DOCUMENTATIE

TEMA 1

NUME STUDENT: Cacaraza Tobias Daniel

GRUPA: 30222

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 4](#_Toc95297887)

[4. Implementare 4](#_Toc95297888)

[5. Rezultate 7](#_Toc95297889)

[6. Concluzii 7](#_Toc95297890)

[7. Bibliografie 7](#_Toc95297891)

# Obiectivul temei

Obiectivul principal al temei este de a proiecta și implementa o aplicație Java cu interfata grafica pentru adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea polinoamelor. In cadrul acestea teme, se va folosi programarea orientată pe obiecte, incapsularea și design-ul cu clase a obiectelor.

Obiective:

* Proiectarea claselor (Definirea claselor necesare pentru reprezentarea polinoamelor si a monomilor) : 2 Analiza problemei;
* Implementarea Map-ului (Folosirea unui Map pentru reprezentarea polinoamelor) :4. Implementare;
* Implementarea foreach (Folosirea buclei foreach in loc de bucla for pentru a itera prin polinoame) : 2. Implementare;
* Implementarea interfetei grafice (Proiectarea si implementarea interfetei grafice folosind Java Swing sau JavaFX) :3 Interfata;
* Implementarea operatiilor de adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea polinoamelor (implementarea metodelor necesare) : 2 si 4;
* Limitarea dimensiunilor claselor si a metodelor ( eficientizarea metodelor): 6;
* Respectarea conventiilor de denumire Java ( ex. testAdd, maxDegree…): 4.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

În continuare, voi prezenta un exemplu de cadrul de cerințe funcționale și non-funcționale, precum și câteva cazuri de utilizare sub forma unor diagrame și descrieri de use-case pentru o aplicație mobilă de livrare a produselor.

1. Cadru de cerințe funcționale:
   1. Utilizatorii sa poata aduna, scadea, imparti, inmulti, deriva, integra polinoame.
   2. Utilizatorii sa poata alege polinoamele dorite.

2. Cadru de cerințe non-funcționale:

a. Performanță ridicată, astfel încât timpul de răspuns să fie cât mai scurt posibil.

b. Interfață de utilizare intuitivă și ușor de navigat.

c. Compatibilitate cu diverse dispozitive mobile și sisteme de operare.

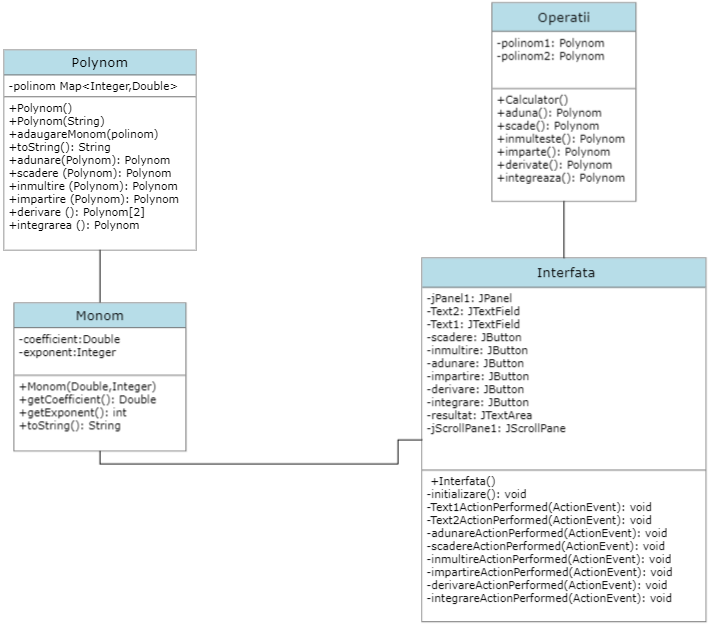
3. Cazuri de utilizare:

a. Matematica.

Modelare :

* Polynom : format din Map<Integer,Double>, formand mai multe monoame !
* Monoame: format din exponent (Integer) si coefficient (Double), cu ajutorul careia despartim in si adaugam in polynom.
* Operatii: metode pe care le folosim pe polinoame.
* Interfata: Pentru utilizator, intuitive si usor de navigat.
* Junit(OperatieTest): Testele metodelor de adunare, scadere, impartire, imnultire, derivare si integrare.

# Proiectare

---🡪 Main() ->>> +main(String[]): void ) ---🡪 Interfata()

# Implementare

public Polynom[] impartire(Polynom p1, Polynom p2)  
{  
 Operatii op = new Operatii();  
 //Polynom n = p1;  
 Polynom rest = new Polynom();  
 for(Map.Entry<Integer,Double> j : p1.polinom.entrySet())  
 {  
 rest.adaugaMonom(j.getKey(),j.getValue());  
  
 }  
 Polynom impartire = new Polynom();  
  
 while(rest.getMaxDegree() >= p2.getMaxDegree())  
 {  
 if(p2.getMaxDegree()<= rest.getMaxDegree()) {  
 Monom m1 = new Monom();  
  
 m1 = new Monom(rest.getMaxDegree() - p2.getMaxDegree(), rest.polinom.get(rest.getMaxDegree()) / p2.polinom.get(p2.getMaxDegree()));  
  
 impartire.adaugaMonom(m1);  
 Polynom s = op.inmultire(p2, new Polynom(m1));  
 rest = op.scadere(rest, s);  
  
 }  
  
 }  
 Polynom[] p = new Polynom[2];  
 p[0] = impartire;  
 p[1] = rest;  
 return p;  
}

Metoda de impartire ↑:

* metoda face o copie numita “rest”(pentru a face copia am folosit implementarea foreach) a deimpartitului apoi in bucla while se cauta un polinom cu care daca se inmulteste se scapa de cel mai mare grad;
* dupa gasire se adauga monomul la cat numit “impartire” si apoi se inmulteste ca sa se scada cu “restul”;
* aceasta metoda se repeta pana restul are grad mai mic ca impartitorul.

public Polynom adunare(Polynom p1, Polynom p2)  
{  
 Polynom suma = new Polynom();  
 for(Map.Entry<Integer,Double> i : p1.polinom.entrySet())  
 {  
  
 Integer ex = i.getKey();  
 Double c = i.getValue();  
 if(p2.polinom.containsKey(ex)) {  
 c = c + p2.polinom.get(ex);  
 p2.polinom.remove(ex);  
 }  
 suma.adaugaMonom( ex,c);  
 }  
 for(Map.Entry<Integer,Double> i : p2.polinom.entrySet())  
 {  
  
 Integer ex = i.getKey();  
 Double c = p2.polinom.get(ex);  
 suma.adaugaMonom( ex,c);  
 }  
 return suma;  
}

Metoda de adunare si scadere foloseste acelasi concept ↑:

* metoda foloseste 2 foreach-uri cu conditii pentru a aduna/scadea doar monoamele cu acelasi exponent.

public Polynom inmultire(Polynom p1, Polynom p2)  
{  
 Polynom inmultire = new Polynom();  
 for(Map.Entry<Integer,Double> i : p1.polinom.entrySet())  
 {  
  
 for(Map.Entry<Integer,Double> j : p2.polinom.entrySet())  
 {  
  
 Integer ex = i.getKey()+ j.getKey();  
 Double c = i.getValue()\* j.getValue();  
 if(!inmultire.polinom.containsKey(ex))  
 inmultire.adaugaMonom( ex,c);  
 else  
 {  
 Polynom add = new Polynom();  
 add.polinom.put(ex,c);  
 Operatii op = new Operatii();  
 inmultire = op.adunare(inmultire,add);  
 }  
 }  
 }  
  
 return inmultire;  
}

Metoda de inmultire e aproape aceeasi dar foloseste si operatia de adunare pentru monoamele rezultate cu acelasi exponent ↑.

private void inmultireActionPerformed(java.awt.event.ActionEvent evt) {  
 Pattern pattern = Pattern.*compile*("(\\+?-?[0-9]+)x\\^([0-9]+)");  
  
 Polynom t = new Polynom();  
 Polynom t2 = new Polynom();  
 Matcher matcher = pattern.matcher(Text1.getText());  
 while(matcher.find()) {  
 String c = matcher.group(1);  
 String c2 = matcher.group(2);  
 if(!c.isEmpty() && !c2.isEmpty())  
 t.adaugaMonom(Integer.*parseInt*(c2),Double.*parseDouble*(c));  
 }  
 Pattern pattern2 = Pattern.*compile*("(\\+?-?[0-9]+)x\\^([0-9]+)");  
  
 Matcher matcher2 = pattern2.matcher(Text2.getText());  
 while(matcher2.find()) {  
 String c = matcher2.group(1);  
 String c2 = matcher2.group(2);  
 if(!c.isEmpty() && !c2.isEmpty())  
 t2.adaugaMonom(Integer.*parseInt*(c2),Double.*parseDouble*(c));  
 }  
 Operatii op = new Operatii();  
 resultat.setText(op.inmultire(t,t2).toString());  
}

Metoda pentru butonul de inmultire din ↑:

* la evenimentul apasarii butonului se fac 2 polinoame din Stringul celor doua texte din interfata de utilizator;
* sunt doua grupuri care sunt luate de stringurile “c” si “c2” apoi sunt transformate in Integer si Double;
* apoi este afisat rezultatul operatiiei in “resultat”.

Metoda de derivare foloseste primul panel!! Si metoda de integrare foloseste al doilea panel!

# Rezultate

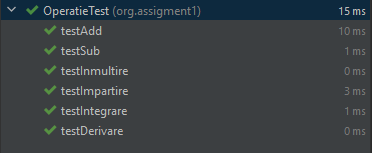
TestAdd – am adunat polinoamele “3x^2+1x^0” + “3x^1+4x^0” care ar trebuii sa returneze “6x^2+5x^0”.

TestSub – am scazut polinoamele “3x^2+1x^0” - “2x^2+3x^1” care ar trebuii sa returneze “1x^0 -3x^1+1x^2”.

TestInmultire – am inmultit polinoamele “3x^2+1x^0” \* “3x^1+4x^0” care ar trebuii sa returneze “6x^4+9x^3+2x^2+3x^1”.

TestImpartire– am impartit polinoamele “6x^5+4x^3+2x^2” \* “6x^2+3x^1” care ar trebuii sa returneze “1x^3-0.5x^2+11/12x^1-1/8x^3” cu restul “0.375x^1”.

TestDerivare– am derivat polinomul “6x^5+4x^3+2x^2” care ar trebuii sa returneze “30x^4+12x^2+4x^1”.

TestIntegrare– am integrat polinomul “30x^4+12x^2+4x^1” care ar trebuii sa returneze “6x^5+4x^3+2x^2”.

# Concluzii

Pentru minimizarea codului, a trebuit sa gandesc mai efficient la metode si probleme!

# Bibliografie

1. *What are Java classes? -* [*www.tutorialspoint.com*](http://www.tutorialspoint.com)
2. [*https://cloud.smartdraw.com/editor.aspx?templateId=d1553a61-d98e-4aa1-a5bc-11a11876ea36&flags=128#depoId=43858984&credID=-46734437*](https://cloud.smartdraw.com/editor.aspx?templateId=d1553a61-d98e-4aa1-a5bc-11a11876ea36&flags=128#depoId=43858984&credID=-46734437)
3. *https://dsrl.eu/courses/pt/*