

# **DOCUMENTAȚIE**

Tehnici de programare

Tema 1

Giesswein Alexia  
Grupa 30225

# CUPRINS

1. Obiectivul temei .....	3
2. Analiza problemei.....	4
3. Proiectare .....	4
4. Implementare .....	6
5. Rezultate .....	8
6. Concluzii.....	9
7. Bibliografie .....	9

# Obiectivul temei

Obiectivul principal al temei este proiectarea și implementarea unui calculator polinomial cu o interfață grafică prin care utilizatorul poate introduce două polinoame de la tastatură, poate selecta o operație matematică care să fie efectuată, precum adunarea, scăderea, înmulțirea, împărțirea, derivarea și integrarea, și poate vizualiza rezultatul.

Obiectivele secundare sunt:

- Analizarea problemei și identificarea cerinței.
- Proiectarea calculatorului polinomial.
- Implementarea calculatorului polinomial.
- Implementarea operațiilor matematice cu polinoame (adunare, scădere, înmulțire, împărțire, derivare, integrare).
- Convertirea unui șir de caractere într-un polinom.
- Integrarea interfeței grafice.
- Folosirea unei structuri.
- Testarea calculatorului polinomial folosind JUnit.

# Analiza problemei

Un calculator polinomial trebuie să îndeplinească următoarele cerințe:

- Utilizatorul trebuie să poată introduce cele două polinoame cu care se vor efectua operațiile matematice.
- Dacă cel puțin unul dintre cele două polinoame nu este introdus corect, programul va afișa un mesaj de eroare.
- Cele două polinoame introduse sunt de fapt șiruri de caractere, care trebuie convertite în obiecte polinom cu care programul să efectueze operațiile matematice.
- Când utilizatorul apasă un buton pentru a alege operația pe care dorește să o efectueze, programul efectuează operația cerută și o afișează în cadrul interfeței grafice pentru utilizator.
- Programul trebuie să poată efectua corect operațiile matematice pe polinoame: adunare, scădere, înmulțire, împărțire, derivare, integrare.

## Proiectare

Pentru proiectarea calculatorului polinomial am implementat clasele: Monom, Polinom, Model, View, Controller, MainClass și ModelTest.

În clasa Monom: un monom este un termen al unui polinom, care este produsul dintre un coeficient și o variabilă  $x$  la o putere. Am implementat această clasă pentru a putea crea un polinom (o sumă de mai multe monoame).

În clasa Polinom: cum un polinom este o sumă de monoame, un polinom este de fapt o listă de monoame. În această clasă am implementat o metodă care să transforme un șir de caractere într-un polinom (pentru a putea citi polinoamele introduse de la tastatură), o metodă

care transformă un polinom în șir de caractere (pentru a afișa polinoamele) și o funcție care îmbină două monoame citite care au aceeași putere.

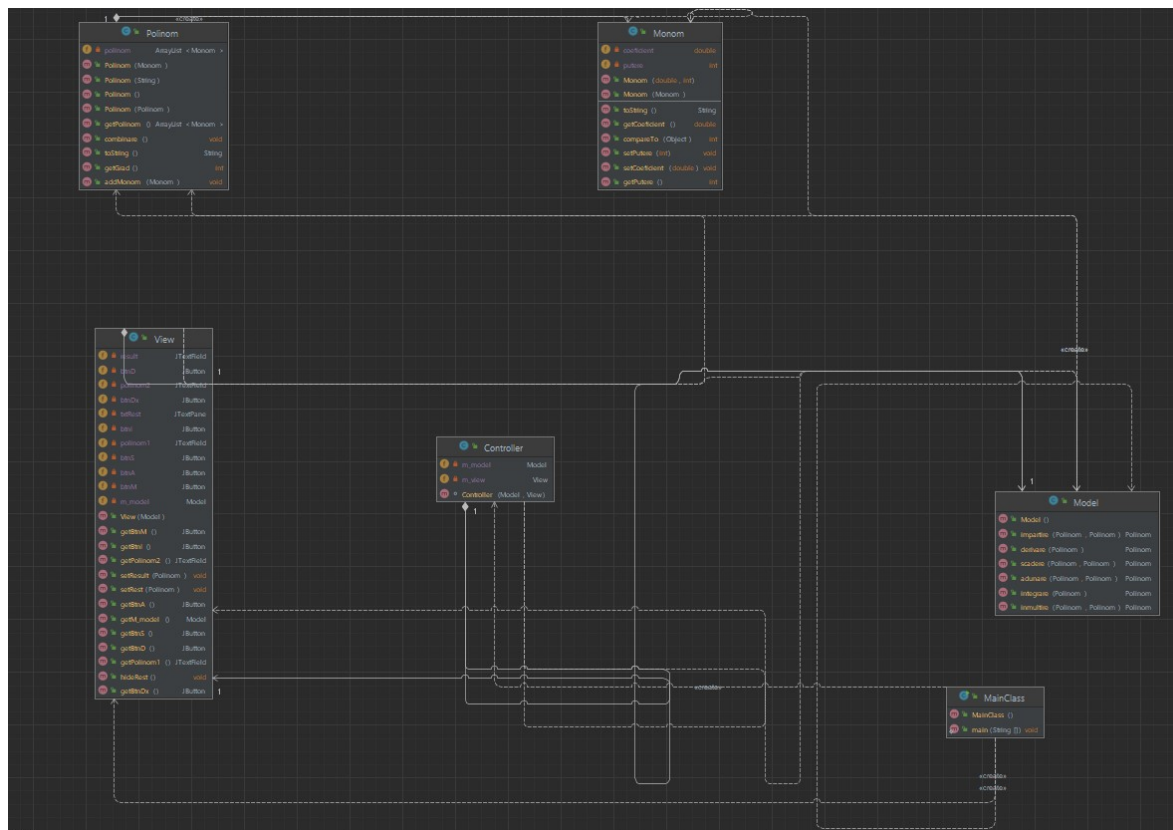
În clasa Model am implementat operațiile matematice cu polinoame: adunare, scădere, înmulțire, împărțire, derivare și integrare.

Clasa Controller procesează solicitările utilizatorului. Controlerul înregistrează ascultătorii care sunt apelați atunci când View detectează o interacțiune a utilizatorului. Aici am implementat ActionListenerii pentru butoane.

În clasa View am implementat interfața grafică. View primește comenzile de la Controller și afișează răspunsul pentru utilizator.

În clasa ModelTest am testat operațiile cu JUnit.

## Diagrama UML



# Implementare

Clasele implementate sunt: MainClass, Monom, Polinom, Model, View, Controller, ModelTest.

Clasa Monom are două câmpuri, coeficient –coeficientul monomului și putere –puterea monomului. Am implementat 2 constructori: unul primește coeficientul și puterea, iar celălalt primește un monom dat, pentru a nu suprascrie (am avut nevoie la adunare). Dacă puterea este negativă, atunci se va arunca o excepție. Am implementat metoda toString pentru a afișa monomul: am luat cazurile când coeficientul este diferit de 0, când coeficientul este -1 sau 1 și puterea este 0, iar când coeficientul este diferit de 1, dacă acesta este real, se scrie doar cu 2 zecimale. La final am implementat metoda compareTo, pentru a putea sorta polinomul începând cu puterea cea mai mare.

Clasa Polinom are un câmp, lista de monoame. Am implementat 3 constructori: unul care primește un polinom, unul care primește un monom și îl adaugă în polinom și unul care primește un string și îl face monom. În acest constructor, am despărțit stringul primit ca parametru după "+", ia fiecare termen și verifică dacă conține "x" (dacă nu, puterea este 0 și coeficientul este tot termenul, altfel dacă "x" este primul caracter atunci coeficientul este 1, dacă "-" este primul caracter și "x" al doilea atunci coeficientul este -1, altfel coeficientul este de la începutul stringului până la "x"), iar dacă stringul conține "^" atunci puterea este de la acest caracter până la finalul stringului, altfel puterea este 1. La final, adăugăm monomul citit la polinom. Apoi am implementat o metodă toString care afișează polinomul ca string. Am implementat și o metodă care primește un monom și îl adaugă la polinom și o metodă care returnează gradul polinomului (pentru împărțire). La final, am implementat o metodă combinare: dacă se adaugă 2 monoame care au același grad, atunci se face suma coeficienților (pentru a avea doar un monom cu acea

putere). La final, sortăm polinomul să fie afișat începând cu puterea cea mai mare.

În clasa Model am implementat operațiile matematice pentru polinoame. La adunare, am adăugat în polinomul rezultat cele 2 polinoame de adunat și am aplicat metoda combinare din clasa Polinom. La scădere am folosit operația de adunare și de înmulțire implementate: am făcut adunarea dintre primul polinom și al doilea înmulțit cu -1. La înmulțire am luat fiecare monom din ambele polinoame, le am înmulțit coeficienții și puterile și am adăugat monomul final la polinomul rezultat. La împărțire am pus în polinomul rest primul polinom, iar cât timp gradul restului este mai mare decât gradul împărțitorului, se face un monom nou care împarte coeficientul primului polinom la al doilea și scade din puterea primului polinom puterea celui alt polinom, iar acest monom se adaugă la rezultat (care va fi câtul împărțirii), iar din rest se scade împărțitorul înmulțit cu monomul creat. La derivare am calculat coeficientul și puterea monoamelor pe rând (coeficientul se înmulțește cu puterea, iar puterea se scade) și se adaugă monoamele la rezultat. La integrare am calculat coeficientul și puterea monoamelor pe rând (coeficientul se împarte la putere, iar puterea crește cu 1) și se adaugă monoamele la rezultat. La fiecare operație se folosește la final metoda de combinare din clasa Polinom pentru a aranja polinomul.

În clasa View am implementat interfața grafică. Am adăugat butoanele, text fieldurile unde se introduc polinoamele de către utilizator și unul unde se afișează rezultatul operației alese de către utilizator.

Clasa Controller face legătura între interfața grafică (View) și logica din spatele calculelor (Model). În clasa Controller am implementat ActionListener pentru toate butoanele din View. În fiecare ActionListener se verifică dacă polinoamele introduse sunt valide (altfel se afișează un mesaj de eroare), se citesc polinoamele ( la derivare și integrare se citește doar un

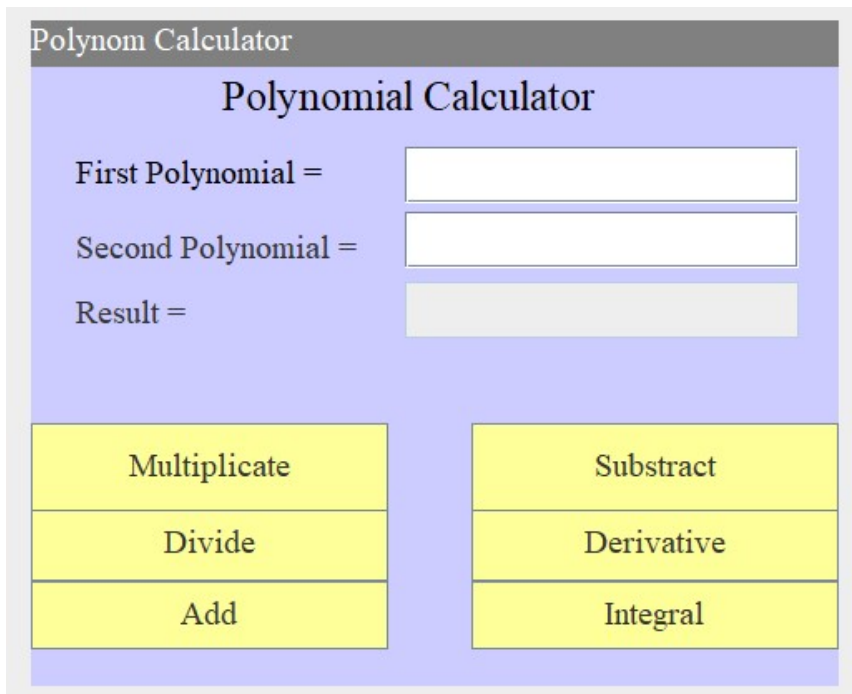
polinom), iar dacă cele două sunt valide, se face operație de pe butonul apăsător (adunare, scădere, înmulțire, împărțire, derivare, integrare). La împărțire se afișează și un rest, iar la celelalte operații acesta este ascuns.

## Rezultate

## Testare

Pentru testare am folosit JUnit. În clasa ModelTest am testat toate operațiile matematice cu Polinoame folosind mai multe exemple.

În final am obținut un calculator funcțional pentru polinoame.



The image shows a Java Swing window titled "Polynom Calculator". The window has a light blue background. At the top, there is a title bar with the text "Polynom Calculator". Below the title bar, the text "Polynomial Calculator" is displayed in a larger font. There are three input fields: "First Polynomial =", "Second Polynomial =", and "Result =". The "Result =" field is highlighted in light gray. Below the input fields, there are six buttons arranged in two columns. The left column contains "Multiply", "Divide", and "Add". The right column contains "Subtract", "Derivative", and "Integral". All buttons are yellow with black text.

Polinoamele se introduc în primele 2 spații goale, rezultatul este afișat în al treilea spațiu gol, lângă "Result". Mai jos sunt butoanele pentru cele 6 operații. Dacă polinoamele nu sunt



valide, apare un mesaj de eroare. Când se face împărțirea polinoamelor, sub "Result" apare un câmp "Rest" și restul împărțirii lor.

## Concluzii

În concluzie, această temă a fost un exercițiu excelent pentru familiarizarea cu limbajul de programare Java, pentru a învăța lucrul cu interfața grafică, pentru înțelegerea modelului MVC și pentru a aprofunda cunoștințele despre limbajul Java: clasele, metodele, constructorii, convertiri din string, lucrul cu liste.

## Bibliografie

- <https://docs.oracle.com/javase/tutorial/uiswing/>
- <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>
- <https://www.baeldung.com/java-tutorial>
- <https://google.github.io/styleguide/javaguide.html>
- <https://www.tutorialspoint.com/questions/category/Java>
- <https://en.wikipedia.org/wiki/Polynomial>