DOCUMENTATIE

TEMA *1 – CALCULATOR POLINOAME*

NUME STUDENT: Dragus Andreea

GRUPA: 30226

# 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 principal al temei se refera la realizarea operatiilor dintre polinoame prin intermediul unei Graphic User Interface (GUI).

Obiectivul secundar al temei este reprezentat de pasii care sunt executati in vederea indeplinirii obiectivului principal. Operatiile care pot fi realizate prin intermediul acestui proiect sunt:

* Adunarea polinoamelor
* Scaderea polinoamelor
* Inmultirea polinoamelor
* Impartirea polinoamelor
* Derivarea polinoamelor
* Integrarea polinoamelor

Atat modalitatea de implementarea cat si logica folosita vor fi detaliate in capitolul „4.Implementare” .

# Analiza problemei, modelare, scenarii, cazuri de utilizare

In cadrul proiectului, utilizatorul interactioneaza cu o GUI, prin intermediul careia acesta poate sa introduca de la tastatura 2 polinoame asupa carora se vor efectua operatiile. De asemenea, acesta poate sa selecteze si operatia care se va executa prin intermediul unor butoane. Rezultatul va fi afisat in cadrul unui nou textField numit „Rezultat” (+”Rezultat2” in cazul impartirii).

Use-case-urile acestui proiect sunt:

* Adunarea polinoamelor: se aduna cele 2 polinoame introduse de utilizator iar rezultatul adunarii este afisat in textFiledul „Result”
* Scaderea polinoamelor: se scad cele 2 polinoame introduse de utilizator iar rezultatul scaderii este afisat in textFiledul „Result”
* Inmultirea polinoamelor: se inmultesc cele 2 polinoame introduse de utilizator iar rezultatul inmultirii este afisat in textFiledul „Result”
* Impartirea polinoamelor: se impart cele 2 polinoame introduse de utilizator iar rezultatul impartirii este afisat astfel: in textFiledul „Result” se afiseaza catul impartirii, iar in textFiledul „Result2” se afiseaza restul (cele 2 sunt afisate sub forma unor polinoame)
* Derivarea polinoamelor: polinomul introdus in textFiled-ul „First Polynomial” va fi cel asupra caruia se va efectua operatia de derivare, iar rezultatul acesteia va afisat sub forma polinomiala in textFiledul „Result”
* Integrarea polinoamelor: polinomul introdus in textFiled-ul „First Polynomial” va fi cel asupra caruia se va efectua operatia de integrare, iar rezultatul acesteia va afisat sub forma polinomiala in textFiledul „Result”

**

Polynomial Calculator

# 

A screenshot of a computer

Description automatically generated

User

# Proiectare

Proiectarea OOP a aplicatiei a fost realizata in concordanta cu principiile formatului arhitectural „Model – View – Controller” (MVC). Pentru a realiza acest lucru au fost implementate 3 pachete: Model, View, Controller.

Pachetul View are rolul de a realiza GUI prin intermediul careia utilizatorul poate beneficia de o experienta placuta vizual atunci cand doreste sa utilizeze programul implementat. Prin urmare, acesta contine clasa denumita „GUI”.

Pachetul Model are rolul de a implementa clasele reprezentative pentru rezolvarea acestei proleme: Monomial, Polynomial, Operation.

Pachetul Controller are rolul de a face legatura dintre Model si View. Astfel, in cadrul acestui pachet sunt regasite clasele: ButtonListener, Regex, Main.

Utilizarea acestui model arhitectural reprezinta un beneficiu major in ceea ce priveste buna organizare a codului, intrucat ajuta la compartimentarea eficienta a claselor.

Prin intermediul diagramei UML se pot observa modul in care clasele sunt concepute dar si modul in care clasele sunt dependente intre ele. Diagrama UML repezinta modul prin care conceptele folosite in acest proiect pot fi intelese cu usurinta de catre alti programatori in cazul in care se doreste dezvoltarea ulterioara a proiectului.

Graphical user interface

Description automatically generated with medium confidence

# Implementare

* 1. Pachetul Model:
* CLASA MONOMIAL:
* Aceasta clasa implementeaza interfata „Comparable” utilizata atunci cand se doreste ordonarea monoamelor
* Monomial are 2 variabile instanta: „private double coefficient” si „private int degree” , reprezentand coeficientul si puterea monomului. Acestea au ca modificator de acces „private” pentru a utiliza concentul de incapsulare a datelor. Cele doua campuri vor fi accesibile prin intermediul metodelor de get si set care vor fi explicate in detaliu in cele ce urmeaza. Prima variabila de instanta este de tip
* Clasa in discutie contine 2 constructori si 7 metode publice. Primul constructor primeste ca si parametrii doi intregi reprezentand coeficientul si puterea unui monom si are rolul de initializare a variabilelor instanta. Cel de al doilea constructor nu are parametrii si este utilizat in contextele in care initializarea variabilelor instanta se doreste a se efectua ulterior declararii obiectului de tip monom, prin intermediul metodelor get si set
* Cele 7 metode sunt:
* public double getCoefficient() : are rolul de a returna un numar real de tip double reprezentand coeficientul monomului. Este utila in contextul in care dorim sa accesam acest camp in interiorul unei clase in care nu este vizibil din cauza modificatorului de acces private.
* public int getDegree () : are rolul de a returna un numar intreg de tip int reprezentand puterea monomului. Este utila in contextul in care dorim sa accesam acest camp in interiorul unei clase in care nu este vizibil din cauza modificatorului de acces private.
* public void setCoefficient(double coefficient) : are rolul de a actualiza variabila instanta coefficient cu valoarea transmisa ca parametru
* public void setDegree(int degree): are rolul de a actualiza variabila instanta degree cu valoarea transmisa ca parametru
* public String toString(): aceasta metoda este suprascrisa cu rolul de a fi folosita in contextul afisarii monoamelor, astfel incat sa se realizeze o afisare sugestiva, care contine toate campurile unui monom si nu doar referinta obiectului cum s-ar intampla in cazul System.out.println(monom);
* public int CompareTo(Monomial o): aceasta metoda necesita suprascriere deoarece clasa Monomial implementeaza interfata Comparable; aceasta metoda este utila in contextul ordonarii monoamelor, astfel incat aceasta sa se realizeze in functie de puterea lor
* public boolean equals(Object obj): aceasta metoda este suprascrisa cu scopul verificarii egalitatii dintre 2 obiecte de tip Monomial, astfel incat egalitatea sa fie verificata pentru fiecare variabila instanta a obiectelor si nu din prisma referintelor acestora
* CLASA POLYNOMIAL:
* Aceasta clasa are ca si variabila instanta un ArrayList de obiecte Monomial
* Clasa in discutie contine 2 constructori si 4 metode publice. Primul constructor primeste ca si parametru un ArrayList de Monomial si are rolul de initializare a variabilei instanta. Cel de al doilea constructor nu are parametrii si este utilizat in contextele in care initializarea variabilei instanta se doreste a se efectua ulterior declararii obiectului de tip polynomial, prin intermediul metodelor get si set.
* Cele 4 metode sunt:
* public ArrayList<Monomial> getMonomialsList(): returneaza continutul variabilei instanta
* public void setMonomialsList(ArrayList<Monomial> monomialsList): actualizeaza continutul variaiblei instanta
* public String toString(): aceasta metoda este suprascrisa cu rolul de a afisa ArrayList-ul de Monomial sub forma unui polinom. Aceasta metoda construieste rezultatul intr-un string initial null, prin concatenarea fiecarui element din ArrayList
* public boolean equals(Object obj): aceasta metoda este suprascrisa cu rolul verificarii egalitatii dintre 2 obiecte de tip Polynomial. Aceasta verifica egalitatea ArrayList-urilor de Monomial ale celor doua obiecte. In cazul in care coincid, atunci polinoamele sunt considerate egale.
* CLASA OPERATION:
* Aceasta clasa contine 6 metode statice, reprezentand fiecare operatie realizabila in cazul polinoamelor
* Cele 6 metode sunt:
* public static Polynomial Addition(Polynomial p1, Polynomial p2): Aceasta metoda are drept scop realizarea operatiei de adunare dintre doua polinoame primite ca parametru si returnarea unui polinom rezultat. Idee: Se incepe parcurgerea celor doua polinoame de pe pozitiile i=0 si j=0. Daca avem la pozitia i un monom care are puterea mai mare decat monomul de pe pozitia j din acelasi polinom, atunci polinomul de pe pozitia i va fi adaugat ArrayList-ului de monoame a polinomului rezultat, incrementandu-se variabila i. Se aplica acelasi procedeu si in cazul invers. Daca cele doua monoame de pe pozitiile i si j din cele 2 polinoame au aceeasi putere, atunci se va creea un obiect Monomial suplimentar care va avea puterea (degree) egala cu puterea comuna si coeficientul egal cu suma coeficientilor celor 2 monoame. Acest monom va fi adaugat in ArrayList-ul polinomului rezultat iar i si j se incrementeaza. Acest procedeu se repeta pana cand am terminat de parcurs unul din polinoame. In acest punct, vom verifica daca au mai ramas elemente in vreunul din polinoame, iar in caz afirmativ acestea vor fi adaugate la polinomul rezultat. La final, vom ordona ArrayList-ul rezultat si vom returna polinomul.
* public static Polynomial Substraction(Polynomial p1, Polynomial p2): Aceasta metoda are drept scop realizarea operatiei de scadere dintre doua polinoame primite ca parametru si returnarea unui polinom rezultat. Idee: Se incepe parcurgerea celor doua polinoame de pe pozitiile i=0 si j=0. Daca avem la pozitia i un monom care are puterea mai mare decat monomul de pe pozitia j din acelasi polinom, atunci polinomul de pe pozitia i va fi adaugat ArrayList-ului de monoame a polinomului rezultat, incrementandu-se variabila i. In cazul invers, se va creea un nou monom care are aceeasi putere ca si cel initial dar coeficientul egal cu -1\*coeficientul initial care va fi adaugat la ArrayList-ul rezultat, dupa care se incrementeaza j-ul. Daca cele doua monoame de pe pozitiile i si j din cele 2 polinoame au aceeasi putere, atunci se va creea un obiect Monomial suplimentar care va avea puterea (degree) egala cu puterea comuna si coeficientul egal cu diferenta coeficientilor celor 2 monoame. Acest monom va fi adaugat in ArrayList-ul polinomului rezultat iar i si j se incrementeaza. Acest procedeu se repeta pana cand am terminat de parcurs unul din polinoame. In acest punct, vom verifica daca au mai ramas elemente in vreunul din polinoame, iar in caz afirmativ acestea vor fi adaugate la polinomul rezultat (cu coeficientul modificat in -coeficient in cazul monoamelor din cel de al doilea polinom). La final, vom ordona ArrayList-ul rezultat si vom returna polinomul.
* public static Polynomial Multiplication(Polynomial p1, Polynomial p2): In cadrum acestei metode sunt parcurse ArrayList-urile celor doua monoame, inmultindu-se componentele lor intre ele. La fiecare pas se creeaza un nou monom care are ca si coeficient produsul coeficientilor celor 2 monoame si ca si putere suma puterilor lor. Acest nou monom este transformat intr-un polinom care se aduna (folosind metoda Addition mentionata mai sus) la polinomul rezultat.
* public static ArrayList <Polynomial> Division ( Polynomial p1, Polynomial p2): Se verifica daca polinomul p2 nu este null. Initial, restul va fi egal cu polinomul p1. Atata timp cat mai exista elemente in polinomul rest si acesta are o putere mai mare decat p2: cautam cea mai mare putere din r care are coeficientul diferit de 0. Odata gasit, vom initializa un nou monom t care va avea puterea egala cu a acestui polinom de putere maximma din r – puterea maxima din polinomul p2 si coeficientul egal cu raportul coeficientilor celor doi.Acest monom se retine in un nou polinom numit aux. In polinomul q, se retine catul impartirii, la care se adauga la fiecare pas polinomul aux ( prin intermediul operatiei Addition). In polinomul r se retine restul impartirii, obtinut prin :r= Substraction(r, Multiplication(aux,p2)). La final, se returneaza un array de polinoame, avand pe prima pozitie catul si pe a doua restul impartirii
* public static Polynomial Derivation(Polynomial p): Se parcurge lista de monoame a polinomului transmis drept parametru si in acelasi timp se construieste un nou polinom rezultat care are o lista de monoame a caror putere este egala cu puterea anterioara -1 iar coeficientul este egal cu coeficientul anterior\*puterea anterioara
* public static Polynomial Integration(Polynomial p): Se parcurge lista de monoame a polinomului transmis drept parametru si in acelasi timp se construieste un nou polinom rezultat care are o lista de monoame a caror putere este egala cu puterea anterioara +1 iar coeficientul este egal cu coeficientul anterior /( puterea anterioara+1)
  1. Pachetul View:
* CLASA GUI:
* are o singura metoda statica startGUI() prin intermediul careia se realizeaza interfata grafica a programului: contine diverse panel-uri, textfield-uri si butoane
  1. Pachetul Controller
* CLASA REGEX:
* are o singura metoda statica getPolynomialFormInput(String textInput): are rolul de a identifica elementele caracteristice unui polinom din Stringul introdus ca input in textField-ul din interfata. Aceasta se bazeaza pe utilizarea unei expresii regulare care ajuta la stabilirea elementelor standard ale inputului. Odata interceptate, acestea sunt utilizate pentru initializarea unui obiect de tip polinom care va fi returnat
* CLASA BUTTONLISTENER:
* are ca variabile instanta cele 4 textfield-uri prezente in GUI. In cadrul metodei actionPerformed se verifica numele butonului care a fost apasat. In functie de acest lucru, textField-urile result sau result2 primesc ceea ce returneaza metodele din clasa Operation care primesc ca parametrii ceea ce returneaza getPolynomialFromInput din clasa Regex
* CLASA MAIN:
* are o singura metoda, cea de main, in cadrul careia se apeleaza metoda startGUI din clasa GUI

# Rezultate

Testarea acestui proiect a fost realizata prin intermediul JUnit. Au fost testate toate operatiile implementate pentru polinoamele p1 = 1x^3-2x^2+6x^1-5x^0 si p2 = 1x^2-1x^0.

A fost definita o clasa numita OperationTest care care are o metoda de setUp cu rol de initializare a rezultatelor care ar trebui obtinute in urma efectuarii operatiilor si 6 metode de test pentru fiecare operatie.

# Concluzii

Utilitatea acestui proiect consta in exersarea si aprofundarea tehnicilor de programare orientata spre obiect, fiind totodata un exercitiu de gandire logica. De asemenea, exista posibilitatea de dezvoltare ulterioara a proiectului prin implementarea unor operatii mai complexe precum derivate de ordin superior sau integrale definite dar si prin definirea conceptului de polinom de mai multe variabile.

# Bibliografie

Informatiile necesare implementarii acestui proiect au fost preluate atat din materialele de indrumare prezentate in cadrul laboratorului cat si de pe stackoverflow/geeksforgeeks.