# Proiectul 1
# CALCULATOR POLINOMIAL

Alexuc Tudor
Grupa 30221-1

# 1. Obiectivul temei

**Cerinţa :**
Proiectati si implementati un calculator de polinoame cu interfaţa grafica în care utilizatorul poate introduce polinoame , alege operaţia pe care vrea sa o efectueze (adunare , scadere , inmultire , impartire , derivare sau integrare ) şi să obţină rezultatul.

**Obiective secundare :**

1. Realizarea functionalitatii calculatorului - implementarea funcţiilor necesare
    a. Adunarea polinoamelor
    b. Scaderea polinoamelor
    c. Înmulţirea polinoamelor
    d. Derivarea
    e. Integrarea
    f. Împărţirea polinoamelor
2. Realizarea interfeţei grafice - crearea structurii şi a butoanelor corespunzătoare funcţiilor
3. Realizarea controller-ului din structura MVC - Crearea puntii dintre interfaţa grafica şi functionalitate
4. Testare - Utilizarea calculatorului pe diferite cazuri problematice
5. Rezolvarea diferitelor probleme apărute - Rezolvarea bugurilor
6. Realizarea documentaţiei

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

**Analiza problemei**

Cu scopul de a rezolva problema, am analizat mai multe aspecte, unul dintre ele fiind reprezentat de cunostiintele de matematica pe care acest subiect le implica. Astfel, am început prin a mă documenta legat de realizarea operaţiilor pe polinoame pentru a asigura un flow simplu funcţiilor implementate.

În matematică, un polinom este o expresie construită dintr-una sau mai multe variabile şi constante, folosind doar operaţii de adunare, scădere, înmulţire şi ridicare la putere constantă pozitivă întreagă.
Astfel am creat doua clase , clasa Monomial şi clasa Polynomial pentru a putea lucra pe monoame, iar in consecinţă, funcţiile sunt implementate mai simplu şi sunt mai uşor de înţeles şi urmărit.
Un monom poate fi definit ca o expresie algebrică reprezentand produsul dintre un coeficient şi o variabila ridicată sau nu la putere. Un polinom este format din mai multe monoame.

**Descrierea funcţiilor**

**Adunarea** a două polinoame se face insumand coeficientii polinoamelor de acelaşi grad şi pastrandu-se partea literara. Operaţiile aritmetice de adunare şi scădere a polinoamelor presupun adunarea şi scaderea coeficientilor de acelaşi ordin.

**Înmulţirea** se face prin produsul fiecărui monom al primului polinom, cu fiecare monom al celui de-al doilea polinom.

**Derivarea** şi **Integrarea** folosesc regulile matematice asupra monoamelor .

**Modelarea problemei**

După dobândirea unor cunoştinţe teoretice in legatura cu tema propusă am creat un proiect, structurat in 5 clase principale( Monomial , Polynomial , CalcView , CalcModel , CalcController ) cu alte două clase suplimentare (Main şi CompareRuleExponent).

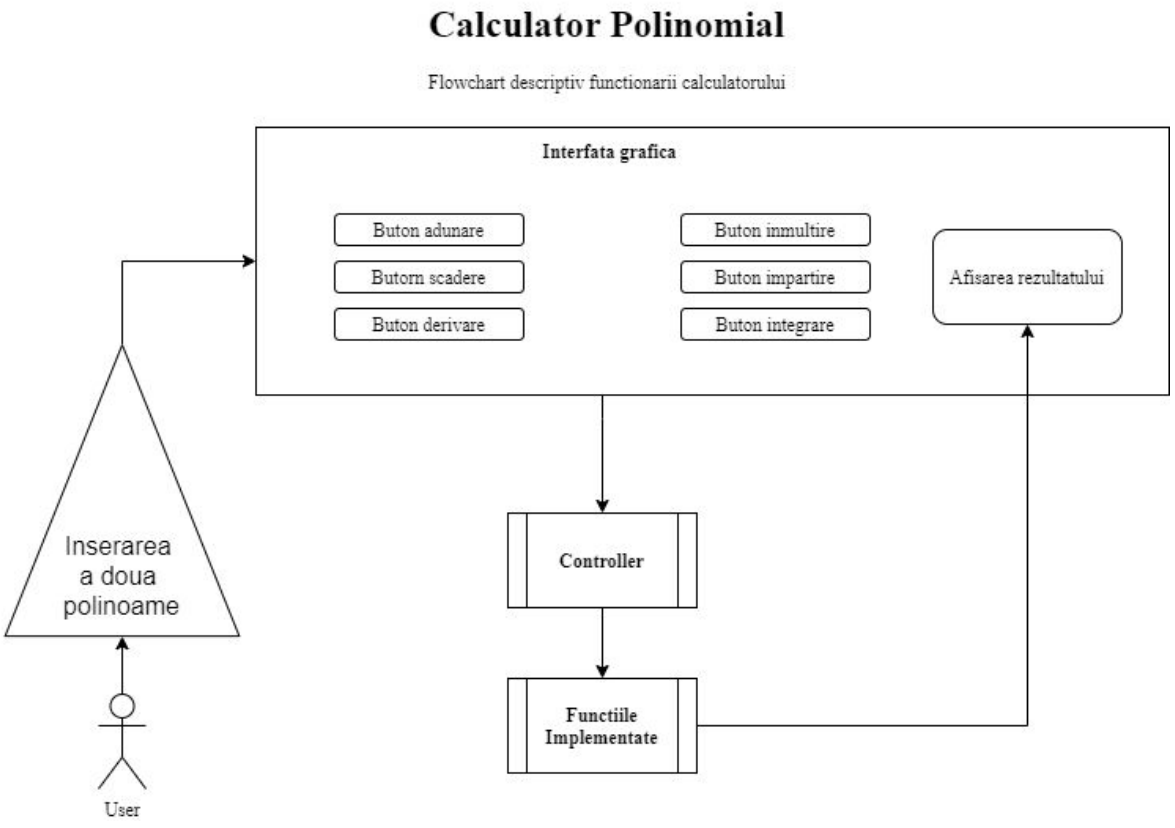
Clasele **Monomial** şi **Polynomial** realizează structura de polinom şi funcţiile cerute .

Clasa **CalcView** realizeaza interfata GUI Swing, care preia gradele si coeficientii a doua polinoame si cu ajutorul butoanelor de operaţii, se alege operaţia dorită de utilizator şi returnează rezultatul.

Clasa **CalcModel** se ocupă de datele de intrare în calculator iar **CalcController** face legătura dintre aceste date , interfaţa grafica şi functionalitatea implementată.

Clasa **Main** are scopul de a rula programul şi foloseşte toate clasele menţionate mai sus după structura MVC, iar clasa **CompareRuleExponent** am folosit-o pentru a crea o funcţie de aranjare a monoamelor în cadrul polinomului, astfel incat sa fie mereu în ordinea descrescătoare a gradelor .

**Scenarii şi cazuri de utilizare**



| Fluxul de evenimente | |
|---|---|
| 1. Utilizatorul introduce gradul si coeficientii ambelor polinoame | 2. Aplicatia preia datele |
| 3. Utilizatorul alege operatia dorita utilizand butoanele | 4. Aplicatia executa operatia<br>5. Aplicatia afiseaza rezultatul in campul corespunzator |
| 6. Utilizatorul apasa butonul "clear" pentru a face loc unei alte operatii | 7. Aplicatia sterge continutul campului corespunzator rezultatului. |

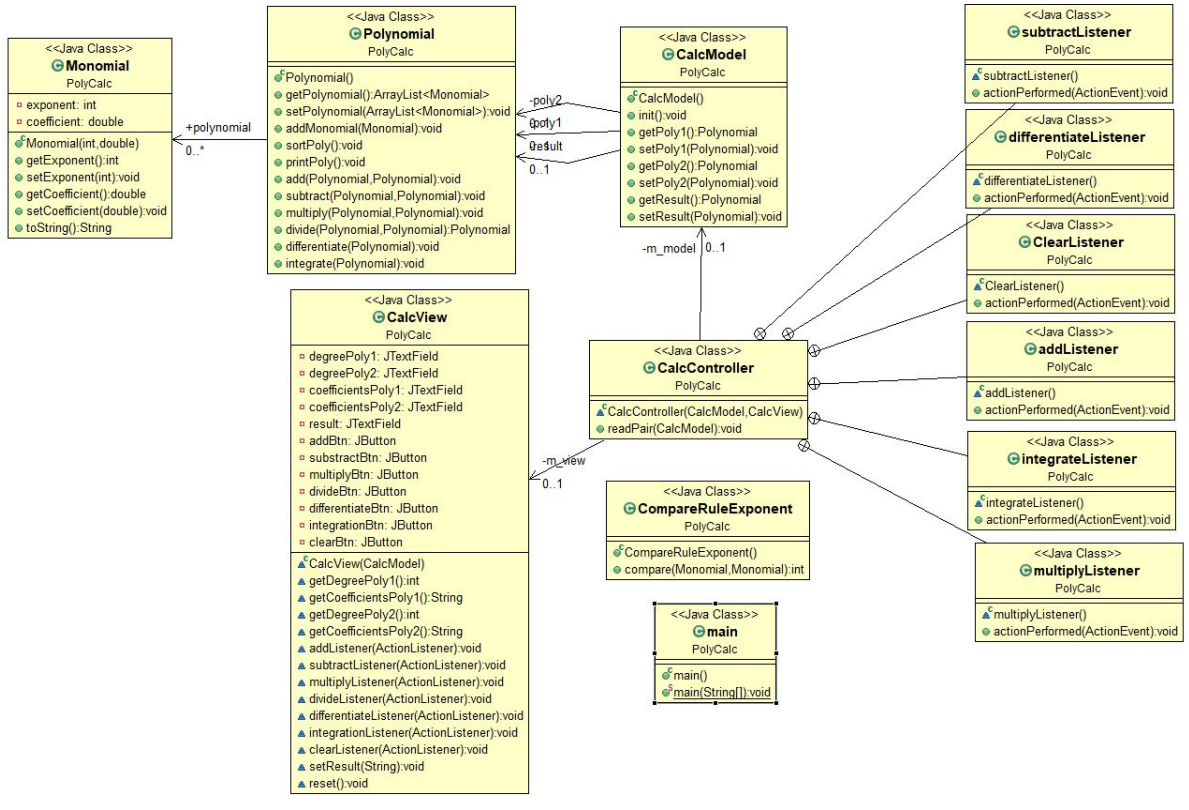
Aplicaţia este folosită de un utilizator după modul descris în fluxul de evenimente şi flowchartul de mai sus . Astfel, utilizatorul va introduce gradul şi o serie de coeficienti a polinomului si va selecta funcţia dorită iar calculatorul va afişa rezultatul . Utilizatorul are opţiunea de a şterge rezultatul şi de a introduce alte polinoame sau a alege alta functie .

# 3.Proiectare

### Decizii de proiectare

În afara de structura MVC şi clasele Monomial şi Polynomial care sunt necesare pentru funcţionarea calculatorului, am decis sa creez si o funcţie de sortare a polinoamelor (SortPoly) descrisă anterior. Altă decizie de proiectare a fost crearea funcţiilor în cadrul clasei Polynomial in loc de CalcModel pentru a o putea folosi şi în viitoare proiecte ce necesita funcţii polinomiale .

### Diagrama claselor



## Pachete

Un pachet Java este un mecanism de organizare a claselor. Programatorii mai folosesc pachetele cu scopul de a organiza clasele de aceeaşi categorie sau cu aceeaşi funcţionalitate. Sunt folosite pentru găsirea şi utilizarea mai uşoară a claselor, pentru a evita conflictele de nume şi pentru a controla accesul la anumite clase. În alte limbaje de programare, pachetele se numesc librării.

Pachete folosite :

java.util - clase şi interfeţe utile , a fost folosit pentru crearea funcţiilor polinomiale

java.awt - pentru interfata grafica cu utilizatorul , în special evenimentele realizate de utilizator

javax.swing - interfata grafica cu utilizatorul mult imbogatita fata de AWT.

În rest, calculatorul a fost realizat în cadrul pachetului PolyCalc.

**Clase**

## Monomial

### Atribute :

- exponent : int
- coefficient : double

### Constructori :

- Monomial(int exponent, double coefficient)

### Metode :

- getExponent() : int
- setExponent(int) : void
- getCoefficient() : double
- setCoefficient(double) : void
- toString() : String

## Polynomial

### Atribute :

- polynomial : ArrayList<Monomial>

### Constructori :

- Polynomial()

### Metode :

- getPolynomial() : ArrayList<Monomial>
- setPolynomial(ArrayList<Monomial>) : void
- addMonomial(Monomial) : void
- sortPoly() : void
- printPoly() : void
- add(Polynomial,Polynomial) : void
- subtract(Polynomial,Polynomial) : void
- multiply(Polynomial,Polynomial) : void
- differentiate(Polynomial) : void
- integrate(Polynomial) : void

## CalcView

### Atribute :

- degreePoly1 : JTextField
- degreePoly2 : JTextField
- coefficientsPoly1 : JTextField
- coefficientsPoly2 : JTextField
- result : JTextField
- addBtn : JButton
- substractBtn : JButton
- multiplyBtn : JButton
- divideBtn : JButton
- differentiateBtn : JButton
- integrationBtn : JButton
- clearBtn : JButton

### Constructori :

- CalcView(CalcModel)

### Metode :

- getDegreePoly1() : String
- getCoefficientsPoly1() : String
- getDegreePoly2() : String
- getCoefficientsPoly2() : String
- + Listeners for the buttons

## CalcModel

### Atribute :

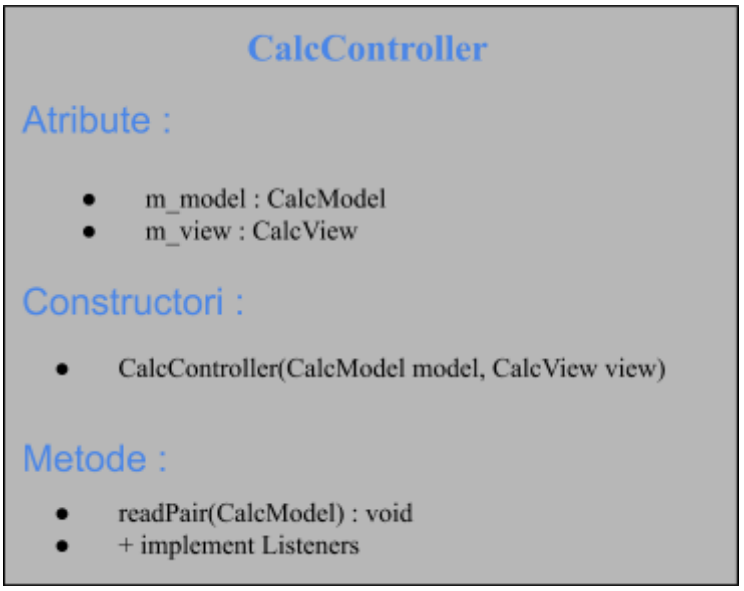
- poly1 : Polynomial
- poly2 : Polynomial
- result : Polynomial

### Constructori :

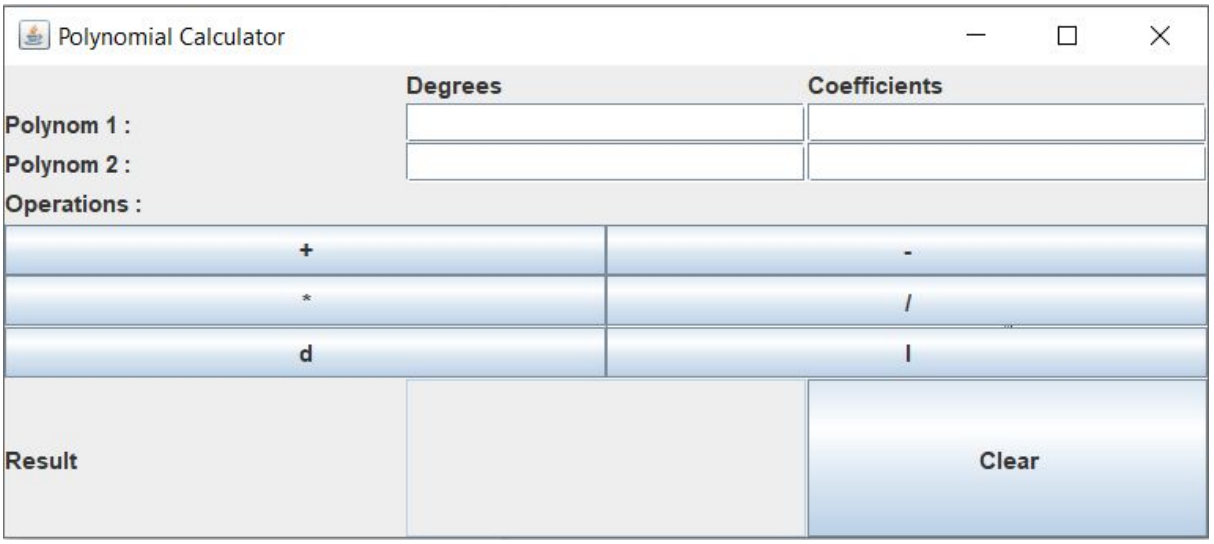
- CalcModel()

### Metode :

- init() : void
- getPoly1() : Polynomial
- setPoly1(Polynomial) : void
- getPoly2() : Polynomial
- setPoly2(Polynomial) : void
- getResult() : Polynomial
- setResult(Polynomial) : void




### Interfaţa




Interfaţa conţine JTextFields pentru introducerea gradelor şi a coeficienţilor(“x1 x2 x3 x4 …” separate prin ‘ ‘) polinoamelor şi butoane pentru cele 6 funcţii cerute . De asemenea, la final observam locul de aflare a rezultatului şi butonul de clear .

# 4.Implementare

### Polynomial class + Monomial class

Clasa Polynomial contine toate funcţiile polinomiale şi majoritatea părţii funcţionale a programului . Câteva funcţii suplimentare sunt funcţiile de afişare şi de sortare a polinoamelor . De asemenea , clasa Polynomial descrie cu ajutorul clasei Monomial modul de funcţionare al polinoamelor în sine şi forma pe care trebuie ca acestea sa o aiba .

**Funcţia de sortare sortPoly :**

```
public void sortPoly()
	{
		Collections.sort(this.polynomial, new CompareRuleExponent());
		this.polynomial.removeIf(m -> (m.getCoefficient() == 0.0));
		if (this.polynomial.isEmpty() == true)
		{
			this.polynomial.add(new Monomial(0, 0.0));
		}
	}
```

Funcţia sort poly va organiza polinomul în forma canonica şi este folosită la sfarsitul fiecărei funcţii (adunare , scadere , inmultire , impartire , derivare , integrare) pentru a asigura rezultatul în forma corecta . În acelaşi timp funcţia va afişa “0” şi nu “0.0” în cazul în care polinomul este nul . Funcţia se foloseşte de clasa **CompareRuleExponent** care foloseşte interfata Comparator şi defineste regula de comparare în funcţie de exponentele monoamelor polinomului .

```
public class CompareRuleExponent implements Comparator<Monomial>
{
	@Override
	public int compare(Monomial o1, Monomial o2)
	{
		return o2.getExponent() - o1.getExponent();
	}
}
```

**Funcţia de printare printPoly :**

```
public String printPoly()
	{
	    String print = "";
	    if (this.polynomial.get(0).getCoefficient() == 0.0 && this.polynomial.get(0).getExponent()==0)
		{
			print = "0.0";
		} else
		{
			boolean f = true;
			for (Monomial monomial : this.polynomial)
			{
				if (f == true)
				{
					if (monomial.getCoefficient() > 0.0)
					{
						print = print + monomial.toString();
					} else
					{
						print = print + monomial.toString();
					}
					f = false;
				} else
				{
					if (monomial.getCoefficient() > 0.0)
					{
						print = print + "+" + monomial.toString();
					} else
					{
						print = print + monomial.toString();
					}
				}
			}
		}
	return print;
}
```

Aceasta functie foloseşte o metoda rescrisă “toString” pentru monoame :

```
public String toString()
	{
		if (exponent == 0)
		{
			return Double.toString(coefficient);
		}
		else if (exponent > 1)
		{
			if (coefficient == 1.0)
			{
				return "x^" + exponent;
			}
			else if (coefficient == -1.0)
			{
				return "-x^" + exponent;
			}
			else
				return String.format("%.2f", coefficient) + "*x^" + exponent;
		}
		else
		{
			if (coefficient == 1.0)
			{
				return "x";
			}
			else if (coefficient == -1.0)
			{
				return "-x";
			}
			else
				return String.format("%.2f", coefficient) + "*x^";
		}
	}
```

Aceasta functie toString din Monomial se ocupă de afisarea coeficientilor şi a exponenţilor fiecărui monom .

Restul funcţiilor sunt cele 6 cerute în enunţul problemei .

**Adunarea** parcurge monoamele de la exponentul cel mai mic la cel mai mare şi aduna coeficientii monoamelor de acelaşi exponent şi apoi copiază restul monoamelor. **Scaderea** foloseşte funcţia de adunare şi inmulteste fiecare monom din cel de-al doilea polinom cu -1 .

**Înmulţirea** parcurge cele doua polinoame şi inmulteste fiecare monom cu fiecare , adunând exponenţii şi înmulţind coeficientii.

**Derivarea** se realizează prin parcurgerea monoamelor şi realizarea următorilor paşi : înmulţirea coeficientului cu exponentul şi apoi decrementarea exponentului .

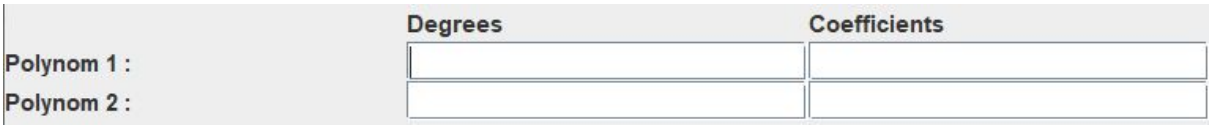
**Integrarea** reprezinta incrementarea exponentului fiecarui monom si impartirea coeficientului la exponentul iniţial.

### B.CalcView , CalcModel , CalcController

**CalcView , CalcModel , CalcController** sunt cele 3 clase necesare pentru structura MVC şi realizează funcţiile necesare structurii descrise şi la capitolul 2 la modelarea problemei . Singura diferenta fata de structura clasică este reprezentată de locul în care sunt implementate funcţiile , în programul meu acestea fiind prezente în clasa Polynomial şi nu în CalcModel.

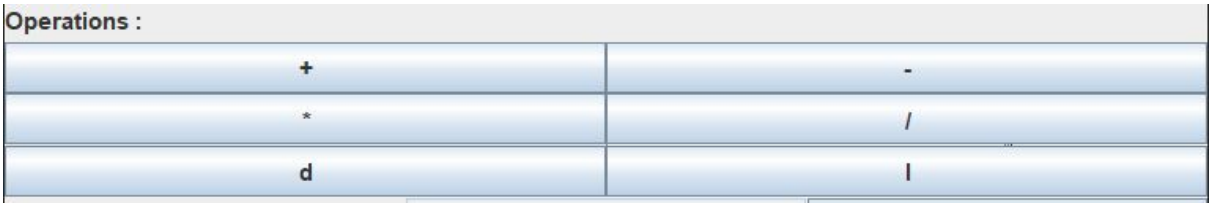
CalcView reprezinta clasa de descriere a interfetei si foloseşte 4 JPanels.

Primul reprezinta partea cu introducerea datelor :




Şi contine 4 JLabels (“Degrees” , “Coefficients”,”Polynom 1 :”,”Polynom 2 :”) şi 4 JTextFields pentru introducerea gradelor şi a coeficientilor polinoamelor .

Al doilea reprezinta partea cu butoanele si contine cele 6 JButtons pentru funcţii şi un JLabel (“Operations :”) :




Al treilea reprezinta linia cu rezultatul si contine un JLabel (“Result :”) un JTextField pentru afişarea rezultatului şi un JButton pentru clear .


Ultimul JPanel contine celelalte 3 JPanels sub forma de coloana

## 5.Rezultate

| **Grad 1 = Grad 2** | Grad | Coeficienţi |
|---|---|---|
| Polinom 1 | 3 | 1 2 3 4 |
| Polinom 2 | 3 | 4 3 2 1 |
| Adunare | 5.00*x^3+5.00*x^2+5.00*x+5.0 | |
| Scadere | 3.00*x^3+x^2-x-3.0 | |
| Înmulţire | | |
| Derivare (Polinom 1) | 12.00*x^2+6.00*x+2.0 | |
| Integrare (Polinom 1) | x^4+x^3+x^2+x | |

| **Grad 1 > Grad 2** | Grad | Coeficienţi |
|---|---|---|
| Polinom 1 | 4 | 1 2 3 4 5 |
| Polinom 2 | 3 | 1 2 3 4 |
| Adunare | 5.00*x^4+8.00*x^3+6.00*x^2+4.00*x+2.0 | |
| Scadere | 5.00*x^4 | |
| Înmulţire | | |
| Derivare (Polinom 1) | 20.00*x^3+12.00*x^2+6.00*x+2.0 | |
| Integrare (Polinom 1) | x^5+x^4+x^3+x^2+x | |

| **Grad 1 < Grad 2** | Grad | Coeficienţi |
|---|---|---|
| Polinom 1 | 3 | 2 3 4 5 |
| Polinom 2 | 4 | 1 2 3 4 5 |
| Adunare | 5.00*x^4+9.00*x^3+7.00*x^2+5.00*x+3.0 | |
| Scadere | -5.00*x^4+x^3+x^2+x+1.0 | |
| Înmulţire | | |
| Derivare (Polinom 1) | 15.00*x^2+8.00*x+3.0 | |
| Integrare (Polinom 1) | 1.25*x^4+1.33*x^3+1.50*x^2+2.00*x | |

Programul a fost testat pe mai multe polinoame, astfel că s-au verificat anumite condiţii pentru care operaţiile eşuează şi s-au remediat problemele . În mare problemele au fost la afişarea rezultatului , programul scriind polinoamele greşit în memorie .

O alta problema recurenta a fost legată de modul de citire din moment ce funcţiile lucrează pe monoame a trebuit sa gandesc moduri de funcţionare a acestora ca polinoame de sine stătătoare .

# 6.Concluzii şi dezvoltări ulterioare

Pentru realizarea proiectului a fost nevoie de cunoştinţe matematice, dar nu mai mult decat de proprietăţi ale polinoamelor. Atenţia la detalii şi condiţii este importantă, mai ales la modul în care citim sau scriem un polinom , în acelaşi timp acest lucru poate fi îmbunătăţit prin schimbarea din citirea cu grad şi coeficienti in citirea directă a polinoamelor scrise în forma matematica cu ajutorul funcţiei regex . Fiecare metodă a ridicat un anumit nivel de dificultate şi nu am găsit o soluţie optimă la împărţirea polinoamelor. Analiza, proiectarea şi modelarea au fost etape importante pentru a realiza cu succes produsul final.

Aceasta aplicatie poate fi mult dezvoltată, în primul rand prin adaugarea tuturor operaţiilor ce se pot face cu polinoame (scoaterea rădăcinilor, împărţire etc) dar şi prin crearea excepţiilor care îndruma utilizatorul, pentru a introduce date corecte. De asemenea, partea grafica poate fi mult imbunatatita pentru un aspect modern sau mai elegant.

De asemenea prin decizia de a implementa funcţiile în clasa Polynomial , aceasta poate fi folosită ca o clasa de sine stătătoare în alte proiecte cum ar fi un calculator stiintific sau alte aplicaţii de manipulare a polinoamelor .

# 7. Bibliografie


http://stackoverflow.com/

https://ro.wikipedia.org/wiki/Polinom

https://docs.oracle.com/javase/tutorial/

https://www.objectaid.com/class-diagram