DOCUMENTATIE

TEMA 1

NUME STUDENT: Zubașcu Maria
GRUPA: 30224

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei îl constituie tehnicile fundamentale de programare prin realizarea unui calculator care efectuează operații pe polinoame. Acest proiect contine si o interfata grafica, in care utilizatorul poate insera polinoame si poate vizualiza rezultatele operatiilor alese.

Obiective secundare:

- *Analiza problemei si identificarea cerintelor (Cap.2)*
- *Proiectarea calculatorului polinomial (Cap.3)*
- *Implementarea calculatorului polinomial (Cap.4)*
- *Testarea calculatorului polinomial (Cap.5)*

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

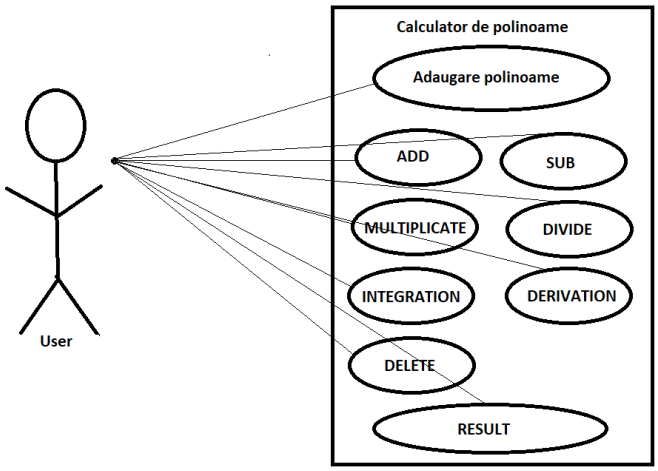
Cerintele functionale:

- *Calculatorul de polinoame trebuuie sa permita utilizatorilor sa introduca polinoame*
- *Calculatorul de polinoame trebuie sa permita utilizatorului sa selecteze operatia matematica dorita*
- *Calculatorul de polinoame ar trebui sa afiseze rezultatatul operatiei dupa selectarea acesteia*
- *Calculatorul de polinoame ar trebui sa contina un buton care sa permita revenirea calculatorului la starea initiala (clear)*

Cerintele non functionale:

- *Calculatorul de polinoame ar trebui sa fie intuitiv si usor de utilizat de catre utilizator*
- *Interfata calculatorului ar trebui sa aiba un aspect prietenos fata de utilizator*
- *Calculatorul de polinoame nu va raspunde la introducerea eronata a unui polinom*

Cazuri de utilizare:



USE-CASE

- ***Adunare polinoame***
  Actorul principal: utilizatorul
  Scenariul de succes:
  1. *Utilizatorul adauga in interfata grafica cele doua polinoame*
  2. *Utilizatorul apasa butonul corespunzator operatiei de adunare (“ADD”)*
  3. *Calculatorul efectueaza adunarea si afiseaza operatia*

- ***Scadere polinoame***
  Actorul principal: utilizatorul
  Scenariu de succes:
  1. *Utilizatorul adauga in interfata grafica cele doua polinoame*
  2. *Utilizatorul apasa butonul corespunzator operatiei de scadere (“SUB”)*
  3. *Calculatorul efectueaza scaderea si afiseaza operatia*

- ***Inmultire polinoame***
  Actorul principal: utilizatorul
  Scenariu de succes:
  1. *Utilizatorul adauga in interfata grafica cele doua polinoame*
  2. *Utilizatorul apasa butonul corespunzator operatiei de inmultire (“MULTIPLICATE”)*
  3. *Calculatorul efectueaza inmultirea si afiseaza operatia*

- ***Impartire polinoame***

Actorul principal: utilizatorul
Scenariu de succes:
1. *Utilizatorul adauga in interfata grafica cele doua polinoame*
2. *Utilizatorul apasa butonul corespunzator operatiei de impartire (“DIVIDE”)*
3. *Calculatorul efectueaza impartirea si afiseaza operatia*

- ***Integrare polinom***
  Actorul principal: utilizatorul
  Scenariu de succes:
  1. *Utilizatorul adauga in interfata grafica polinomul in campul denumit “Polynom 1”*
  2. *Utilizatorul apasa butonul corespunzator operatiei de integrare (“INTEGRATION”)*
  3. *Calculatorul efectueaza integrarea si afiseaza operatia*

- ***Derivare polinom***
  Actorul principal: utilizatorul
  Scenariu de succes:
  1. *Utilizatorul adauga in interfata grafica polinomul in campul denumit “Polynom 1”*
  2. *Utilizatorul apasa butonul corespunzator operatiei de derivare (“DERIVATION”)*
  3. *Calculatorul efectueaza derivarea si afiseaza operatia*

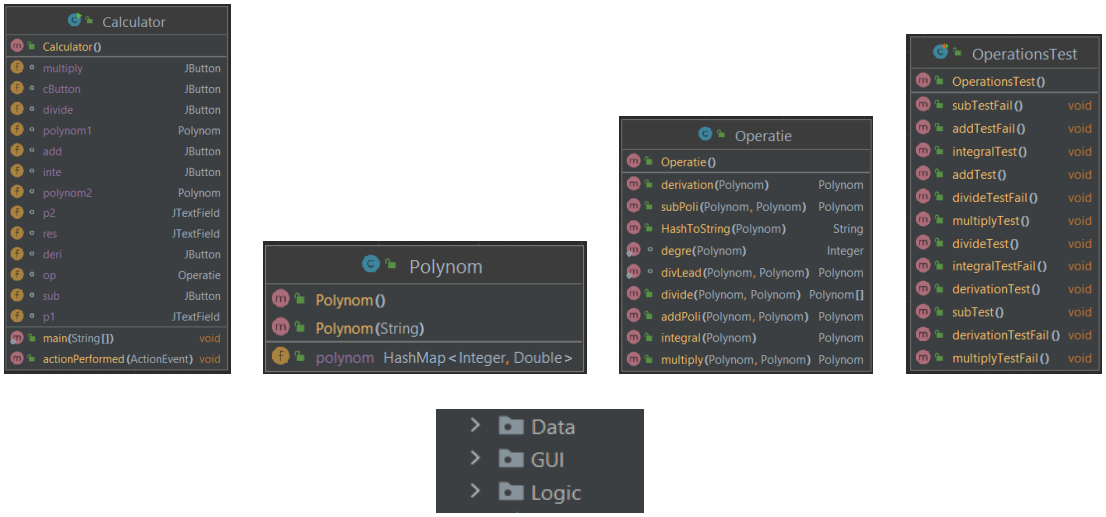
  Secventa alternativa: polinoamele sunt introduse gresit (trebuie sa fie de forma CoeficientLitera^Putere, coeficientul este optional daca variabila este precedata de semn, litera este optionala daca exista un coeficient, ^Putere este optionala daca exista litera sau coeficient)
  -utilizatorul a introdus polinoamele gresit
  -utilizatorul a selectat o operatie
  -se va afisa un rezultat partial sau niciun rezultat
  -utilizatorul poate sterge informatia din campuri apasand butonul de stergere (“C”)
  -scenariul revine la pasul 1

3. Proiectare

Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite, interfetele definite si algoritmii folositi

Pentru a stoca polinomul citit initial ca String, am folosit structura HashMap<Integer, Double> in care am adaugat fiecare monom (Cheia este gradul monomului, iar coeficientul reprezinta valoarea), folosind Pattern si Matcher am determinat fiecare monom. Rezultatul operatiei de impartire l-am stocat intr-un tablou de tip “Polynom”. Pentru dezvoltarea interfetei grafice am folosit Java Swing.

Diagrame UML de clase si pachete



4. Implementare

Clasa Polynom

-se afla in pachetul "Data"
-este clasa in care se afla polinomul in structura de tip HashMap<Integer,Double>
-contine un constructor care primeste un String, am folosit Matcher si Pattern pentru a diviza polinomul in 3 grupuri: semn, variabila, putere, dupa care am abordat cazurile optionale si am adaugat puterea si coeficientul intr-un HaspMap<Integer, Double>.
-al doilea constructor care nu primeste niciun parametru este folosit pentru initierea rezultatelor in care urmeaza sa fie pus noul polinom in urma unei operatii

Clasa Operatie

-se afla in pachetul "Logic"
-contine un constructor fara parametri
-aceasta clasa contine metodele cu operatiile pe polinoame implementate astfel:

- ***addPoli:*** *aceasta metoda parcurge al doilea polinom, la fiecare pas verifica daca in primul polinom exista cheia curenta, daca da face adunare intre cele doua valori, altfel adauga valoarea si cheia, curente, in primul polinom. Rezultatul este stocat in primul polinom.*

```java
public Polynom addPoli(Polynom a, Polynom b)
{
    for(Map.Entry<Integer, Double> entry: b.polynom.entrySet())
    {
        if(a.polynom.containsKey(entry.getKey()))
        {
            Double s= a.polynom.get(entry.getKey());
            s+=entry.getValue();
            if(s.compareTo(0.0)==0)
                a.polynom.remove(entry.getKey());
            else
                a.polynom.put(entry.getKey(),s);
        }
        else a.polynom.put(entry.getKey(),entry.getValue());
    }
    return a;
}
```

- ***subPoli:*** *aceasta metoda parcurge al doilea polinom, la fiecare pas verifica daca in primul polinom exista cheia curenta, daca da face scadere intre cele doua valori, altfel adauga valoarea cu semn schimbat si cheia, curente, in primul polinom. Rezultatul este stocat in primul polinom.*

```java
public Polynom subPoli(Polynom a, Polynom b)
{
    for(Map.Entry<Integer, Double> entry: b.polynom.entrySet())
    {
        if(a.polynom.containsKey(entry.getKey()))
        {
            Double s= a.polynom.get(entry.getKey());
            s-=entry.getValue();
            s=Math.round(s*100)/100.0;
            if(s.compareTo(0.0)==0)
                a.polynom.remove(entry.getKey());
            else
                a.polynom.put(entry.getKey(),s);
        }
        else a.polynom.put(entry.getKey(),-entry.getValue());
    }
    return a;
}
```

- ***derivation:*** *aceasta metoda parcurge un polinom si la fiecare pas adauga in polinomul rezultat cheia si coeficientul, calculate dupa metoda de derivare a unui polinom*

```java
public Polynom derivation(Polynom a)
{
    Polynom rez=new Polynom();

    for(Map.Entry<Integer,Double> entry : a.polynom.entrySet())
    {
        Integer k=entry.getKey();
        Double val=entry.getValue();
        rez.polynom.put(k-1,k*val);
    }

    return rez;
}
```

- ***integral:*** *aceasta metoda parcurge un polinom si la fiecare pas adauga in polinomul rezultat cheia si coeficientul, calculate dupa metoda de integrare a unui polinom*

```java
public Polynom integral(Polynom a)
{
    Polynom rez=new Polynom();
    a.polynom.forEach((k,val) ->
    {
        Double s=val/(k+1);
        s=Math.round(s*100)/100.0;
        rez.polynom.put(k+1,s);
    });
    return rez;
}
```

- ***multiply:*** *aceasta metoda returnează un obiect de tip Polynom care reprezintă înmulțirea a două obiecte de același tip. Se parcurg cele două polinoame, se adună puterile celor două monoame și se înmulțesc coeficienții.În cazul în care gradul polinomului rezultat conține deja un monom de același grad, se adună coeficienții, după care se stohează rezultatul în noul polinom*

```java
public Polynom multiply(Polynom a, Polynom b)
{
    Polynom rez=new Polynom();

    a.polynom.forEach((ka,va) ->
    {
        b.polynom.forEach((kb,vb) ->
        {
            Integer pow=ka+kb;
            Double coef=va*vb;
            if(rez.polynom.containsKey(pow))
            {
                Double s=coef+rez.polynom.get(pow);
                rez.polynom.replace(pow,s);
            }
            else rez.polynom.put(pow,coef);
        });
    });
    return rez;
}
```

- ***divide:*** *aceasta metoda este implementata dupa pseudocodul de impartire a doua polinoame de mai jos:*

```
function n / d is
    require d ≠ 0
    q ← 0
    r ← n             // At each step n = d × q + r

    while r ≠ 0 and degree(r) ≥ degree(d) do
        t ← lead(r) / lead(d)       // Divide the leading terms
        q ← q + t
        r ← r − t × d

    return (q, r)
```

- *aceasta metoda foloseste alte doua metode: determinarea gradului unui polinom (degre), iar cealalta returneaza rezultatul impartirii a doua monoame (divLead)*

```java
static Integer degre(Polynom a)
{
    Integer c = 0;
    for (Map.Entry<Integer, Double> entry : a.polynom.entrySet()) {
        c= entry.getKey();
    }

    return c;
}
```

```java
static Polynom divLead(Polynom r, Polynom d)
{
    Polynom t=new Polynom();
    Integer key =degre(r)-degre(d);
    Double val=r.polynom.get(degre(r)) / d.polynom.get(degre(d));
    val=Math.round(val*100)/100.0;
    t.polynom.put(key,val);
    return t;
}
```

- ***HashToString:*** *aceasta metoda returneaza un String care reprezinta conversia unui polinom stocat in HashMap<Integer, Double> intr-un sir de caractere de forma "aX^b+a1X^b1+..."*

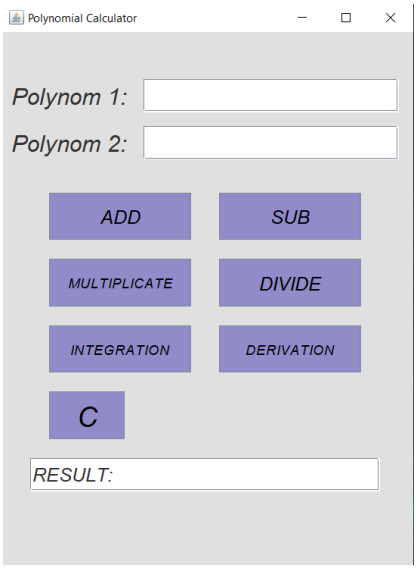
```java
public String HashToString(Polynom p)
{
    String s="";
    for(Map.Entry<Integer, Double> entry: p.polynom.entrySet())
    {
        if(entry.getValue()>0)
        s+="+"+entry.getValue() + "X^" + entry.getKey();
        else s+=entry.getValue() + "X^" + entry.getKey();
    }
    char c=s.charAt(0);
    if(Character.compare(c, '+')==0)
        s=s.substring( beginIndex: 1);

    return s;
}
```

Clasa Calculator

-se afla in pachetul GUI
-in aceasta clasa este implementata interfata calculatorului de polinoame folosind Java Swing.

Am folosit obiecte de tip JFrame (cadrul principal), JTextField (field-uri pentru scrierea polinoamelor si afisarea rezultatelor), JLabel (folosit in descrierea text field-urilor) si JButton
(butoane pentru selectarea operatiilor si clear)
-contine metoda main care lanseaza aplicatia



5. Rezultate

Testarea operatiilor a fost facuta cu testarea unitara JUnit in clasa "OperationsTest". Am adaugat dependentele in pom.xml pentru a configura Marven. Pentru fiecare operatie am creeat o metoda de test si o metda de testFail. (in total au fost 12 teste, 6 gresite si 6 corecte).

6. Concluzii

In concluzie, Tema1 m-a ajutat sa-mi imbunatatesc modul de a aborda o problema, am invatat cum sa organizez codul Java astfel incat sa fie lizibil si simplu, usor de inteles.
De asemenea, am invatat cum sa fac diferenta intre cerintele functionale si non functionale ale unei probleme, fapt care faciliteaza rezolvarea unei probleme.

Posibilele dezvoltari ulterioare ale acestui proiect ar fi: o implementare eficienta a operatiilor precum inmultirea si impartirea; tratarea unor exceptii care apar la delimitarea polinomului si atentionarea utilizatorului; de asemenea, implementarea operatiilor care sa permita calculul cu radicali sau afisarea rezultatelor sub forma de fractie.

7. Bibliografie

1. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.*
2. *What are Java classes? - www.tutorialspoint.com*
3. *https://www.javatpoint.com/java-hashmap*
4. *https://www.w3schools.com/java/java_regex.asp*
5. *https://regexr.com/*