DOCUMENTATIE

TEMA *1*

NUME STUDENT: Circeie Gavrila-Nicolae

GRUPA: 30232

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\Gabi%20Circeie\Downloads\PT2021-2022_Documentation_Template%20(3).doc#_Toc95297891)

# Obiectivul temei

Obiectivul acestui proiect este de a realiza o aplicatie de calcul polinomial cu o interfata interactiva a utilizatorului prin care acestuia ii este permis sa efectueze sase operatii polinomiale de baza: adunare, scadere, multiplicare, divizare, derivare si integrare. Utilizatorul ar trebui sa poata introduce singur aceste polinoame si sa vada ceea ce o sa se afiseze pe ecran. In viziunea mea, scopul acestor teme este de a permite studentilor sa practice conceptele Java si OOP prin construirea aplicatiei mentionate, care necesita unele abilitati pe care studentii le dezvolta in timp ce fac acest lucru.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Aceasta problema parea simpla la prima vedere, deoarece operatiile polinomiale obisnuite nu sunt o sarcina dificila in matematica, dar treptat am realizat ca, in programare, este necesara o abordare eficienta si modulara pentru a rezolva aceasta problema. M-am gandit mai intai la doua clase principale care ar modela obiectele: clasa Monom si clasa Polinom. Cele doua sunt discutate mai detaliat in capitolele viitoare. In ceea ce priveste interfata cu utilizatorul, am considerat ca este nevoie de o interfata grafica simpla si minimalista pentru a fi usor de inteles de catre orice utilizator.

2.1. Adunarea a doua polinoame – presupune adunarea coeficientilor termenilor cu acelasi exponent

Scenariu :

* Utilizatorul introduce cele doua polinoame de la tastatura calculatorului
* Polinoamele introduse sunt analizate, verificandu-se formatul acestora
* Se creeaza polinoamele
* Se executa operatia de adunare asupra lor
* Se afiseaza rezultatul .

2.2. Scaderea a doua polinoame – presupune scaderea coeficientilor termenilor cu acelasi exponent .

Scenariu :

* Utilizatorul introduce cele doua polinoame de la tastatura calculatorului
* Polinoamele introduce sunt analizate, verificandu-se formatul acestora
* Se creeaza polinoamele
* Se executa operata de scadere asupra lor
* Se afiseaza rezultatul operatiei

2.3. Inmultirea a doua polinoame – presupune inmultirea tuturor termenilor din primul polinom pe rand cu fiecare termen din al doilea polinom

Scenariu:

* Utilizatorul introduce cele doua polinoame de la tastatura calculatorului
* Polinoamele introduse sunt analizate, verificandu-se formatul acestora
* Se creeaza polinoamele
* Se executa operatia de inmultire asupra acestora
* Se afiseaza rezultatul operatiei .

2.4. Derivarea unui polinom – presupune inmultirea coeficientului cu exponentul si scaderea exponentului cu 1, in cazul in care acesta este mai mare decat 0.

Scenariu :

* Utilizatorul introduce polinomul de la tastatura calculatorului
* Polinomul introdus este analizat, verificandu-se formatul acestuia
* Se creeaza polinomul
* Se exectura operatia de derivare asupra ambelor polinoame
* Se afiseaza rezultatele operatiilor .

2.5. Integrarea unui polinom – presupune impartirea coeficientului cu exponentul + 1 si adunarea exponentului cu o unitate.

Scenariu :

* Utilizatorul introduce polinomul de la tastatura calculatorului
* Polinomul introdus este analizat, verificandu-se formatul acestuia
* Se creeaza polinomul
* Se executa operatia de integrare
* Se afiseaza rezultatele operatiilor .

In randurile urmatoare, am analizat cateva use cases pe care le consider relevante pentru aplicatie:

Titlu: Efectuarea unei operatii intre doua polinoame.

Descriere: Utilizatorul introduce doua polinoame si alege operatia pe care doreste sa o efectueze din urmatoarea lista: adunare, scadere, inmultire, impartire, derivare, integrare. Toate aceste operatii se fac printr-un simplu clik pe butonul cu operatia corespunzatoare.

Persoana care utlizeaza: Oricare utilizator al aplicatiei.

Conditii prealabile: Utilizatorul este obligat sa introduca doua polinoame, fara spatii intre monoame si in ordinea descrescatoare a gradelor monoamelor.

Scenariul meu de succes:

1. Utilizatorul introduce doua polinoame corecte;

2. Utilizatorul face click pe operatia dorita;

3. Cele doua polinoame sunt citite si interpretate corect;

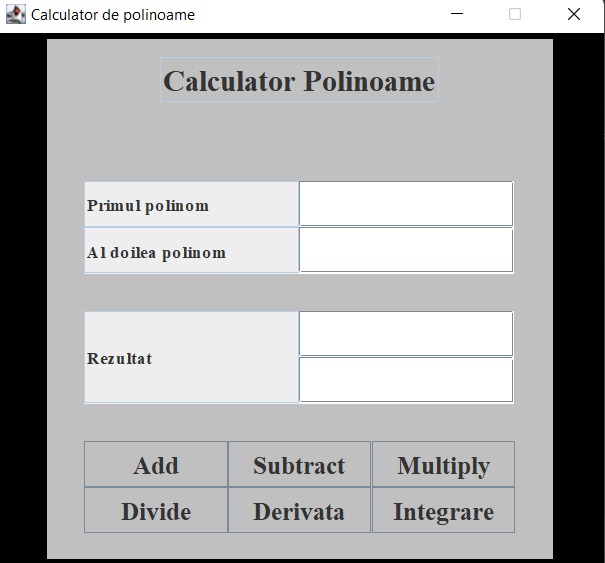
4. Se obtine si se afiseaza rezultatul corect;

Frecventa utilizarii: Foarte des, aceasta fiind functionalitatea de baza a aplicatiei.

# Proiectare

3.1. GUI

Interfata grafica este simpla si intuitiva, deoarece consider ca utilizatorul ar trebui sa-si poata da seama cum sa foloseasca aplicatia. Atasat mai jos este o captura de ecran a interfetei cu utilizatorul si explicatia ei imediat dupa.



Fereastra este un obiect JFrame simplu de 500 x 470 care contine 7 JPanel-uri, dupa cum urmeaza:

- Primul JPanel (titlePanel) contine titlul aplicatiei noastre si anume “Calculator de polinoame” – cu FlowLayout.

- Al doilea JPanel (inputPanel) contine reprezinta zona unde se pot introduce cele 2 polinoame prin intermediul carora vor fi efectuate operatiile – cu GridLayout.

- Al treilea JPanel (outputPanel) cuprinde JTextField-ul pentru rezultat - GridLayout.

- Al patrulea JPanel (resultArea) cuprinde cele 2 JTextField-uri unde vor fi afisate rezultatele – cu GridLayout.

- Al cincelea JPanel (buttonPanel) reprezinta zona de butoane – cu GridLayout.

- Al saselea si al saptelea JPanel (originPanel1 si originPanel2) au rolul de a unifica celelate 5 JPanel-uri si de a da textura GUI-ului prin alternanta de culoare negru-gri.

Am ales sa dezvolt aplicatia astfel, deoarece am vrut ca experienta utilizatorului sa fie cat mai placuta. Rezultatul operatiei este dat prin apasarea butonului de functionare, fara a fi nevoie sa se apese din nou pe alte butoane, spre deosebire de a avea un buton de egal. Mai mult decat atat, aplicatia are nevoie de cele doua polinoame (mai putin in cazul calculului derivatei si a integralei) de intrare si efectueaza calculele pe cele doua indiferent de numarul de apasari ale unui buton, spre deosebire de situatia in care, spre exemplu ar fi proiectat un singur camp de text pentru introducere si apasarea butonului de adaugare (ex: prin apasarea de cinci ori a butonului ar insemna ca polinomul de intrare sa fie adaugat de cinci ori). In viziunea mea, aceasta abordare este mai usor de inteles si implementat ca un exercitiu de invatare ce se comporta ca un sistem combinational, in loc de unul secvential. Am decis sa ma concentrez pe calitatea programarii din spatele aplicatiei si aceasta implementare mi-a permis sa scriu un cod Java usor de inteles. De asemenea, am decis sa permit intrari fara spatii, deoarece expresiile pot deveni greu de citit avand multe spatii. Nu exista nicio limita pentru gradul polinoamelor de intrare, dar monoamele lor trebuie sa fie in ordinea descrescatoare a gradelor lor.

3.2. Pachete

Singurele pachete create de mine sunt pachetele de baza, pachetul principal, care contine toate clasele cu codul de implementare si pachetul de testare cu clasa de testare a aplicatiei. Am considerat ca folosirea de pachete pentru modelul de proiectare MVC este redundanta, intrucat am avut un singur Controller si un singur View .

3.3. Structuri de date

Unele structuri de date mai notabile pe care le-am utilizat sunt ArrayLists pentru listele de monoame ale unui polinom. Alte structuri de date sunt doar valorile primitive.

# Implementare

* 1. **Main**

Clasa Main este clasa generata standard cu metoda principala. Singurul scop, din punctul meu de vedere al implementarii, este sa instantieze cele doua clase, View si Controller, si sa se faca legatura dintre ele.

Metode implementate:

public static void main(String args[]) – metoda main apelata la executarea proiectului.

* 1. **Monom**

Clasa Monom este prima clasa care proiecteaza un obiect, un monom. Scopul sau este de a detine un singur monom. Nu am implementat operatii cu monoame in aceasta clasa, deoarece le-am considerat banale si, prin urmare, redundante. Aceasta clasa respecta principiul incapsularii si toate campurile sunt private. Accesul la acestea este limitat prin intermediul getters si setters.

Campuri:

* private Double coeficient: coeficientul monomului
* private int putere: puterea monomului

Constructori:

* public Monom(Double coeficient, int putere): instantiaza un nou obiect de Monom cu un coeficient si o putere date ca parametru

Metode implementate:

* public Double getCoeficient: getter pentru coeficient
* public void setCoeficient(Double coeficient): setter pentru coeficient
* public int getPutere(): getter pentru putere
* public void setPutere(int putere): setter pentru putere
  1. **Polinom**

Clasa Polinom este a doua clasa care proiecteaza un obiect, un polinom. Un polinom este reprezentat ca un HashMap de tipul Integer and Double. Aceasta este, fara indoiala, cea mai importanta clasa a proiectului, deoarece implementeaza toate operatiunile pe polinoame. Aceasta clasa, ca si clasa Monom, respecta principiul incapsularii si toate campurile sunt private. Accesul la acestea este limitat prin intermediul getters si setters.

Campuri:

* private HashMap<Integer, Double >
* private int gradPolinom: gradul maxim al monoamelor ce da gradul polinomului

Constructori:

* public Polinom(HashMap<Integer, Double > polinom): instantiaza un obiect de tip Polinom caruia i se atribuie o lista data de monoame si se determina si gradul polinomului respectiv.

Metode implementate:

* public Polinom add(Polinom a, Polinom b): adauga un polinom la un alt polinom prin succesiunea de monoame avand grade identice si returneaza lista de monoame astfel rezultata.
* public Polinom subtract(Polinom a, Polinom b): scade un polinom la un alt polinom prin succesiunea de monoame avand gradul identic si returneaza lista de monoame astfel rezultata.
* public Polinom multiply(Polinom a, Polinom b): inmulteste un polinom la un alt polinom si returneaza lista de monoame astfel rezultata.
* public Polinom derivate(Polinom a, Polinom b): determina polinomul derivat din polinomul dat ca obiect al metodei diff si returneaza lista de monoame astfel rezultata.
* public Polinom integrate(Polinom a, Polinom b): determina polinomul integral din polinomul dat ca obiect al metodei integrate si returneaza lista de monoame astfel rezultata.
* public String toString(): returneaza polinomul rezultat in urma uneia dintre operatii ca un String.
* gettere si settere pentru campurile private

**1.4. Controller**

Aceasta clasa corespunde controlerului in modelul de proiectare MVC si detine logica din spatele aplicatiei. Controller-ul contine algoritmul de analiza polinomiala si seteaza actiunile pentru butoanele din interfata grafica.

Campuri:

* private View graphicalUserInterface: camp al interfetei grafice

Constructori:

* public Controller(View graphicalUserInterface): instantiaza un nou Controller care primeste o interfata grafica gui

Metode implementate:

* public HashMap<Integer, Double> getPolinom (String polynom): parseaza String-ul primit de catre utilizator si il grupeaza pe 2 grupuri: primul grup daca are semnul minus sau plus in fata, al doilea grup daca are un coefficient. Se va construi astfel un monom cu coeficientul si puterea respectiva. Functia va returna o lista de astfel de monoame create.
* private void OperationsListeners(): sunt setate actiunile de pe butoanele din interfata grafica.

**1.5. View**

Clasa View, dupa cum ii mentioneaza si numele, implementeaza partea de interfata a modelului de proiectare MVC.

Campuri: JFrame-ul pentru initializarea ferestrei, JPanel-uri pentru plasarea ordonata a JTextField-urilor si a butoanelor, precum si un camp de nume pentru interfata grafica.

Constructori:

* public View(String name): creeaza o instanta a clasei View si efectueaza operatiile pentru crearea si atribuirea de JPanel-uri si JFrame.

Metode implementate:

* gettere si settere pentru campurile private.

# Rezultate

- add() : se introduc doua polinoame sub forma de String si rezultatul adunari acestora tot sub forma de String si se compara cel de al treilea polinom cu rezultaul operatie de adunare . Exemplu: polinom1=”2X^5+X+2” , polinom2=”4X^2+X”, rezultat=”2x^5+4x^2+2x+2”, iar rezultatul comparatiei este “true”;

- sub() : se introduc doua polinoame sub forma de String si rezultatul scaderi acestora tot sub forma de String si se compara cel de al treilea polinom cu rezultaul operatie de scadere . Exemplu: polinom1=”2x^5+x+2” , polinom2=”4x^2+x”, rezultat =”2x^5-4x^2+2”, iar rezultatul comparatiei este “true”;

- multiply() : se introduc doua polinoame sub forma de String si rezultatul inmultiri acestora tot sub forma de String si se compara cel de al treilea polinom cu rezultaul operatie de inmultire . Exemplu: polinom1=”2x^5+x+2” , polinom2=”4X^2+X”, rezultat= “8x^7+2x^6+4x^3+9x^2+2x”, iar rezultatul comparatiei este “true”;

- diff(): se introduc doua polinoame sub forma de String, unul fiind polinomul de derivat si celalalt polinomul derivat si se compara cu cel rezultat in urma metodei de derivare. Exemplu: polinom1=”2x^5+x+2” si rezultat=”10x^4+1”, iar rezultatul comparatiei este “true”;

- integrate(): se introduc doua polinoame sub forma de String, unul fiind polinomul de integrat si celalt polinomul integrat si se compara cu cel rezultat in urma metodei de integrare. Exemplu: polinom1=”2x^5+x+2” si rezultat=”0.33x^6+0.5x^2+2x ”, iar rezultatul comparatiei este “true”;

# Concluzii

Consider ca acest assignment m-a invatat multe despre impartirea codului in metode in asa fel incat nicio metoda sa nu contina o cantitate imensa de cod. De asemenea, am castigat si putina experienta in ceea ce priveste codarea in Java. Am invatat cum sa folosesc expresii regulat si cum sa configurez o interfata mai buna, lucru pe care l-am tot evitat.

In ceea ce priveste o dezvoltare viitoare a aplicatiei, cred ca este nevoie de un interpretare mai buna a polinomului de intrare, care sa accepte atat spatii intre monoame, cat si sa sorteze monoamele, astfel incat sa nu fie necesara introducerea polinomului intr-o anumita ordine a monommelor. O alta dezvoltare viitoare ar fi implementarea unui mod de utilizare a rezultatului ca polinom fara a fi necesar sa fie rescris manual, salvand practic polinoamele in memorie.

# Bibliografie

https://dsrl.eu/courses/pt/materials/A1\_Support\_Presentation.pdf