – View – Controller) pentru a putea forma un GUI ( Graphic User Interface ), a clasei Polinom, Monom etc. | 4 |
| Dezvoltarea algoritmilor | Vor fi descrise structurile de date necesare pentru atingerea obiectivului principal, schema UML ( diagrama de clase ) precum si algoritmii folositi pentru realizarea operatiilor. | 4 |
| Implementarea solutiei | Vor fi descrise fiecare clasa cu campurile si metodele importante precum si descrierea interfetei utilizator; | 5 |

# Analiza Problemei

( use case-uri / scenarii care pot sa apara )

**3.1 Use case-uri / Scenarii**

Utilizarea programului presupune introducerea de catre utilizator (in TextField-uri ) a celor doua polinoame ce urmeaza a fi folosite pentru efectuarea anumitor operatii ( adunare, scadere, inmultire, impartire, derivare, integrare ). Prin apasarea unor Butoane ce au semne specifice pentru operatii, intr-un alt TextField o sa apara rezultatul asteptat.

Deoarece e posibil ca unii utilizatori sa introduca polinoame gresite, de fiecare data cand un polinom introdus in TextField are un format gresit fata de cel normal al unui polinom, sau sunt introduse diferite Stringuri care nu formeza un polinom ( ex. Ana are mere ), la apasarea unui buton ce semnifica operatia va aparea o fereastra ce va avertiza utilizatorul ca polinomul introdus de el nu este valid ( nu respecta formatul ). E posibil ca utilizatorul sa introduca polinomul cu grade diferite, calculatorul isi va da seama si le va calcula correct fara a conta ordinea in care monoamele se afla in polinom.

Proiectare

( etapele de proiectare )

## 

## Structuri de date

(structurile de date folosite in program)

Ca si structura de data principala ce am folosit-o este un ArrayList de Monoame (deoarece un polinom este format din mai multe monoame, un arraylist ar fi fost cel mai potrivit in rezolvarea acestei probleme ).

private ArrayList<Monomial> monomial = new ArrayList<Monomial>();

Un monom este format dintr-un coeficient, un grad si o variabila ( ex. 2X^3 , unde 2 – coeficientul, x – variabila, iar 3 – gradul ( puterea ) ). Monom-ul va fi in java o clasa separata care va contine ca variabile instanta un intreg, reprezentat de putere ( grad ) si un float care reprezinta coeficientul ( acesta va fi de tip float deoarece la anumite operatii, se poate obtine un numar care contine , : ex ½ =0.5 ).

## Diagrama de clase

A close up of text on a white background

Description automatically generated

(schema UML a proiectului )

## Algoritmi

Am folosit operatiile de baza adunare,scadere,inmultire,derivare,integrare, asemenea in matematica ,doar ca trebuai sa le aplicam monoamelor de aceea in aceasta faza am intampinat niste dificultati dar in final am reusit sa realizez. A fost nevoie si de niste biblioteci externe, am importat java.util.\* pentru a putea folosi regex pattern sau matcher, java.swing.\* pentru a putea folosi in clasa view butoane, label-uri si textfield-uri.

# Implementare

In clasa principala avem urmatorii parametri:lista de monoame in care stocam datele despre fiecare monom, numarul de monoame, un string care reprezinta pattern-ul dupa care se aleg monoamenele si o constanta care reprezinta gradul maxim al polinomelor, ulterior se poate modifica in functie de situatie si de problema. In clasa polinom avem mai multi constructori pentru polinom , unul in care nu avem nicio valoare Polinom() care nu face nimic, a fost folosit pentru crearea unor polinoame care ulterior au fost actualizate. Un constructor pentru care are ca paramentru String-ul pentru regex si care creeaza polinomul in functie de string. Alt constructor care are ca parametri o lista de polinoame si de un numar de monoame si atribuie polinomului lista de monoame si numarul de monoame si inca un constructor cu numarul de monoame care creeaza un polinom, iar in lista de monoame vor fi coeficienti =0 iar se vor crea monoame pana se va atinge numarul de monoame date ca atribut.

Pentru lista de monoame , numarul de monoame avem setter si getter, pentru a le putea accesa deoarece ele sunt trecute in private pentru a nu putea fi accesate din alte clase decat cu ajutorul acestor metode.

Am creat o metoda care adauga un nou monom la lista de monoame deja existente si dupa adaugare se mareste numarul de elemente cu 1.

Suprascrierea metodei toString() a fost necesara deoarece dupa fiecare operatie cu monoamele din lista de monoame acestea trenuiau sa fie scrie in forma lor din definitie (ax^b) asa ca am folosit un string in care am reprodus aceasta formula cat mai exact.

Pentru a calcula gradul maxim al unui polinom care parcurge lista de monoame si actualizeaza o gradul cu fiecare monom.

Pentru a aduna doua polinoame a fost nevoie de polinoamele care rezultau din string-ul de la interfata grafica. Aici a fost nevoie de pattern pe care l am folosit pentru a putea sa separ fiecare monom din string si dupa inca 2 pattern-uri pentru a separa coefficient de grad. Pentru a le aduna am folosit adunarea monom cu monom. Asemenea adunarii am procedat asemenea si al scadere, inmultire, derivare, si integrare.

Pentru a putea scoate din String monomul de care avem nevoie, am verificat daca string-urile sunt valide prin verificarea fiecarui caracter.

Clasa Monom care repezinta elementele din lista Polinomului are ca parametri, coeficientul, care este un numar real, deoarece in unele operatii e nevoie de rezultatul exact, si de grad care reprezinta un numar natural pozitiv, alaturi de aceste doua elemente fundamentale ale clasei Monom se afla si String-urile pentru detectarea de coeficient si grad din fiecare string monom. In clasa monom se afla un constructor Monom() care nu returneaza nimic folosit pentru a crea un monom, folosit ulterior la crearea propriu-zisa . Un constructor care are ca parametri doua int-uri care reprezinta coeficientul si gradul care atribuie monomului aceste valori, si un constructor care are ca parametru un string care este folosit pentru a separa coeficientul si gradul din string si creaza un monom cu aceste valori.

In aceeasi clasa trebuia suprascrisa metoda toString() in care s-a facut constructia propriu-zisa, in forma monoamelor clasice ax^b unde a reprezinta coeficientul si b gradul.

Tot in aceasta clasa au fost implementate operatiile de :

* Adunare P1(x)+ P2(x)
* Scadere P1(x)-P2(x)
* Inmultire P1(x)\* P2(x)
* Derivare P1(x)’
* Integrare ∫ P1(x)

Pentru interfata cu utilizatorul a fost implemenata aplicatia sub forma MVC (Model-View-Controller). In view se afla interfata pe care o vede utilizatorul,cu butoane , textfield si label-uri, model-ul reprezinta puntea dintre controller si view in care sunt legate polinoamele la string-uri si in care se string-urile din view vor fi folosite pentru a le converti la polinom si invers. In controller se afla cea mai importanta parte din cod, controllerul reprezinta creierul acestui proiect unde acolo totul capata sens si astfel putem sa definim sa atribuim fiecare buton apasare si astfel toate ideile capata niste contururi reale. Si la final clasa principala main-ul proiectului este MVC unde aici se afla public static void main cu args, aici avem declarate numai model-ul view-ul si controller-ul si si atribuirea la controller a view-ului si model-ului.

# Testare

Pentru testare de exemplu daca utilizatorul introduce 6x^2-3x^1-4 , penteu primul polinom si 3x^2+2x^1+1, pentru al doilea polinom si va apasa pe butonul de adunare, string-ul din textfield-ul din view, cu ajutorul regex, pattern si matcher se separa fiecare monom care va fi atrbuit unui polinom din clasa Polinom, dupa in aceeasi metoda vac rea un nou polinom cu grad maxim in care fiecare monom va fi creat cu coeficientul 0 si cu gradul unui contor care va pleca de la 0 pana la gradul maxim. Aces polinom cu coefficient 0 se va aduna cu primul polinom cu conditia daca gardul celor doua polinoma e e egal atunci se vor aduna coeficienti si dupa suma celor doua polinome va fi stocata in primul, cel cu gradul maxim, dupa aceeasi metoda si aplica si cu al doilea polinom, suma lor este pastrata in polinomul cu grad maxim. La final polinomul maxim se atribuie polinomului care a apleta metoda astfel se va stoca in unul din cele doua polinoame ne fiind nevoie si de un polinom splimentar. Aceasta metoda a fost aleasa pentru evitarea cazului in care cele doua polinoame au grade diferite si astfel suma lor va fi stocata in polinomul cu grad maxim. La citire monoamele care vor avea coefficient =0 vor fi evitate. Pentru scadere pentru acelasi exemplu am folosit acceeasi tactica doar la prima data am adunat polinomul cu grad maxim la primul polinom iar dupa aceea din polinomul suma am scazut al doilea polinom. Argumentul pentru folosirea acestei metode este acelasi ca si in cazul adunarii.

Pentru inmultirea a doua polinoame, se vor folosi string-urile din interfata se vor comverti in monoamen si dupa aceea se va crea un nou polinom care va fi format din produsul celor doua monoame. Produsul celor doua monoame reprezinta parcurgerea celor doua monoame si inmultirea fiecarul monom din cele doua polinoame. Inmultirea se realizeaza astfel fiecare coefficient dintr-un monom se va inmulti cu fiecare coeificient din celalalt monom iar gradele celor doua monoame se vor aduna. Pentru derivarea monomului, se va lua primul monom scris in textfiel-ul din interfata grafica, iar derivarea se va realiza mathematic coefficientul fiecarui monom se va inmulti cu gradul iar gradul se va scadea cu o unitate, daca gradul este 0 atunci coeficientul devine 0. Pentru integrare, asemenea derivarii a fost folosita matematica adica pentru fiecare monom di polinom se va coeficientul se va imparti la grad+1 iar gradul se va incrementa cu o unitate.

Rezultatele fiecarei operatii se vor afisa in textfield-ul pentru rezultat, este posibila schimbarea operatiilor fara a fi nevoie de rescrierea polinoamelor deci pentru aceleasi polinoame se poate calcula adunarea scaderea inmultirea, fara schimba vreunul dintre aceste polinoame. Nu am reusit sa implementez junit-urile pentru operatiile polinoamelor de aceea nu pot demonstra functionalitatea lor, dar din testele facute programul functioneaza normal.

# Concluzii si Dezvoltari Ulterioare

In concluzie aceasta tema care la inceput parea sa fie usor de implementat a fost destul de greu de realizat deoarece, nu realizarea operatiilor a consumat cel mai mult timp ci realizarea interfetei si convertirea string-ului in polinom si dupa aceeea in monom. Folosirea pattern-ului si matcher-ului pentru string a reprezentat o dificultate mai ridicata. Petru o dezvoltarea ulterioara as recomanda niste metode de eficienta deoarece codul mi se pare hard codat putin in unele privinte deoarece nu am stiut exact cum sa implementez mai eficiente prin reducerea numarului de atriburi si de variabile.

# Bibliografie

<https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression>

<https://ideone.com/cgaleS>