DOCUMENTATIE

TEMA 1

NUME STUDENT: SAND ELENA-ANDREEA

GRUPA: 30223

# 

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Obiectivul acestei teme este de a implementa un calculator polinomial cu o interfata grafica care permite introducerea a doua polinoame de o singura variabila, selectarea unei operatii de catre utilizator si afisarea rezultatului.

Operatiile sunt: adunarea, scaderea, inmultirea, impartire a doua polinoame, integrarea si derivarea unui polinom.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

**Cerintele functionale ale acestui proiect sunt:**

* Introducerea polinoamelor in interfata de catre utilizator, fie prin scrierea unui string de la tastatura, fie prin apasarea butoanelor din interfata
* Alegerea operatiei dorite de catre utilizator
* Realizarea operatiei cerute
* Afisarea rezultatului in interfata

**Cerintele non-functionale ale acestui proiect:**

* Calculatorul polinomial trebuie sa fie intuitiv si usor de folosit pentru orice utilizator
* Interfata grafica sa fie prietenoasa

Pentru realizarea acestui proiect este necesara cunoasterea conceptelor OOP si stapanirea unor cunostiinte matematice pentru realizarea operatiilor.

Clasele de baza a acestui proiect sunt : Monomial si Polinom in care contin carcateristicile esentiale ale unui monom sau polinom. Metodele din Clasa Operation returneaza un obiect de tip Polinom si primesc ca parametrii obiecte de tip Polinom. Un polinom contine o lista de monoame (ArrayList <Monomial> ), iar un monom are ca atribute gradul si coeficientul. Gradul este de tipul int, iar coeficientul este de tip double pentru o precizie mai buna a rezultatului (in cazul operatiilor de impartire sau integrare). Polinomul este introdus de catre utilizator sub forma unui sir care va fi transformat intr-un sir de monoame. De exemplu, daca este introdus sirul “3x^2 + 2x^1 + 5x^0”, lista de monoame va fi [3x^2, 2x^1, 5x^0]. Coeficientul poate sa fie si de forma double, nu este afectata transformarea polinomului, insa daca se introduce dupa x si ‘^’ sau apar litere , aplicatia o sa afiseze o eroare despre introducerea incorecta a polinomului. Utilizatorul poate alege din interfata operatia dorita. Dupa selectarea operatiei, se presupune ca primul polinom este cel folosit la operatia de integrare sau derivare. Daca la operatia de adunare se introduce un singur polinom in interfata, se presupune ca al doilea este 0. La operatia de impartire, se verifica gradul celor doua polinoame si in functie de acesta se alege deimpartitul si impartitoul (deimpartitul= polinomul cu gradul cel mai mare, impartitorul= polinomul cu gradul mai mic). Rezultatul operatiei se afiseaza sub forma unui String prin apelul metodei toString() din clasa Polinom.

**Cazurile de utilizare:**

Adunarea, scaderea, inmultirea, impartirea a doua polinoame, derivarea, integrarea unui polinom

Actor principal: utilizatorul

Scenariul principal de success: -utilizatorul insereaza doua polinoame in interfata grafica

-utilizatorul selecteaza operatia “add”

-calculatorul polinomial realizeaza operatia de adunare a doua polinoame si

afiseaza rezultatul

Scenariul alternativ: inserarea gresita a polinoamelor ( de exemplu: cu doua sau mai multe variabile)

# Proiectare

Aplicatia contine 6 clase : Monomial, Polinomial, Operation (modelul aplicatiei), CalculatorView (vizualizarea – reprezentarea vizuala a datelor din model) , CalculatorController (controlor- preia intrarea de la utilizator si o transpune in schimbari in model) , Main.

Clasele Polinom si Monomial sunt folosite in principal pentru pastrarea datelor intr-o forma potrivita si pentru prelucrarea acsetora,, ele fiind usor de accesat. Operatiile sunt descrise in clasa Operation, iar main-ul aplicatiei se afla in clasa Main.

Pentru crearea interfetei grafice am folosit arhitectura Swing care isi are radacinile in arhitectura model – view – controller (MVC). Aceasta arhitectura cere ca aplicatia vizuala sa fie divizata in trei parti separate:

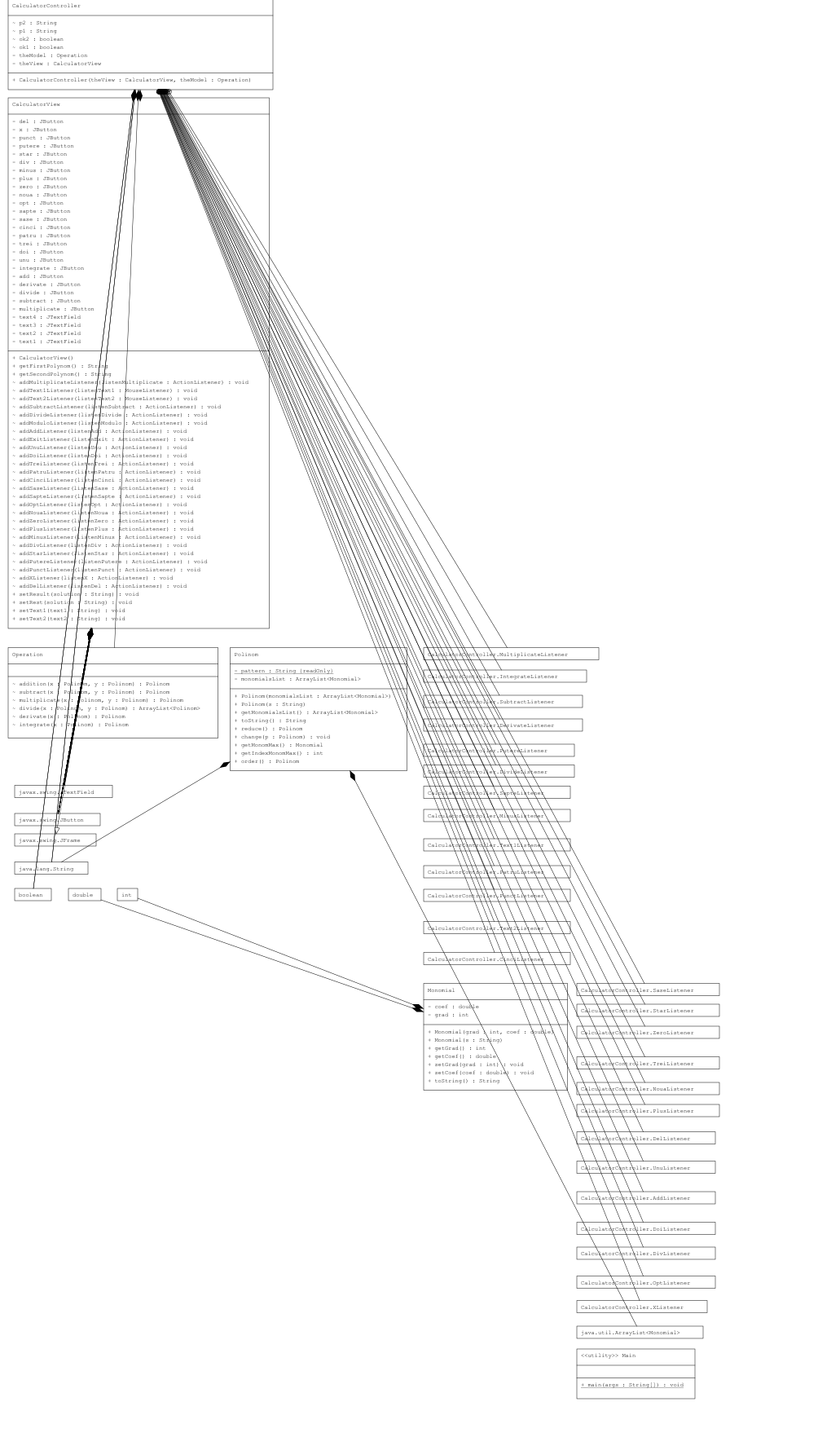
* Un model – reprezinta intern datele aplicatiei
* O vizualizare (view) – reprezentarea vizuala a datelor respective
* Un controlor (controller) – care preia intrarea de la utilizator si o transpune in schimbari in model

In acest proiect, crearea interfetei grafice se face in clasa CalculatorView, iar interactiunea dintre utilizator si interfata este descrisa in clasa CalculatorController.

Interfata contine doua campuri pentru text (JtextField) care permit introducerea polinoamelor, inca doua folosite pentru afisarea rezultatului. Interfata mai contine 24 butoane (6 pentru selectarea operatiei si 18 pentru caracterele necesare introducerii polinoamelor).

La fiecare apasare de buton se citesc polinoamele introduse de utilizator, astfel, orice schimbare pe care utilizatorul o face asupra polinoamelor, este perceputa in momentul in care utilzatorul alege operatia. Polinomul poate fi introdus de la tastatura sau prin apasarea butoanelor din interfata.

Diagrama de clase:



# Implementare

**Clasa Monomial** contine doua campuri: coeficientul de tip private double si gradul (exponentul) de tip private int. Sunt implementati doi constructori: public Monomial(int grad, double coef) si public Monomial(String s) care extrage coeficientul si gradul cu ajutorul metodei s.split((“[+-]?(x\\^)”)). De asemenea, este implementata si metoda toString( ) care transforma monomul intr-un String. In clasa mai sunt implementate metodele de setter si getter pentru atributele private double si coef.

In **clasa Polinom** exista doi constructori : unul care primeste ca si parametru o lista de monoame si unul care primeste ca parametru un String s. Al doilea constructor face lagatura dintre interfata si restul programului deoarece aceasta metoda transforma polinomul din sir in lista de monoame pentru a se putea efectua operatiile. Sirul este impartit in subsiruri care reprezinta un monom, iar din aceste subsiruri se construiesc monoamele prin transformarea din String in Double si Int a coeficientului si exponentului. Este suprascrisa metoda toString care se foloseste pentru afisarea rezultatului,iar metoda reduce() din aceasta clasa are rolul de a ordona polinomul, astfel incat sa nu exista doua monoame diferite cu acelasi exponent intr-un polinom.Aceasta metoda parcurge polinomul care trebuie ordonat si realizeaza suma dintre doua monoame care au acelasi exponent, polinomul ordonat se salveaza intr-un alt polinom.Metoda change(Polinom p) este o metoda care suprascrie polinomul referinta cu un polinom trasnmis ca si parametru. Exista si metoda getMonomMax care returneaza monomul care are gradul maxim. Acesta metoda se foloseste la algoritmul de impartire si a fost implementata printr-o parcurgere simpla a polinomului si extragerea maximului. Metoda getIndexMonomMax( ) returneaza indexul din lista de monoame a monomului cu grad maxim. Aceasta este folosita in metoda order() care ordoneaza polinomul dupa grad.

**Clasa Operation** contine toate operatiile care se realizeaza asupra polinoamelor, la selectarea utilizatorului din interfata grafica.

Metoda Polinom addition (Polinom x, Polinom y) returneaza un alt polinom (rez) care reprezinta suma dintre x si y. Am folosit un vector verifj[] pentru a verifica daca toate monoamele din polinomul y au fost introduse in rez. Cu doua for-uri parcurg lista de monoame a celor doua polinoame, iar monoamele ede acelasi grad se aduna. Cu o variabila ok verific pentru fiecare monom din polinomul x daca a fost adaugat la rez.

Metoda Polinom subtract( Polinom x, Polinom y ) este similara metodei add, doar ca intre monoamele de acelasi grad se face diferenta, iar pentru monoamele din y care nu au fost adaugate in rez, se modifica semnul coeficientului inainte sa fie adaugate.

Metoda Polinom multiplicate( Polinom x, Polinom y ) returneaza un alt polinom ( rez ) care reprezinta rezultatul inmultirii dintre x si y. Se folosesc doua for-uri in care fiecare monom din polinomul x este inmultit cu fiecare monom din polinomul y, iar monomul rezultat se adauga in lista polinomului rez. Dupa parcurgerea celor doua for-uri se apeleaza metodele reduce() si order() pentru rez pentru a ordona polinomul rezultat.

Metoda Polinom divide( Polinom x, Polinom y ) returneaza un array de polinoame. Primul element din lista este catul, iar al doilea este restul. La inceputul metodei se alege deimpartitul (polinomul cu grad mai mare) si impartitorul ( polinomul cu grad mai mic ). In variabila rest se stocheaza deimpatitul,, in m1 monomul maxim din polinomul rest, iar in m2 monomul maxim din polinomul imp. Cat timp variabila rest este diferita de 0 si gradul impartitorului are gradul mai mic decat al deimpartitului, se imparte monomul maxim din deimpartit cu monomul maxim din impartitor, iar rezultatul (mrez) se adauga in polinomul cat. La fiecare pas, rest <= rest- (mrez\*imp). Dupa ce iesim din bucla, se ordoneaza polinoamele cat si rest si se adauga in array-ul ce urmeaza sa fie returnat.

Metoda Polinom derivate(Polinom x) returneaza un polinom care reprezinta derivata polinomului x. Se parcurge cu un for lista monoamelor si pentru fiecare monom care are gradul mai mare sau egal cu 1, coeficientul primeste coef\*grad, iar gradul se decrementeaza. Monoamele de grad 0, nu se adauga pentru ca derivate lor este 0.

Metoda Polinom integrate(Polinom x) returneaza un polinom care reprezinta integrate polinomului x. Cu un for se parcurge lista monoamelor din polinomul x si pentru fiecare monom, coeficientul primeste coef/(grad+1), iar gradul se incrementeaza cu 1.

**Clasa CalculatorView** contine elementele din interfata grafica. Interfata contine 4 JPanel-uri, dintre care unul este cel mare si le contine pe celelalte 3. Primul panel secundar contine un JLabel in care este scris titlul “Polynomial Calculator” si este pozitionat in nordul panel-ului principal, pozitia fiind specificata astfel: this.add(panel\_0, BorderLayout.NORTH). Al doilea panel secundar este construit cu GridLayout(7,2) si contine in primele patru linii JLabel-urile si JTextField-urile pentru introducere celor doua polinoame si pentru afisarea rezultatului si a restului (in cazul impartirii). Pe urmatoarele 3 linii sunt pozitionate cele 6 butoane care permit alegerea operatiei dorite. Acest panel este pozitionat in centrul panel-ului principal. Ultimul panel este construit cu GridLayout(3,6), in care sunt introduse butoanele cu caracterele specific pentru introducerea polinomelor (1,2,3,4,5,6,7,8,9,0,+,-,/,\*,^,^,.,x,del). Acest panel este pozitionat in sudul panel-ului principal. Tot in aceasta clasa sunt implementate metodele getFirstPolynom() si getSecondPolynom() care returneaza polinoamele introduse de Utilizator si metodele setResult(), setRest(), setText1(), setText2() care afiseaza rezultatele in interfata. De asemenea, sunt implementate si metodele care adauga listeneri butoanelor din interfata.

**Clasa CalculatorController** face legatura intre clasa CalculatorView si clasa Operation (model). Se implementeaza metodele MouseListener sau ActionPerformed pentru fiecare buton sau pentru cele doua text field-uri in care se introduc polinoamele. Pentru introducerea celor doua polinoame de la butoanele din interfata a fost necesar sa implementez metoda MouseListener pentru cele doua text field-uri. Daca s-a facut click in text field-ul pentru primul polinom, am setat variabilele booleene ok1=true si ok2=false (pentru a stii in ce text field introduce), iar daca s-a facut click in text field-ul pentru polinomul 2, am setat variabilele booleene ok1=false, ok2=true. Dupa care, pentru fiecare buton specific caracterelor folosite pentru introducerea polinoamelor, in metoda ActionPerformed verific ce variabila Boolean este true pentru a stii in ce text field actualizez textul. Textul se actualizaeaza folosind doua string-uri p1 si p2 declarate in aceasta clasa, prin concatenarea lor cu caracterul selectat de Utilizator : p1=p1+”caracterul”; theView.setText1(p1). Butoanele pe care utilizatorul le foloseste pentru a alege ce operatie sa se realizeze asupra polinoamelor introduse au denumire semnificative. La fiecare apasare de buton, se citesc inca o data polinoamele , astfel , fiecare modificare facuta de utilizator este perceputa la fiecare apasare de buton. In metodele actionPerformed specific butoanelor pentru selectarea operatiei, am folosit doua string-uri in care memorez polinoamele introduse si cu ajutorul carora creez polinoamele trimise ca parametrii metodei selectate (theView.setResult(theModel.operatie(p1,p2).toString()). Pentru fiecare buton am suprascris metoda actionPerformed astfel incat fiecare buton sa controleze operatia care se efectueaza.

# Rezultate

Testarea s-a realizat cu Junit in clasa OperationTest.S-au construit polinoame si s-au apelat metodele din clasa Operation, cu ajutorul unui obiect m de tip Operation . Astfel s-au verificat toate operatiile care se pot realiza din interfata. Rezultatele apar in urmatorul tabel:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ce se testeaza** | **Date de Intrare** | **Rezultat asteptat** | **Rezultat efectiv** | **Pass/Fail** |
| Suma  addition(p1,p2) | P1= 1x^7-1x^8+9x^1-14x^5-1x^3+12x^6+6x^1-8x^0  P2=1x^5+5x^4+3x^5+3x^3-1x^8-6x^1+8x^0 | P= -2.0x^8+1.0x^7+12.0x^6-10.0x^5+5.0x^4+2.0x^3+9.0x^1 | P= -2.0x^8+1.0x^7+12.0x^6-10.0x^5+5.0x^4+2.0x^3+9.0x^1 | pass |
| Diferenta  subtract(p1,p2) | P1= 1x^7-1x^8+9x^1-14x^5-1x^3+12x^6+6x^1-8x^0  P2=1x^5+5x^4+3x^5+3x^3-1x^8-6x^1+8x^0 | P= 1.0x^7+12.0x^6-15.0x^5-3.0x^5-5.0x^4-4.0x^3+15.0x^1+6.0x^1-16.0x^0 | P= 1.0x^7+12.0x^6-15.0x^5-3.0x^5-5.0x^4-4.0x^3+15.0x^1+6.0x^1-16.0x^0 | pass |
| Inmultirea  multiplicate(p1,p2) | P1= 3x^2-1x^1+1x^0  P2= 1x^1-2x^0 | P= 3.0x^3-7.0x^2+3.0x^1-2.0x^0 | P= 3.0x^3-7.0x^2+3.0x^1-2.0x^0 | pass |
| Impartirea  divide(p1,p2) | P1= 1x^3-2x^2+6x^1-5x^0  P2= 1x^2-1x^0 | Rez= 1.0x^1-2.0x^0  Rest= 7.0x^1-7.0x^0 | Rez=1.0x^1-2.0x^0  Rest=7.0x^1-7.0x^0 | pass |
| Derivarea  derivate(p1) | P1=1x^3-2x^2+6x^1-5x^0 | Rez=3.0x^2-4.0x^1+6.0x^0 | Rez=3.0x^2-4.0x^1+6.0x^0 | pass |
| Integrarea  integrate(p1) | P1= 1x^3+4x^2+5x^0 | Rez= 0.25x^4+1.33x^3+5.0x^1 | Rez= 0.25x^4+1.33x^3+5.0x^1 | pass |

# Concluzii

Rezultatul acestui proiect este o aplicatie functionala de efectuat operatii pe polinoame. S-au fixat prin realizarea acestui proiect , cunostiintele depsre cum se construieste o interfata, cum se leaga aceasta de program astfel incat sa functioneze asa cum se asteapta utilizatorul , dar si conceptele OOP.Am aprofundat ArrayList-urile si am folosit pentru prima data un regex pentru a desparti un sir in subsiruri.

In viitor, se poate construi o interfata mai prietenoasa.sau se pot adauga mai multe operatii care sa se efectueze asupra polinoamelor.

# Bibliografie

* <https://docs.oracle.com/javase/tutorial/uiswing/>
* <https://www.vogella.com/tutorials/JUnit/article.html>
* https://www.baeldung.com/junit-5