**Documentatie Tema 1**

**Calculator Polinoame**

Hitu Octavian

Grupa:30226

Profesor laborator: Dorin Vasile Moldovan

**Cuprins**

1.Obiectivul temei

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

2.1.Analiza problemei

2.2.Modelare

2.3.Sceanrii

2.3.Cazuri de utilizare

3.Proiectare

4.Implementare

4.1.Clasa Monom

4.2.Clasa Polinom

4.3.Clasa View

4.4.Clasa Controller

5.Testare

6.Concluzii

7.Bibliografie

1-Obiectivul temei

Obiectivul a fost proiectarea unui sistem de procesare a polinoamelor avand coeficienti si exponenti intregi. Functiile ce trebuie implmemtate sunt: adunare, scadere, inmultire, impartire, derivare si integrare. Pe langa aceste functii a mai fost implementata si o interfata pentru a sigura usurinta in utilizare a utilizatorului. Utilizatoul poate introduce doua polinoame dupa care selecteaza operatia si va vedea rezultatul.

**Obiective secundare**

**Analiza problemei, modelare, scenarii, cazuri de utilizare**- capitolul 2 -aici va avea loc prezentarea cadrului de cerinte functionale formalizat si cazurile de utilizare ca si diagrame si descrieri de use-case.

**Proiectare**- capitolul 3 -aici se va prezenta proiectarea orientat pe obiecte a calculatorului de polinoame ,diagemele UMl de clase, structurile de date folosite, algoritmii folositi si interfetele definite

**Implementare**- capitolul 4 - aici se va discuta despre descrierea claselor cu campurile si metodele importante si despre descrierea implementarii interfetei utilizatorului.

**Testare**- capitolul 5 -prezentarea scenariilor pentru testare, Junit fiind framework-ul de testare folosit.

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

2.1.Analiza Problemei

Pentru analiza problemei, am folosit metoda de design bottom-up,prin care am pornit si am implementat componentele specifice si de baza ale proiectului. Prin aceasta metoda se continua cu compunerea la un nivel mai ridicat de componente, acestea utilizand componentele implementate anterior sau componentele de baza.

Pentru implementarea proiectului am folosit o serie de etape bine definite. Am inceput cu identificarea obiectelor, dupa care a urmat gruparea lor in clase. Urmatorul pas a fost definirea ierarhiei claselor si a relatiilor dintre acestea. Un alt pas a fost acela de a alege denumiri sugestive pentru numele claselor respectiv a metodelor pentru a diminua numarul comentariilor.

2.2.Modelare

Modelarea aplicatiei se bazeaza pe conceptele programarii orientate pe obiecte, acestea fiind:

Clasele- prin clasa intelegem o forma a unui obiect deoarece aceasta reprezinta o descriere generalizata a acestuia. Clasa defineste toate atributele pe care un obiect le are ,iar metodele ce fac parte din acea clasa definesc functionalitatea acelui obiect;

Obiect- un obiect reprezinta o componenta software care incorporeaza atribute si operatii . Operatiile se pot efectua asupra atributelor .

Incapsularea- aceasta are rolul de a uni informatiile importante ale unui obiect. Ea totodata restrictioneza accesul la date si metode din exterior.

Mostenirea- din punct de vedere ierarhic aceasta permite claselor similare sa se suprapuna. Datorita ei clasele inferioare pot folosi metode din clasele superioare.

Polimorfismul- prin definitie aceasta reprezinta abilitate de a lua mai multe forme .Prin polimorfism aceeasi metoda ce este folosita intr-o superclasa poate fi suprascrisa fiind reutilizara in subclase ,ele avand o functionalitate diferita

2.3.Scenarii

La deschiderea interfetei utilizatoul va trebui sa introduca cele doua polinoame ,dupa care va trebui sa selecteze operatia. Pentru fiecare polinom exista 3 butoane: „coeficient” ,”exponent” si „adauga”. Se va adauga exponentul si coeficientul monomului dupa care butonul adauga va finaliza adaugarea monomului in polinom. Aceasta operatiune trebuie repatata pana cand se vor finaliza polinoamele. In interafata sunt 6 butoane ,cate unul pentru fiecare operatie. Aplicatia nu are o functie de sortare a polinoamelor ,de aceea polinomul trebuie introdus in ordine descrescatoare.

Exista si cateva situatii neadecvate cum ar fi:

-introducerea de valori necorespunzatoare ale coeficientului sau ale exponentului

-introducerea de caractere sau stringuri

-Introducerea directa a polinomului în campul adresat afisarii rezultatului

-Introducerea celui de-al doilea polinom, chiar daca se realizeaza o operatie ce efectuează modificari doar asupra primului polinom, cum ar fi derivare sau integrare

2.4.Cazuri de ultilizare

Calculatorul de polinoame ofera urmatoarele functionalizati:

-operatiile de adunare, scadere, inmultire, impartire pentru doua polinoame

-operatiile de derivare si integrare pentru primul polinom

**3.Proiectare**

Calculatorul de polinoame este construit din 5 clase : main, monom, polinom, view, controller. Aplicatia este oraganiza pe sablonul model-view-controller(MVC). Prin acest tip de organizare se separa interfata utilizatorului intr-o vedere (view) ce interactioneaza cu modelul si intr-un controller care raspunde la cererile utilizatorului, la randul sau interactionand cu view ul si modelul.

Text

Description automatically generated

Clase:

Monom- aceasta clasa opereaza asupra elementelui unui polinom. Are ca atribute un coeficient si un exponente si contine metode de prelucrare a monoamemelor precum comparare, negare, inmultire , derivare , integrare. In aceasta clasa mai este folosita o metoda toString care are scopul de a afisa monoamele pe ecran in locul adreselor.

Polinom- in aceasta clasa sunt implemenatate operatiile dintre polinoame cum ar fi adunare , scadere, inmultire, impartire ,derivare ,integrare. In aceasta clasa a fost ultilizat colectia de tip <List> deoarece polinoamele sunt privite ca o lista de monoame. Aceast clasa reprezinta modelul pentru sablonul model view controller.

View- In aceasta clasa am realizat interfata: am declarat butoanele, textfieldurile si am realizat aranjarea lor in interfata, setandu-le locatia in panel si marimea pe care o vor avea.

Controller- aceasta clasa realizeaza interactiunea utilizatorului cu interfata folosind ascultatorii ,care sunt implementati sub forma unor clase interne. In aceatsa clasa am definit view-ul, polinoamele si am asignat butoanele view-ului ,legandu-le la ascultatori.

Main- aceasta clasa are rolul de a instantia celelalte clase fiind top-ul acestui proiect.

**4.Implementare**

4.1.Clasa Monom

Clasa „monom” are ca si atribute coeficientul de tip float si exponentul de tip int.

public class monom {  
 private float coeficient;  
 private int exponent;  
  
 public monom(){}  
  
 public monom(float coeficient,int exponent)  
 {this.coeficient =coeficient;  
 this.exponent =exponent;}  
  
 public float getcoeficient()  
 {  
 return this.coeficient;  
 }  
 public int getexponent()  
 {  
 return this.exponent;  
 }  
  
 public void negare()  
 {this.coeficient=(-1) \* this.coeficient;  
 }  
  
 public String toString()  
 {if(this.exponent !=0 && this.coeficient >0)  
 {  
 return "+" + String.*format*("%.2f",this.coeficient) + "X^"+ this.exponent;  
 }else if(this.exponent != 0 && this.coeficient <0)  
 {  
 return String.*format*("%.2f",this.coeficient) +"X^" + this.exponent;  
 }else if(this.exponent==0 && this.coeficient>0)  
 {  
 return "+" + String.*format*("%.2f",this.coeficient);  
 }else if(this.exponent==0 && this.coeficient<0) {  
 return " " +String.*format*("%.2f",this.coeficient);  
 }else if(this.coeficient==0)  
 {return " ";}  
 else  
 return " ";}  
  
 public void inmultiremonoame(monom mon)  
 {  
 this.coeficient= this.coeficient\* mon.coeficient;  
 this.exponent = this.exponent + mon.exponent;  
  
 }  
  
 public void derivaremonom()  
 {  
 if(this.exponent >1)  
 {  
 this.coeficient=this.coeficient \* this.exponent;  
 this.exponent = this.exponent -1;  
 }else if(this.exponent ==1)  
 {  
 this.exponent=0;  
  
 }else  
 {  
 this.coeficient=0;  
 }  
 }  
  
 public void integraremonom()  
 {  
 this.coeficient=this.coeficient /(this.exponent +1);  
 this.exponent= this.exponent +1;  
 }  
  
}

Clasa monom contine metode pentru monoame care vot ajuta la implementarea operatiiilor finale pe polinoame, acestea avand nume sugestive scopul pe care il indeplinesc. Cateva metode pe care le-am implementat sunt:

Negare- monomul curent se inmulteste cu valoarea (-1). aceasta metoda se foloseste la scadere pentru a putea face suma dintre primul polinom si cel de-al doilea negat.

ToString-aceata are roulul de a afisa monoamele in interfata in fiecare caz posibil.

InmultireMonoame- aici coeficientii se vor inmulti ,iar exponentii se vor aduna. In cazul implementarii inmultirii mari aceasta metoda se va folosi pentru fiecare monom din cele doua polinoame

Derivaremonom- aceasta metoda are rolul de a deriva un monom in fiecare caz posibil

Integraremonom- aici un monom va fi integrat impartind coeficientul la exponentul caruia i s-a adunat valoarea 1, iar exponenul se va incrementa cu 1

4.2.Clasa Polinom

private List<monom> polinom = new ArrayList<monom>();

Clasa polinom are in structura sa o colectie de tip <list>, ceea ce permite o manevrare mai usoara a polinomului. Monoamele sunt stocate in polinom in ordinea adaugarii lor.

public void adaugaelementinpolinom(monom mon)  
{  
 monom monom = new monom(mon.getcoeficient(),mon.getexponent());  
 this.polinom.add(monom);  
}

Metoda adaugaelementinpolinom ,dupa cum sugereaza si numele, se ocupa cu adaugarea fiecarui monom in polinom.

public void adunarepolinoame(polinom pol1 ,polinom pol2) {  
 int i = 0;//contor polinom 1  
 int j = 0;//contor polinom 2  
 while (i < pol1.getpolinom().size() && j < pol2.getpolinom().size()) {  
 if (pol1.getpolinom().get(i).getexponent() < pol2.getpolinom().get(j).getexponent()) {  
 this.polinom.add(pol2.getpolinom().get(j));  
 j++;  
 } else if (pol1.getpolinom().get(i).getexponent() > pol2.getpolinom().get(j).getexponent()) {  
 this.polinom.add(pol1.getpolinom().get(i));  
 i++;  
 } else if (pol1.getpolinom().get(i).getexponent() == pol2.getpolinom().get(j).getexponent())  
 {  
 monom x = new monom(pol1.getpolinom().get(i).getcoeficient() + pol2.getpolinom().get(j).getcoeficient(), pol1.getpolinom().get(i).getexponent());  
 this.polinom.add(x);  
 i++;  
 j++; } }  
 while (i < pol1.getpolinom().size()) {  
 this.polinom.add(pol1.getpolinom().get(i));  
 i++;  
 }  
 while (j < pol2.getpolinom().size()) {  
 this.polinom.add(pol2.getpolinom().get(j));  
 j++;  
 }  
}

Algoritmul pentru suma de polinoame este bazat pe interclasare. Nu a fost implemenatata o metoda de sortare de aceea suma va functiona doar daca polinomul va fi introdus in ordine descrescatoare a exponentului. Se verifica daca cel de-al doilea polinom are gradul mai mare decat primul polinom. Daca din if va iesi valoarea ’1’ atunci primul element din al doilea polinom va fi adaugat in polinomul final. Daca if ul va ave valoarea ’0’ atunci se va verifica daca gradul polinomului 1 este mai mare decat gradul polinomului 2. Daca cel de-al doilea if va alea valoarea ’1’ se va adauga in polinomul final primul monom al polinomului 1. Daca si cel de-al doilea if va avea valorea ’0’ atunci se va ajunge la i ful 3 care verifica daca polinoameleau grade egale si se vor insuma fiind adauga monoamele in polinomul final. Daca polinoamel noastre vor contine mai multe moname ,cu ajutorul structurilor while se vor de la finalul metodei, elementele ce nnu au fot parcurse vor fi aduagate ulterior.

public void scaderepolinom(polinom pol1,polinom pol2)  
{  
 for(monom i : pol2.getpolinom())  
 {  
 i.negare();  
 }  
 this.adunarepolinoame(pol1,pol2);  
}

Metoda de scadere de polinoame are la baza suma de polinoame. Diferenta o reprezinta negarea celui de-al doilea polinom. La final vom avea suma dintre polinomul 1 si cel de-al doilea inmultit cu „1”.

for (monom i : pol1.getpolinom())  
{  
 for(monom j : pol2.getpolinom())  
 {  
 monom m= new monom(1,0);  
 m.inmultiremonoame(i);  
 m.inmultiremonoame(j);  
 this.polinom.add(m);  
 }

for(int i=0;i<this.polinom.size();i++)  
{  
 for(int j=i+1;j<this.polinom.size();j++)  
 {  
 if(this.polinom.get(i).getexponent()==this.polinom.get(j).getexponent())  
 {  
 this.getpolinom().get(i).adunaremonoame(this.getpolinom().get(j));  
 this.getpolinom().remove(j);  
  
  
 }  
 }  
}

}

Metoda folosita pentru inmutire va parcurge fiecare polinom si va inmulti pe rand fiecare monom din polinomul 1 cu fieacre monom din polinomul 2. La final avem cele doua for-uri care vor aduna elementele cu acelasi exponent.

public void derivarepolinoame()  
{  
 for(monom i: this.getpolinom())  
 {  
 i.derivaremonom();  
 }  
}

Metoda de derivare polinom parcurge polinomul si apeleaza functia de derivare pentru fiecare monom.

public void integrarepolinom()  
{  
 for(monom i : this.getpolinom())  
 {  
 i.integraremonom();  
 }

Metoda de integrare monoame lucreza pe aceeasi idee la fel ca cea de derivare.

4.3.Clasa View

this.setBounds(500,500,650,589);  
this.setVisible(true);  
this.setTitle("Calculator de polinoame");  
this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
panel.setBounds(0,0,600,600);  
this.add(panel);  
panel.setLayout(null);

Acest frame contine un singur panel. Layout ul l-am setat „null” pentru a putea aseza elementele grafice asa cum am dorit. Aceasta clasa are declararile butoanelor, textfiled-urile, label-urile. M-am ocupat in aceasta clasa de asezarea lor in panel.

coeficient1lbl.setBounds(30,10,80,20);  
panel.add(coeficient1lbl);

Acesta este un exemplu de asezare pentru label-ul coeficientul primului polinom. L-am setat pe coordonatele de axe x, y dupa care am setat marimea label-ului.

4.4.Clasa Controller

Aici s-au declarat elementele modelului ,in cazul nostru clasa polinom, si ale vederii(view).

private view view;  
private polinom polinom1 = new polinom();  
private polinom polinom2 = new polinom();  
private polinom polinomf = new polinom();  
private monom monom1 = new monom();  
private monom monom2 = new monom();

Am declarat view-ul, polinoamele pe care le vom folosi si monoamele care fac parte din primele doua polinoame.

Totodata in aceasta clasa am adaugat butoanele care vor fi folosite pentru utilizarea operatiilor dintre polinoame. Pentru fiecare buton in parte am creat o clasa interna pentru gestionarea ascultatorilor.

class adaugalistener implements ActionListener{  
 public void actionPerformed(ActionEvent e)  
 {  
 int coeficient =0;  
 int exponent=0;  
 try{  
 coeficient= Integer.*parseInt*(view.getcoef1tf().getText());  
 exponent= Integer.*parseInt*(view.getexp1tf().getText());  
 monom mon= new monom(coeficient,exponent);  
 polinom1.adaugaelementinpolinom(mon);  
 view.getpolinom1tf().setText(polinom1.printarepolinom());  
 view.getcoef1tf().setText("");  
 view.getexp1tf().setText("");  
 }catch(NumberFormatException n) {  
 JOptionPane.*showMessageDialog*(view.getpanel(),"Nu e bine!");  
 } }}

In exemplul de mai sus este asculatorul pentru buton „adauga ” destinat primului polinom. Acest buton ajuta la crearea treptata a primului polinom, adaugand pe rand cate un monom in primul polinom. Variabilele „coeficient” si „exponent” vor prelua valorile introduse in textfiled-urile corespunzatoare , vor crea un monom ,acesta urmand sa fie adaugat in polinom in momentul apasarii butonului „adauga”. Polinomul trebuie introdus ordine descrescatoare .Daca nu vor fi introduse valori pentru coeficient si exponent si ulterior va fi apasat butonul „adauga” va aparea o exceptie. Pentru butonul doi de adaugare clasa este identica.

class sumalistener implements ActionListener{  
 public void actionPerformed(ActionEvent e)  
 { view.getpolinomfinaltf().setText("");  
 polinomf= new polinom() ;  
 polinomf.adunarepolinoame(polinom1,polinom2);  
 view.getpolinomfinaltf().setText(polinomf.printarepolinom());  
 }}

Aceasta clasa implemenetaza ascultatorul pentru butonul de suma .Aici se apeleaza functia de adunare polinoame din clasa Polinom si se va face suma celor doua polinoame adaugate initial la apasarea butonuli suma.

Pentru scadere si inmultire asculatorii sunt asemanatori; difera doar functia apelata in interior.

Asculatorii pentru butoanele de derivare si integrare difera putin deoarece nu am mai lucrat cu un polinom rezultat. In interiorul asculatorului functiile de derivare si integrre vor fi apelate direct pe polinomul introdus, nefiind nevoie de un polinom rezultat in care sa introducem operatia.

class derivarelistener implements ActionListener{  
 public void actionPerformed(ActionEvent e)  
 {view.getpolinomfinaltf().setText("");  
 polinom1.derivarepolinoame();  
 view.getpolinomfinaltf().setText(polinom1.printarepolinom());  
 }}

Uramatorul exemplu il reprezinta ascultatorul pentru butonul clear. In momentul in care acest buton va fi apasat se vor elibera toate textfiled-rile ,iar polinoamele elemnetele din polinoame vor fi sterse. Astfel vom putea introduce noi polinoame pentru a putea folosi alte operatii.

class clearlistener implements ActionListener{  
public void actionPerformed(ActionEvent e)  
{ view.getpolinomfinaltf().setText("");  
 view.getpolinom1tf().setText("");  
 view.getpolinom2tf().setText("");  
 polinom1.clear();  
 polinom2.clear();  
 polinomf.clear();  
}}

**5.Testare**

Am creat o clasa numita „testpolinom” destinata testarii operatiile implementate pentru polinoame. In acesta clasa am declarat 4 polinoame : pol1,pol2-acestea vor fi polinoamele carora le vom aplica operatiile ;polr-reprezinta polinomul rezultat dupa aplicarea operatiilor pe cele doua polinoame initiale; polrc-reprezinta un polinom pe care l-am declarat si initalizat cu valorile corecte pentru a putea verifica daca rezultatul va fi egal cu polinomul corect.

public class testpolinom {  
  
 polinom pol1;  
 polinom pol2;  
 polinom polr;//polinom rezultat  
 polinom polrc;//polinom rezultat corect

Pentru a putea exemplifica o testare am pus codul de la testarea adunarii.

//(3x^4+3x^3-2x^2+1x^1)  
pol1.testadaugare(3,4);  
pol1.testadaugare(3,3);  
pol1.testadaugare(-2,2);  
pol1.testadaugare(1,1);  
  
//(2x^3+3x^2+2x^1)  
pol2.testadaugare(2,3);  
pol2.testadaugare(3,2);  
pol2.testadaugare(2,1);  
  
//(3x^4+5x^3+1x^2+3x^1)  
polrc.testadaugare(3,4);  
polrc.testadaugare(5,3);  
polrc.testadaugare(1,2);  
polrc.testadaugare(3,1);  
  
polr.adunarepolinoame(pol1,pol2);  
Assertions.*assertEquals*(polr.printarepolinom(),polrc.printarepolinom());

Am initalizat pol1 si pol2 cu valori date de mine dupa care am facut suma lor pe foaie si am verificat-o.Polrc a fost initalizat cu rezultatul corect pentru a putea face verificarea ulterior. In polr am salvat adunarea dintre cele doua polinoame. La final am verificat cu comanda”Assertion assertequals” daca cele doua doua polianome sunt egale ,iar dupa rulare s-a afisat”Test Passed”.

Pentru toate operatiile am efectuat acelasi proces de verficare,intre polinomul corect calculat si initializat de catre mine si polinomul rezultat dupa operatia celor doua polinoame initiale .La final ,la toate testele mi-a fost afisat acelasi mesaj: ”Test passed”.

6.Concluzii

In concluzie ,calculatorul de polinoame realizat indeplineste cerintele oferite ,cu exceptia operatiei de impartire intre polinoame. Am reusit sa implemntez operatia de impartire, am adaugat butonul in view ,dupa care in controller am pus ascultatorul,dar nu o pot face sa functioneze.

Imbunatatiri:

Ca si imbunatatiri ale codului pe viitor as adauga o functie care sorteaza polinomul exact dupa ce a fost introdus in Textfield. La momentul de fata polinoamele din programul meu trebuie introduse in ordine de catre utilizator pentru a putea rula fara nicio eroare. O fucntie care ar sorta polinomul exact dupa introducere ar face programul sa functioneze si in cazul unei introduceri aleatoare a monoamelor in polinom.

7.Bibliografie

- [www.stackoverflow.com](http://www.stackoverflow.com)

- algs4.cs.princeton