DOCUMENTAȚIE

TEMA1

Nume Student: Pode Dana Ioana

Grupa: 30222

Cuprins

1. Obiectivul temei

Scopul temei este de a implementa un calculator polinomial cu interfață grafică prin intermediul căruia utilizatorul poate introduce polinoame pentru care se vor efectua diverse operații precum adunare, scădere, împărțire, înmulțire, derivare și integrare.

Obiectivele secundare:

i. Înțelegerea cerinței pentru întocmirea claselor si sublaselor necesare, structura acestora *(cap 2)*
ii. Structura codului într-o arhitectura tip MVC(Model-View-Controller) *(cap 3)*
iii. Crearea claselor de Monom (care conține operațile pe monoame) si Polynomial (care conține o listă de monoame cu scopul de a reprezenta un polinom ca și întreg) ca și parte a Model ului *(cap 4)*
iv. Crearea unei interfeței grafice care sa permită interacțiunea cu utilizatorul, ușor de implementat și folosit
v. Implementarea operaților necesare pentru calculul cu polinoame
vi. Testarea operaților folosind testarea unitară-Junit

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

Cerințele funcționale: calculatorul de polinoame să poată primi două polinoame și în funcție de cerința aleasă să realizeze calcule cu acestea. La analiza cerinței identificăm datele de intrare/ieșire astfel:

Date de intrare: - primul polinom

- al doilea polinom

- operația selectată de utilizatorul aplicației (respectiv adunare, scădere, înmulțire, derivare, integrare)

Date de ieșire: - rezultatul operației

Cerințe non-funcționale: aplicație accesibilă la fiecare nouă operație (consistența operaților) , dispunere mesajelor de eroare in cazul unui input(polinom) greșit, transparență

Scenariul principal de utilizare a calculatorului polinomial: utilizatorul introduce de la tastatura două polinoame sub o formă de scriere prestabilită. Acesta selectează operația dorită printr un click la butonul operației și in partea de jos îi apare rezultatul.

Polinoamele date sunt parsate in clasa GUI care joacă rolul de view și controller ca mai apoi, folosindu se clasele Monom, Operations si Polynomial cu scopul de Model a aplicației pentru realizarea operației, să se afișeze rezultatul dorit. Pentru simplitate am ales ca clasa GUI să facă atât interfața grafică (view) cât si legătura între interfață cu modelul aplicației (controller).

Scenariu secundar:- utilizatorul nu adaugă un polinom dar selectează o operație, caz în care primeste mesajul de eroare ,,You have to add something"

-utilizatorul adaugă ambele polinoame dar acestea nu respectă pattern ul (forma) prestabilită, se afișează mesajul ,,Wrong Input"

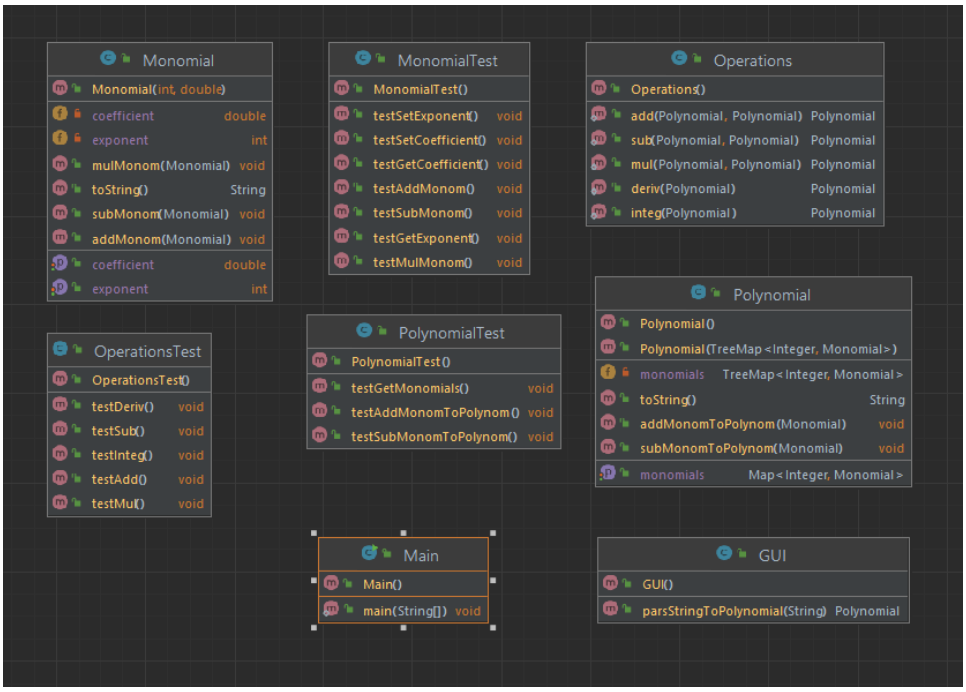
3. Proiectare

Arhitectura: Model-View-Controller. . View-ul cuprinde elementele din interfata, acea parte din program cu care utilizatorul intra in contact direct. Controller-ul este cel care face legătura intre View și Model. Acesta este cel care ia polinoamele introduse de către utilizator, le prelucrează pentru a corespunde modelului de date folosit, apelează operațiile din model și preia rezultatul din model, pe care-l transmite în interfață. Modelul cuprinde toate structurile de date folosite si operatiile care se pot realiza cu acestea.

Pentru a descrie modelul aplicatiei am folosit in schimb trei clase: Monomial care contine descrierea unui monom prin exponent si coeficient, Polynomial care contine un TreeMap de monoame care descriu astfel un polinom întreg și clasa Operations care implementează operațile necesare pentru calculul cu polinoame.

Pentru a descrie interfata aplicatiei si legatura dintre aceasta cu modelul, am folosit o singura clasa numita GUI. Interfața este alcătuită din label-uri pentru inserarea celor doua polinoame, butoane pentru operații, un buton de reluare si un loc unde se afisează rezultatul operației. Legătura dintre interfață si model am realizat-o in funcție de ActionListener, iar datele au fost preluate parsând șirul de caractere de polinoame după un anumit pattern(regex).

Diagrama UML este următoarea:



4. Implementare

Clasa Polynomial

Clasa care conține descrierea polinoamelor ca și colecții de monoame. Aceasta conține un constructor util care instațiază noul obiect de tip TreeMap. Acest TreeMap reprezinta o colectie de polinoame <Integer, Monomial>, permițând accesul pentru fiecare monom în fucție de cheia acestuia.

Cele doua funcții addMonomToPolynom și subMonomToPolynom adaugă respectiv scad un monom la un polinom astfel: dacă acel monom există, se verifică dacă mai exista monomcu acelasi exponent, iar dacă da se inlocuiește acel monom cu monomul sumă/diferență schimbând coeficientul.

```java
public void addMonomToPolynom( Monomial monom){
    if(monom.getCoefficient()!=0) {
        if(monomials.containsKey(monom.getExponent())){
            double coefAux=monomials.get(monom.getExponent()).getCoefficient();
            monomials.remove(monom.getExponent());
            if(coefAux+monom.getCoefficient()!=0) {
                monomials.put(monom.getExponent(), new Monomial(monom.getExponent(),  coefficient: coefAux + monom.getCoefficient()));
            }
        }
        else{
            monomials.put(monom.getExponent(), new Monomial(monom.getExponent(),  monom.getCoefficient()));
        }
    }

}
```

```java
public void subMonomToPolynom( Monomial monom){
    if(monom.getCoefficient()!=0) {
        if(monomials.containsKey(monom.getExponent())){
            double coefAux=monomials.get(monom.getExponent()).getCoefficient();
            monomials.remove(monom.getExponent());
            if(coefAux-monom.getCoefficient()!=0) {
                monomials.put(monom.getExponent(), new Monomial(monom.getExponent(),  coefficient: coefAux - monom.getCoefficient()));
            }
        }
        else{
            monomials.put(monom.getExponent(), new Monomial(monom.getExponent(),  -monom.getCoefficient()));
        }
    }
}
```

Această clasa conține și o metodă de toString() aferentă care afișează un polinom.

Clasa Monomial

Clasa care conține operațiile elementare de adunare, scădere, înmulțire și împărțire și exponentul (tip int) respectiv coeficientul (tip double) unui monom. Aceste operații sunt relative simple, lucrul cu monoamele fiind mult mai ușor in comparative cu polinoamele.

```java
public void addMonom(Monomial monom){

    if(monom.exponent==this.exponent){
        this.coefficient=this.coefficient+monom.coefficient;
    }
}
1 usage
public void subMonom(Monomial monom){

    if(monom.exponent==this.exponent){
        this.coefficient=this.coefficient-monom.coefficient;
    }
}
1 usage
public void mulMonom(Monomial monom){
    if(monom.exponent==this.exponent){
        this.coefficient=this.coefficient*monom.coefficient;
        this.exponent=this.exponent+monom.exponent;
    }
}
```

Clasa Operations

Această clasă descrie operațile pe polinoame, și anume cele de:adunare, scădere, înmulțire, derivare și integrare a unui sau a două polinoame.

Metodele pentru adunare/scădere a două polinoame:

```java
public static Polynomial add(Polynomial pol1, Polynomial pol2){
    Polynomial addResult=new Polynomial();
    addResult=pol1;
    for(Monomial m1: pol2.getMonomials().values()){
        addResult.addMonomToPolynom(m1);
    }
    return addResult;
}
2 usages
public static Polynomial sub(Polynomial pol1, Polynomial pol2){
    Polynomial subResult=new Polynomial();
    subResult=pol1;
    for(Monomial m1: pol2.getMonomials().values()){
        subResult.subMonomToPolynom(m1);
    }
    return subResult;
}
```

Operțile de adunare și scădere sunt similare. Acestea parcurg colectia de monoame a unui polinom si adaugă folosind metodele addMonomToPolinom/ subMonomToPolinom fiecare monom din al doilea polinom in primul polinom în care se stochează rezultatul final.

Metoda pentru înulțire a două polinoame:

```java
public static Polynomial mul(Polynomial pol1, Polynomial pol2){
    Polynomial mulResult=new Polynomial();
    Polynomial mulAux=new Polynomial();
    Monomial mulMonom=new Monomial( exponent: 1, coefficient: 1);
    for(Monomial m1: pol1.getMonomials().values()){
        for(Monomial m2: pol2.getMonomials().values()){
            mulMonom=new Monomial( exponent: m1.getExponent()+ m2.getExponent(),  (int) (m1.getCoefficient()* m2.getCoefficient()));
            mulAux.getMonomials().put(m1.getExponent()+ m2.getExponent(),mulMonom);
        }
        mulResult=Operations.add(mulResult,mulAux);
        mulAux.getMonomials().clear();
    }
    return mulResult;
}
```

Această metoda relizează inmultirea succesiva a doua poliniame, monom cu monom și adaugă rezultatul in variabila mulResult.

Metodele pentru derivare/integrare a unui polinom:

```java
public static Polynomial deriv(Polynomial pol1){
    Polynomial derivResult=new Polynomial();
    for(Monomial m1: pol1.getMonomials().values()){
        if(m1.getExponent()!=0){
            m1.setCoefficient(m1.getCoefficient()*m1.getExponent());
            m1.setExponent(m1.getExponent()-1);
            derivResult.addMonomToPolynom(m1);
        }
    }
    return derivResult;
}
2 usages
public static Polynomial integ(Polynomial pol1){
    Polynomial derivResult=new Polynomial();
    for(Monomial m1: pol1.getMonomials().values()){
        if(m1.getExponent()!=0){
            m1.setCoefficient((m1.getCoefficient()/(m1.getCoefficient()+1)));
            m1.setExponent(m1.getExponent()+1);
            derivResult.addMonomToPolynom(m1);
        }else{
            m1.setExponent(m1.getExponent()+1);
            derivResult.addMonomToPolynom(m1);
        }
    }
    return derivResult;
}
```

Aceste metode sunt similare. Preiau un monom pe care il modifica monom cu monom in forma derivata/integrata, având modificări asupra coeficientului și exponentului.

Clasa GUI

Partea View:

În această clasă am declarat componentele interfeței.

Partea Controller:

Printr-un ActionListener am implementat comportamentul fiecărui buton la apasarea acestuia pentru a se reliza operația din controller și a se afișa în interfața utilizatorului rezultatul.

```java
bAdd.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Polynomial pol1,pol2,res;
        pol1=new Polynomial();
        pol2=new Polynomial();
        res=new Polynomial();
        pol1=parsStringToPolynomial(firstPolynom.getText());
        pol2=parsStringToPolynomial(secondPolynom.getText());
        res= Operations.add(pol1,pol2);
        result.setText(res.toString());
    }
});
```

Una dintre cele mai importante metode o reprezintă metoda *parsStringToPolynomial(String string)*

*Aceasta parsează sirul de caractere dat ca date de intare sub forma unui pattern prestabilit și se tranformă in sir de monoame, respectiv polinom. Totodată aici tratez cazul in care nu a fost dat nici o dată de intare sau datele introduse de utilizator nu corespund pattern-ului, afisand mesaje aferente „*You have to add something” *și „*Wrong Input*”.*

```java
public Polynomial parsStringToPolynomial(String string){
    Polynomial result=new Polynomial();
    //daca string ul exista
    if(!string.equals("")){
        Matcher matcher= pattern.matcher(string);
        String stringAux="";
        while(matcher.find()){
            String matchGroup= matcher.group();//grupul la care se afla
            stringAux=stringAux+matchGroup;
            Scanner scaner;
            scaner = new Scanner(matchGroup);
            scaner.useDelimiter( pattern: "x\\^");
            double saveCoeff;
            int saveExp;
            saveCoeff=scaner.nextDouble();
            saveExp=scaner.nextInt();
            result.addMonomToPolynom(new Monomial(saveExp,saveCoeff));
        }
        if(!stringAux.equals(string)){
            JOptionPane.showMessageDialog( parentComponent: this, message: "Wrong Input");
            return new Polynomial();
        }
        return result;
    }else{
        JOptionPane.showMessageDialog( parentComponent: this, message: "You have to add something");
        return new Polynomial();
    }
}
```

5. Rezultate

Rezultatele JUnit arată posibilitatea implementari corecte a operaților și funcționarea aplicației.

```java
public void testAdd() {
    Monomial m1 = new Monomial( exponent: 2,  coefficient: 3.0);
    Monomial m2 = new Monomial( exponent: 1,  coefficient: 2.0);
    Polynomial pol1 = new Polynomial();
    Polynomial pol2 = new Polynomial();
    pol1.addMonomToPolynom(m1);
    pol1.addMonomToPolynom(m2);
    pol2.addMonomToPolynom(m1);
    new Polynomial();
    Polynomial res1 = Operations.add(pol1, pol2);
    assertEquals(res1.toString(),  actual: "+6.0x^2+2.0x^1");
}
```

6. Concluzii

Prin această temă consider că am reusit sa aprofundez cunostinte de POO învățate și am învățat sa folosesc lucruri noi ,precum folosirea Pattern Matchingului folosind expresii Regex.

Posibile dezvoltări viitoare:

-implementarea operației de impărțire

-dezvoltarea unui pattern mai complex pentru a trata posibilele erori de scriere

-integrarea unui meniu de exit, help sau readme în care sa fie descrire aceste operați și funcționarea matematică a acestora.

7. Bibliografie

https://regexr.com/

https://www.w3schools.com/java/java_regex.asp

https://dsrl.eu/courses/pt/