# DOCUMENTATIE

## TEMA 1


NUME STUDENT: VOICU LAURA-LUISA
GRUPA: 30226

# CUPRINS

# 1. Obiectivul temei

***Obiectivul temei il constituie implementarea si proiectarea unui <u>calculator polinomial</u> ce dispune de o interfata grafica dedicata.***



# 2. Analiza problemei, modelare, scenarii, cazuri de utilizare

## Descrierea problemei:

*Un polinom reprezinta o expresie matematica formata dintr-o suma de termeni in care coeficientii sunt numere reale ( sau complexe) iar exponentii sunt numere intregi. Termenii unui polinom poarta numele de monom , ce reprezinta un termen format dintr-o singura variabila ridicata la o putere intreaga,pozitiva.*

## ➢ Cerinte functionale

- ***Adunarea, scaderea, inmultirea polinoamelor**:*
  - *Aplicatia permite utilizatorului sa adune, sa scada sau sa inmulteasca doua polinoame*
- ***Impartirea polinoamelor:***
  - *Aplicatia permite utilizatorului impartirea a doua polinoame, afisant atat rezultatul impartirii, cat si restul*
- ***Derivarea si Integrarea polinoamelor:***
  - *Aplicatia permite derivarea sau integrarea unui polinom*
- ***Buton de informatii pentru utilizarea aplicatiei:***

- *Aplicatia are un buton dedicat informatiilor necesare pentru introducerea corecta a polinomului'*

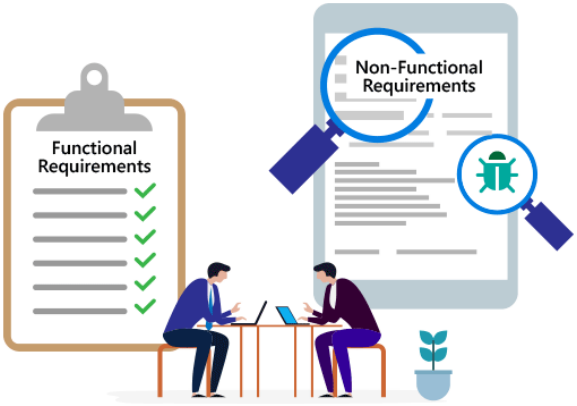
❖ ***Interceptarea erorilor:***

- *Aplicatia permite interceptarea erorilor si afisarea unei noi ferestre cu mesaje sugestive ce descriu cauza erorii.*

## ➢ Cerinte non-functionale

❖ ***Salabilitate** :*

- *Calculatorul trebuie sa poata gestiona polinoame de grade foarte mari. Capacitatea de a manipula astfel de polinoame trebuie sa fie extensibilia si sa poata fi scalata in functie de necesitati*
- *Calculatorul trebuie sa fie rombust si sa poata gestiona erorile intr-un mod adecvat, prin semnalarea utilizatorului sau auto-complete.*




❖ ***Eficienta:***

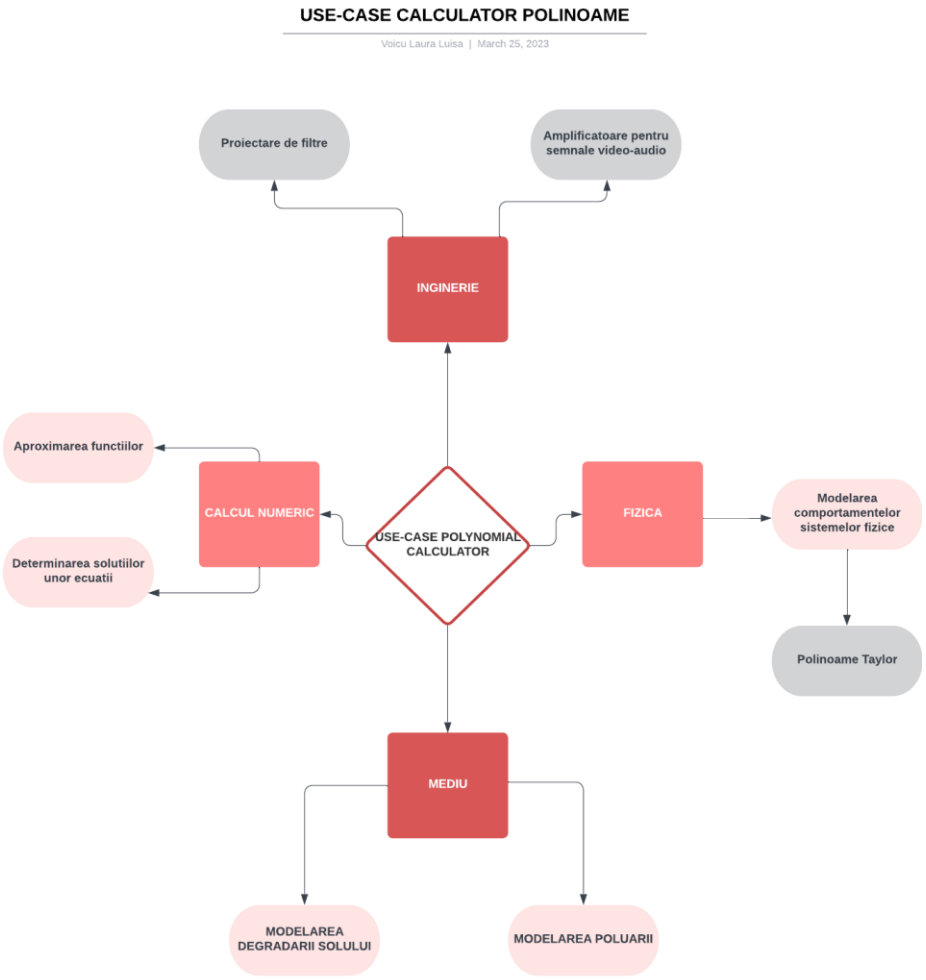
- *Calculatorul trebuie sa fie usor de utilizat si sa ofere o interfata intuitiva pentru utilizator. Functionalitatea ar trebui sa fie disponibila intr-un mod clar si usor de inteles pentru utilizatori, indiferent de nivelurile de experienta.*
- *Calculatorul Polinomial trebuie sa poata procesa rapid operatiile , performanta sa ramanand constanta, indiferent de dimensiunea datelor date de utilizatori*
- *Sugestii sau corectarea automata a operatiilor; o posibila imbunatatire a calculatorului consta in retinerea operatiilor intr-o baza de date ce poate fi accesata atunci cand un utilizator introduce o operatie ce a mai fost calculata in trecut si de alti utilizatori.*

❖ ***Monitorizare:***

- *Calculatorul trebuie sa fie monitorizat pentru detectarea problemelor si prevernirea esecurilor*

- ***Fiabilitate:***
  - *Calculatorul trebuie sa fie stabil si sa ofere rezultate corecte si consistente in timp. Acest lucru se poate realiza prin testarea si verificarea corectitudinii operatiilor prin diferite metode.*

## ➢ USE-CASE



# 3. Proiectare

## ➢ Proiectarea OOP

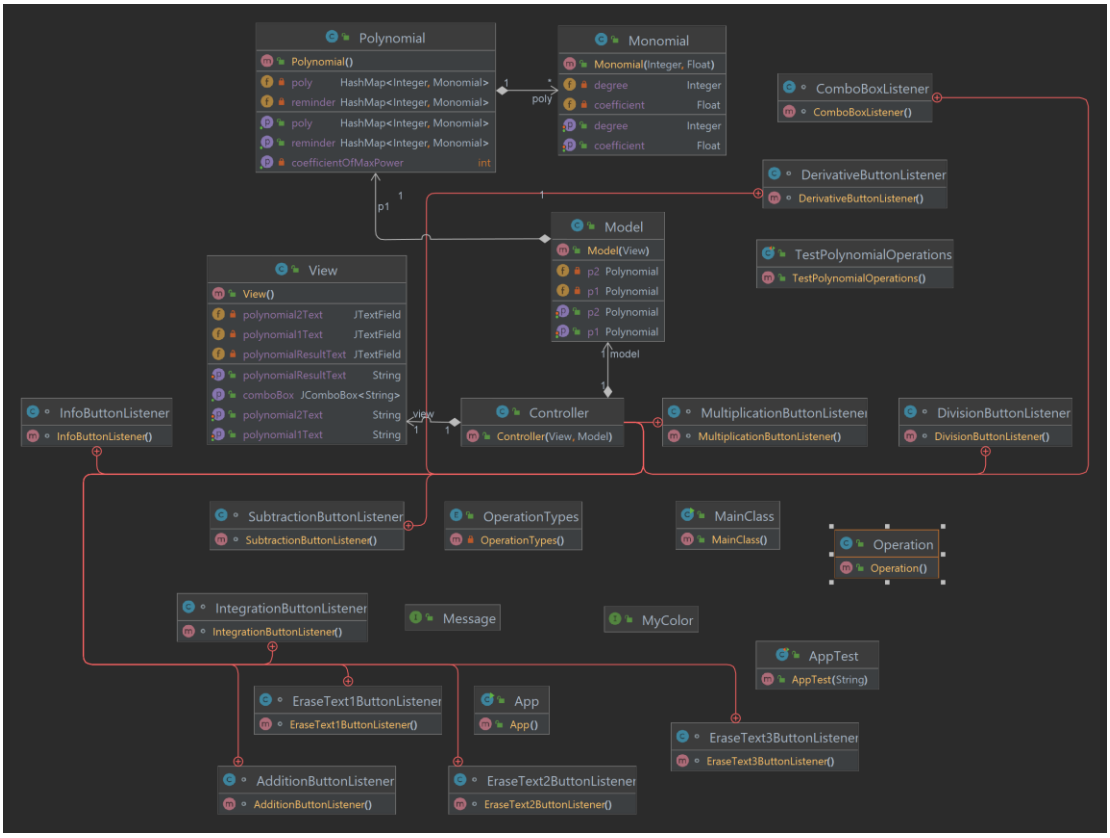
❖ Functionalitatea aplicatiei este bazata pe 2 clase principale:

- **Monomial** → clasa reprezinta un monomial ( un polinom cu un singur termen)
- **Polynomial** → clasa reprezinta o multime de monomiali

Suplimentar, aplicatia implementeaza si clasa **Operation** ce implementeaza anumite operatii ce nu au legatura directa cu operatiile legate de Polinom.

De asemenea, aplicatiei este construita dupa pattern-ul MVC . Prin urmare , aceasta este alcatuita din 3 componente interconectate : **Model** ( formata din clasele Monomial,Polynomial,Operation) , **View** (ce afiseaza datele din model) si **Controller** (care conecteaza modelul si view-ul si controleaza fluxul de date : primeste input-ul dat de utilizator, este translatat prin intermediul clasei Operation si transmis mai depare clasei Polynomial, procesul fiind reversibil pentru afisarea rezultatelor);

Descrierea detaliata a fiecarei clase poate fi regasita in sectiunea **4.**



- **Structura de date** folosita in program este un Tabel de dispersie, cheile fiind date de puterea unui termen al polinomului , iar valoarea efectiva stocata in cheia respectiva este un termen de tip Monomial: H**ashMap<Integer,Monomial>**

- Printre algoritmii folositi se regasesc :
  - Algoritm de separare a coeficientilor si puterilor de variabila x prin intermediul bibliotecii **REGEX**.
  - Algoritmi elementari de adunare, scadere, inmultire, impartire, derivare, integrare a polinoamelor.

# 4. Implementare

Aplicatia , construita dupa pattern-ul MVC, implementeaza cele 3 componente astfel:

- **Model :**
  - Modelul aplicatiei este constituit 3 clase ( in pachetul **com.datamodel** )
    - **Clasa Monomial**
      - Este alcatuita din campurile **degree** (Integer) si **coefficient** (Float) ce reprezinta gradul si respectiv coeficientul unui monom;
      - Metodele ce se regasesc in cadrul acestei clase sunt:
        - metode efectueaza adunarea /scaderea /inmultirea /impartirea monomului specific obiectului din care e instantata clasa cu o variabila de tip Monom:
          - **public void addition (Monomial a) ;**
          - **public void subtraction (Monomial a) ;**
        - metode statice ce efectueaza operatii inmultire/ impartire/ integrare /derivare cu argumentele date . Asadar, pentru aceasta categorie de operatii nu e necesara instantarea obiectelor de tip Monomial. Implementarea lor a fost necesara pentru eficientizare si lizibilitatea codului din clasa Polynomial:
          - **public static Monomial multiplication (Monomial a, Monomial b);**
          - **public static Monomial division(Monomial a, Monomial b);**
          - **public static Monomial derivative(Monomial a);**
          - **public static Monomial integral(Monomial a);**
        - metoda ce returneaza un monom sub forma unui string:
          - **public String toString();**
        - constructor, gettere, settere;

    - **Clasa Polynomial**

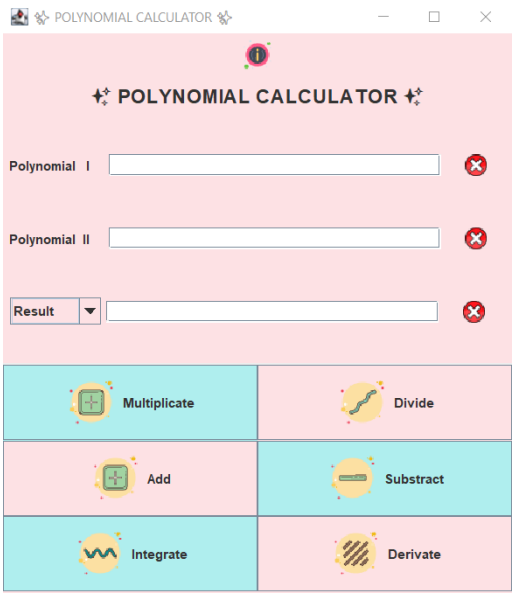
- Este alcatuita din campurile **poly** (HashMap< Integer, Monomial >) si **reminder** (HashMap< Integer, Monomial >) ce reprezinta polinomul efectiv si respectiv restul ( care doar in urma operatiei de impartire are o valoare diferita de zero )
- Metodele ce se regasesc in cadrul acestei clase sunt:

  - metode de clasa ce efectueaza operatiile necesare efectuarii calculelor cu multipli monomiali:
    - **public void additionPolynomial(Polynomial q);**
    - **public void subtractionPolynomial(Polynomial q);**
    - **public void multiplicationPolynomial(Polynomial q);**
    - **public void divisionPolynomial(Polynomial q);**
    - **public void derivativePolynomial();**
    - **public void integralPolynomial();**
  - metoda privata ce returneaza cel mai mare grad din polinom:
    - **private int getCoefficientOfMaxPower();**
  - metoda ce returneaza un monom sub forma unui string; in functie de valoarea parametrului s, ea poate sa afiseze polinomul propriu-zis sau polinomul reminder:
    - **public String toString(String s);**
  - constructor, gettere, settere;

- **Clasa Operation**
  - Implementeaza metode ce nu au neaparat legatura cu efectuarea de operatii pe polinoame
    - Este definita o singura metoda ce converteste un string intr-un tip de data Polynomial **:**

      **public static Polynomial stringWithRegex(String exp);**

❖ **View-ul aplicatiei** (in pachetul **com.mvc** )




- In cadrul acestei clase sunt definite multiple **campuri**, clasificandu-se astfel:
  - pentru butoanele de operatii;
  - pentru field-urile de text;
  - pentru label-uri;
  - pentru panel-uri;
    - metoda ce returneaza un monom sub forma unui string:
      - **public String toString();**
    - constructor, gettere, settere;

- Pentru lizibilitatea si mai buna organizare a codului am ales sa implementez mai multe metode ce executa diverse task-uri comune pentru obiecte diferite:
  - **public void initElements();** → initializeaza fiecarui obiect folosit.
  - **public void setColor();** → coloreaza butoanele, background-ul, etc.
  - **public void setIcon();** → seteaza icon-uri pentru butoanele de operatii.
  - **public void addElementsToPanels();** → adaugarea elementelor in panel-uri / reuniunea mai multor panel-uri cu layout-uri diferite**.**
  - **public void setLayoutsForPanels();** → setare layout-uri pentru panel-uri.
  - **public void setTextFont();** → setarea fontului pentru text;
  - constructor, gettere, settere,etc.

❖ **Controller-ul aplicatei** (in pachetul **com.mvc** ) :

- Spre deosebire de restul claselor descrise pana acum, in cadrul acestei clase sunt implementate clase abstracte pentru Listeners; am abordat aceasta metoda de implementare deoarece numarul de Ascultatori este este mai natural si mai simplu de vizualizata clasa ce e apelata din controller.
- **Campuri: view** si **model**;
- **Metode:**
  - **public void operation(OperationTypes opType);** → folosita pentru a decide care dintre butoanele de operatii a fost apasat si pentru a apela operatia corespunzatoare din model;
  - **private void showMessageDialogCustom(String s, boolean asError);** → deschide o noua fereastra in aplicatie ce afiseaza mesajele primite la interceptarea unor Exceptii sau Erori

❖ **Interfete** ( in pachetul **com.constant** )

- **Message** → interfata in care sunt definite 2 constante String ***INFO*** si ***NULL_POLY*** ce retin mesaje folosite ulterior in aplicatie;
- **MyColor** → interfata in care sunt definite 3 constante de tip Color ***MY_PINK, MY_SECOND_PINK*** si ***MY_BLUE*** care definesc 3 culori RGB;

❖

❖ **Enum** ( in pachetul **com.enums** ) : implementeaza **OperationTypes**.

# 5. Rezultate

Testarea functionalitatii aplicatiei a fost constituita atat din teste din perspectiva userului cu date ale caror rezultat era cunoscut, introduse direct din interfata grafica, cat si din testarea unitara cu utilitarul JUnit.

Implementarea programului ce foloseste utilitarul se regaseste in pachetul **src.test ,** unde sunt definite metodele urmatoare ( + input / resultat asteptat ) :

- additionTest();
  - **input:** → p1 : 4x^5-3x^4+x^2-8x+1;
    → p2 : 3x^4-x^3+x^2+2x-1;
  - **rezultat:** 6x^1+2x^2-1x^3+4x^5;

- subtractionTest();
  - **input:** → p1 : 4x^5-3x^4+x^2-8x+1;
    → p2 : 3x^4-x^3+x^2+2x-1;
  - **rezultat:** 2-10x^1+1x^3-6x^4+4x^5;

- multiplicationTest();
  - **input:** → p1 : 3x^2-x+1;
    → p2 : x-2;
  - **rezultat:** -2+3x^1-7x^2+3x^3;

- divisionPolynoimial();
  - **input:** → p1 : x^3-2x^2+6x-5;
    → p2 : x^2-1;
  - **rezultat:** → polinom : -2+1x^1
    → reminder : -7+7x^1

- integralPolynomial();
  - **input:** → 12x^5-15x^4+3x^2-8x+1;
  - **rezultat:** 1x^1-4x^2+1x^3-3x^5+2x^6;

- derivativePolynomial();
  - **input:** → 12x^5-15x^4+3x^2-8x+1;
  - **rezultat:** -8+6x^1-60x^3+60x^4;

# 6. Concluzii

In concluzie, aplicatia ofera o modalitate eficienta si usor de utilizat pentru efectuarea operatiilor de baza cu polinoame. Proiectul in sine m-a ajutat sa ma familiarizez cu software-ul GitHub ( crearea unui repository, push, commit, pull, etc) si cu IDE-ul IntelliJ IDEA. Crearea aplicatiei m-a ajutat la extinderea cunostintelor in ceea ce priveste design-ul OOP, crearea unei interfete grafice folosind Java Swing, lucrul cu expresii regulate, dar si aspecte referitoare la organizarea codului (utilizarea pachetelor, conventii de nume,etc).

Posibile imbunatatiri ale aplicatei pot fi crearea unei baze de date pentru a stoca input-urile date de utilizator astfel incat sa nu fie necesara efectuarea acelorasi operatii de mai multe ori. Tot prin intermediul unei baze de date se poate dezvolta o functie de completare automata sau selectie de input-uri, istoric cu operatiile recente efectuate, s.a. O alta functionalitate utila ar fi aceea de corectare automata a greselilor. Operatiile pot fi extinse la efectuarea de ecuatii si reprezentari grafice.

# 7. Bibliografie

I. FUNDAMENTAL PROGRAMMING TECHNIQUES – curs : https://dsrl.eu/courses/pt/
II. Programare Orientata pe Obiecte – Ionel Giosan : https://users.utcluj.ro/~igiosan/teaching_poo.html
III. Utilizare Regex pentru expresii polinomiale : https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression
IV. Layouts in Java : https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
V. Free Icons : https://icons8.com/icons/set/button
VI. How to add Icon to JButtons : https://www.tutorialspoint.com/how-to-add-icon-to-jbutton-in-java
VII. JUnit setup for IntelliJ IDEA : https://www.jetbrains.com/help/idea/junit.html
VIII. JUnit – IntelliJ with Maven setup tutorial: https://www.youtube.com/watch?v=cTEtSmNOtlE&t=290s&ab_channel=Randomcode