



**Facultatea de Automatica și Calculatoare**

## **Tema 1:**

# **Sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi**

**Disciplina: Tehnici de programare**

**Student:**

**Coman Vasile**

**An II**

**Grupa 30221**

**Profesor coordonator:**

**Antal Marcel**

**Profesor curs:**

**Ioan Salomie**



## **Cuprins**

<b>1. Obiectivul temei.....</b>	<b>3</b>
<b>2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori.....</b>	<b>3</b>
<b>3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor).....</b>	<b>4</b>
<b>4. Implementare.....</b>	<b>5</b>
<b>5. Testare.....</b>	<b>7</b>
<b>6. Rezultate.....</b>	<b>9</b>
<b>7. Concluzii.....</b>	<b>11</b>
<b>8. Bibliografie.....</b>	<b>11</b>



## 1. Obiectivul temei

Obiectivul principal al temei este proiectare și implementarea unui sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi.

Obiective secundare:

- construirea polinomului folosindu-se o listă de monoame;
- implementarea operațiilor de adunare, scădere, înmulțire și împărțire pentru două monoame, astfel se simplifică implementarea operațiilor pentru polinoame;
- transformarea polinomului dat sub forma unui string de către utilizator în monoame adică convertirea unui string în instanța unei clase, mai exact o listă de monoame;
- transformarea inversă prin care lista de monoame este afișată sub forma unui string utilizatorului ca și rezultat în urma operațiilor pe monoame și polinoame.

Toate aceste obiective încapsulate în același proiect duc la rezolvarea cerinței. Obiectivele vor fi detaliate în capitolele care descriu proiectarea și implementarea.

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

Forma generală a unui polinom:

$$P(X) = a_0X^n + a_1X^{(n-1)} + \dots + (a_{n-1})X + a_n, \text{ unde } a_0 \neq 0.$$

Proprietăți:

- suma/produsul/diferența a două polinoame este tot un polinom;
- derivata unui polinom este tot un polinom.

Cerințe și funcționalități:

- introducerea unui polinom;
- afișarea unui polinom;
- adunarea a două polinoame;
- diferența a două polinoame;
- înmulțirea a două polinoame;
- împărțirea a două polinoame;
- integrarea unui polinom;
- derivarea unui polinom.

Use-case

Avem un sistem în care un utilizator poate să introducă două polinoame și să efectueze pe aceste două polinoame operațiile amintite mai sus primind un rezultat specific operației selectate.

Asumptii făcute:

- se presupune că utilizatorul introduce un polinom corect sub forma menționată mai sus și nu introduce un polinom null care ar genera erori în cazul unor operații;
- tipul de date pentru coeficient este double pentru a facilita operația de integrare.



### 3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)

Am ales sa impart proiectul in trei clase:

- o clasa pentru a defini un monom prin coeficient si exponent(  $ax^b$ );
- o clasa pentru a defini un polinom printr-o lista de monoame;
- o clasa pentru implementarea interfetei grafice.

Ulterior am adaugat si o clasa de test in care am verificat unele operatii implementate pe polinoame.

Monom	Polinom	Gui
-coef: double -exp: int	-poli: ArrayList<Monom>	-pane: Jpanel -l1, l2 ,l3 ,l4: JLabel -tf1 ,tf2 ,tf3 ,tf4: JTextField -btn1, btn2, btn3, btn4, btn5 ,btn6, btn7, btn8: JButton - c1, c2: JcheckBox - p,q: Polinom
+sumaMonom(Monom): Monom +diferentaMonom(Monom): Monom +produsMonom(Monom): Monom +raportMonom(Monom): Monom +getCoef(): double +getExp(): int +compareTo(Monom): int +toString(): String	+sumaPolinom(Polinom): Polinom +diferentaPolinom(Polinom): Polinom +produsPolinom(Polinom): Polinom +raportPolinom(Polinom): Polinom -restrangePolinom(): void -grad(): int +derivarePolinom(): Polinom +integrarePolinom(): Polinom +parsarePolinom(String): void +afisarePolinom(): String -getPoli(): ArrayList<Monom> - setPoli(ArrayList<Monom>): void	+actionPerformed(ActionEvent): void



## 4. Implementare

Clasa Monom definește un monom de forma  $ax^b$  unde  $a$  este coeficientul și  $b$  este exponentul. Am implementat următoarele metode:

- metode pentru suma, diferența, raportul și produsul a două monoame care returnează un obiect de tip monom utilizând regulile din matematică (suma/diferența a două monoame de același grad este egală cu un monom de acel grad având coeficientul egal cu suma/diferența coeficienților, produsul a două monoame returnează un monom având gradul egal cu suma gradelor celor două monoame și coeficientul egal cu produsul coeficienților iar raportul a două monoame ne returnează un monom cu gradul egal cu diferența gradelor și coeficientul egal cu raportul coeficienților);
- am utilizat gettere pentru exponent și coeficient, variabilele instanță având modul de vizibilitate private;
- am suprascris metoda compareTo pentru a putea sorta lista de monoame descrescător, această metoda returnează -1 dacă gradul primului monom este mai mare decât celui de-al doilea, 1 pentru condiția inversă și 0 în caz de egalitate;
- am suprascris metoda toString pentru a putea lega între ele monoamele din lista de monoame astfel încât să aibă aspectul unui polinom, această metoda returnează un string de forma coef+ $x^{\text{exp}}$ ;
- avem doi constructori cel implicit care inițializează variabile instanță la valorile specifice fiecărui tip de dată și un constructor care primește coeficientul de tip double și puterea de tip int.

Clasa Polinom este formată dintr-o variabilă instanță de tip `ArrayList<Monom>` adică este compusă dintr-o listă de monoame.

Metode implementate:

- pentru suma a două obiecte de tip polinom, care sunt formate dintr-o listă de monoame, am parcurs cu foreach cele două liste de monoame iar unde găsim egalitate între grade efectuăm suma între cele două monoame și adăugăm rezultatul într-o nouă listă de monoame, apoi adăugăm în acea nouă listă monoamele din prima listă care au gradul diferit de toate monoamele din a doua listă și invers. Returnăm un nou polinom folosind un constructor care primește lista de monoame nou construită;
- pentru diferența ideea este aceeași cu excepția că monoamele din a doua listă care au gradul diferit de toate monoamele din prima listă le vom adăuga cu semn schimbat în lista construită;
- pentru produsul a două polinoame se parcurg cele două liste de monoame și se înmulțesc fiecare element din prima listă cu fiecare element din a doua listă utilizându-se operația de înmulțire a două monoame implementată în clasa Monom, monoamele rezultate se adăuga într-o listă care va fi transmisă ca și parametru pentru constructorul din clasa Polinom astfel se va construi un nou polinom fiind rezultatul înmulțirii celor două polinoame;



-pentru impartirea a doua monoame se utilizeaza algoritmul prezentat in cursul orelor de laborator, am adaugat cazul in care gradul deimpartitului este mai mic decat al impartitorului funcia sa returneze ca si cat 0 iar restul va fi impartitorul;

-metoda de parsare a unui polinom primeste polinomul dat sub forma unui string de catre utilizator si cu ajutorul functie replaceAll se editeaza textul astfel incat sa poata fi despartit in stringuri asemanatoare unui monom. Despartirea(split) se face dupa caracterul „+” iar apoi cu foreach se parcurg stringurile formate in urma operatie split. Astfel ca din stringurile formate putem construi o lista de monoame facand cast la double la ce este in fata lui „x” si facand cast la int la ce este dupa „^”. Astfel transmitem constructorului din clasa Monom un parametru de tip double care va reprezenta coeficientul si un parametru de tip int care va reprezenta exponentul. Constructorul va crea un obiect de tip monom care va si adauga in lista de monoame. Astfel parcurgandu-se fiecare string se va crea lista de monoame corespunzatoare polinomului dat de catre utilizator;

-metoda de restrangere a unui polinom este conceputa pentru ca in cazul in care utilizatorul introduce un polinom care are mai multe monoame de acelasi grad sau daca in urma unei operatii apar mai multe monoame de acelasi grad acestea sa fie adunate sub forma unui singur monom. Metoda sorteaza lista de monoame, folosind functia sort si metoda suprascrisa compareTo, si cat timp doua monoame vecine au acelasi grad le aduna intr-un monom apoi acest monom este adaugat intr-o noua lista de monoame iar la final polinomul este setat cu noua lista de monoame adica cu lista restransa;

-metoda de afisare polinom poate fi considerata echivalentul invers al metodei de parsare, astfel ca parcurgem lista de monoame al polinomului si fiecarui monom apelam metoda toString suprascrisa in clasa Monom si adaugam intr-un string prin concatenare fiecare monom transformat. La final folosim functia replaceAll pentru a edita stringul creat cu scopul de a genera o varianta cat mai fidela de afisare a polinomului.

Clasa Gui extinde clasa JFrame si implementeaza interfata ActionListener.

In clasa Gui am construit interfata cu utilizatorul care permite o comunicare usoara intre utilizator si program. Astfel ca utilizatorul va avea posibilitatea de a introduce doua polinoame intr-un textfield. Acesta trebuie sa selecteze butonul Citeste pentru ca polinomul dat sa fie trimis spre parsare. Dupa ce a introdus cele doua polinoame utilizatorul poate sa selecteze operatiile de adunare, scadere, inmultire si impartire. La impartire catul se va afisa intr-un textfield iar restul in alt textfield. Pentru integrare si derivare acesta dispune de un checkbox care permite alegerea polinomului care sa fie trimis spre operatia de integrare, respectiv derivare.

Constructorul din clasa Gui primeste ca parametru un string care va reprezenta numele frame-ului(ferestrei). In constructor se face apel catre constructorul din clasa predefinita JFrame transmitandu-se numele frame-ului folosind super(); De asemenea in constructor se pozitioneaza fiecare label, buton, textfield si checkbox si se adauga ascultatori unde este nevoie pentru a putea modela in metoda actionPerformed comportarea interfetei grafice.



Suprascrierea metodei `actionPerformed` ne permite sa controlam comportarea interfetei grafice cand utilizatorul interactioneaza cu aceasta.

Utilizatorul poate introduce polinoame care au mai multi termeni cu acelasi grad fara a afecta rezultatele operatiilor deoarece in momentul citirii polinomului programul va sorta descrescator in functie de grad si va restrange lista de monoame, la fel facand si inainte de a trimite spre afisare rezultatul;

Polinoame care au mai multe spatii intre termeni sau in loc de  $x$  au  $X$  nu afecteaza rezultatele. De asemenea au fost tratate cazurile cand utilizatorul introduce  $x$  sau  $-x$  sa se considere  $+1x^1$  respectiv  $-1x^1$ .

Singura conditie care trebuie indeplinita este ca polinomul sa fie in functie de  $x$  sau  $X$  si fiecare termen sa aiba un numar inainte de  $x$  si un numar dupa  $^$  cu exceptia cazurilor cand avem  $x$ ,  $-x$  sau termen liber.

Main-ul este implementat in clasa `Gui` unde este creat un obiect de tip `JFrame` care este instantiat ca si un obiect de tip `Gui` atribuindu-se astfel toate labelurile, butoanele si textfieldurile definite.

Librarii folosite:

-`java.awt.event`, `javax.swing` pentru implementarea interfetei grafice;  
 -`java.util` pentru utilizarea colectiilor;  
 -`org.junit.jupiter.api.Test` folosita pentru testarea rezultatelor unor metode implementate.

## 5. Testare

In acest proiect pentru clasa `Polinom` am creat un Junit Test Case pentru a verifica operatiile de adunare, scadere, inmultire si impartire implementate in aceasta clasa.

Tip test	Date intrare	Rezultat asteptat	Rezultat obtinut	Pass/fail
Adunare	Primul polinom= $2x^2+4x+3$ Al doilea polinom= $x+1$	$2.0x^2+5.0x+4.0$	$2.0x^2+5.0x+4.0$	Pass
Scadere	Primul polinom= $2x^2+4x+3$ Al doilea polinom= $x+1$	$2.0x^2+3.0x+2.0$	$2.0x^2+3.0x+2.0$	Pass
Inmultire	Primul polinom= $2x^2+4x+3$ Al doilea polinom= $x+1$	$2.0x^3+6.0x^2+7.0x+3.0$	$2.0x^3+6.0x^2+7.0x+3.0$	Pass
Raport	Primul polinom= $2x^2+4x+3$ Al doilea polinom= $x+1$	$2.0x+2.0$	$2.0x+2.0$	Pass



Printscreen-urile urmatoare arata ca testarea cu Junit a celor patru operatii s-a efectuat cu succes.

```

1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class OperatiiPolinomTest {
6     @Test
7     void testSumaPolinom() {
8         Polinom p=new Polinom();
9         Polinom q=new Polinom();
10        p.parsarePolinom("2x^2+4x+3");
11        q.parsarePolinom("x+1");
12        Polinom result=p.sumaPolinom(q);
13        assertEquals("2.0x^2+5.0x+4.0",result.afisarePolinom());
14    }
15    @Test
16    void testDiferentaPolinom() {
17        Polinom p=new Polinom();
18        Polinom q=new Polinom();
19        p.parsarePolinom("2x^2+4x+3");
20        q.parsarePolinom("x+1");
21        Polinom result=p.diferentaPolinom(q);
22        assertEquals("2.0x^2+3.0x+2.0",result.afisarePolinom());
23    }
24    @Test
25    void testProdusPolinom() {
26
27        @Test
28        void testProdusPolinom() {
29            Polinom p=new Polinom();
30            Polinom q=new Polinom();
31            p.parsarePolinom("2x^2+4x+3");
32            q.parsarePolinom("x+1");
33            Polinom result=p.produsPolinom(q);
34            assertEquals("2.0x^3+6.0x^2+7.0x+3.0",result.afisarePolinom());
35        }
36        @Test
37        void testRaportPolinom() {
38            Polinom p=new Polinom();
39            Polinom q=new Polinom();
40            p.parsarePolinom("2x^2+4x+3");
41            q.parsarePolinom("x+1");
42            Polinom result=p.raportPolinom(q);
43            assertEquals("2.0x+2.0",result.afisarePolinom());
44        }
45    }
46 }
  
```





## 6. Rezultate

Dupa ce utilizatorul a introdus cele doua polinoame si a selectat butoanele de citire pentru a trimite textul spre parsare, el are posibilitatea de a efectua urmatoarele operatii cu polinoamele citite:

a) Adunarea celor doua polinoame citite

The screenshot shows the 'Polinom' application window. It has a title bar with a document icon and the text 'Polinom'. The window contains four input fields on the left: 'Polinom 1' with the value  $2x^2-6x+10$ , 'Polinom 2' with the value  $x+2$ , 'Rezultat/Cat' with the value  $2.0x^2-5.0x+12.0$ , and 'Rest Impartire' which is empty. To the right of these fields are two buttons labeled 'Citeste', each with a small square checkbox to its right. Further right is a vertical toolbar with buttons for '+', '-', '\*', and '/'. At the bottom right are two larger buttons labeled 'Integrare' and 'Derivare'.

b) Scaderea celor doua polinoame citite

This screenshot is similar to the previous one, showing the 'Polinom' application window. The 'Polinom 1' and 'Polinom 2' fields remain the same. However, the 'Rezultat/Cat' field now displays  $2.0x^2-7.0x+8.0$ , which is the result of subtracting the second polynomial from the first. The 'Rest Impartire' field is still empty. The 'Citeste' buttons, toolbar, and 'Integrare'/'Derivare' buttons are also present in the same positions.



c) Inmultirea celor doua polinoame citite

Polinom					
Polinom 1	$2x^2-6x+10$	Citeste	<input type="checkbox"/>	+	
Polinom 2	$x+2$	Citeste	<input type="checkbox"/>	-	
Rezultat/Cat	$2.0x^3-2.0x^2-2.0x+20.0$			*	
Rest Impartire				/	
				Integrare	
				Derivare	

d) Impartirea celor doua polinoame citite si afisarea atat a catului cat si a restului

Polinom					
Polinom 1	$2x^2-6x+10$	Citeste	<input type="checkbox"/>	+	
Polinom 2	$x+2$	Citeste	<input type="checkbox"/>	-	
Rezultat/Cat	$2.0x-10.0$			*	
Rest Impartire	$30.0$			/	
				Integrare	
				Derivare	

e) Derivarea unde a fost ales primul polinom spre a fi derivat.

Polinom					
Polinom 1	$2x^2-6x+10$	Citeste	<input checked="" type="checkbox"/>	+	
Polinom 2	$x+2$	Citeste	<input type="checkbox"/>	-	
Rezultat/Cat	$4.0x-6.0$			*	
Rest Impartire				/	
				Integrare	
				Derivare	



f)Integrare aici a fost ales cel de-al doilea polinom spre a fi integrat.

## 7. Concluzii

În concluzie acest proiect implementează corect operațiile efectuate pe polinoame de orice ordin și oferă utilizatorului rezultatele dorite.

În urma rezolvării acestei teme am aprofundat și am fixat cunoștințele de programare orientată pe obiect învățate semestrul trecut. Am învățat cum să utilizez clasele de test pentru a-mi testa diferitele funcții pe care trebuie să le îndeplinească programul. Am învățat cum să încarc, să updatez, să dau share unui proiect de tip Maven folosind bitbucket. Am învățat să implementez o interfață grafică care poate fi utilizată ușor de orice tip de utilizator. Am învățat să scriu cod pentru un proiect funcțional pornind de la diagrama UML descrisă în cursul orelor de laborator.

De asemenea, am învățat cum să îmi gestionez timpul și resursele pentru a putea preda temele și proiectele la termenul de predare stabilit.

Acest proiect are o multime de posibilități de dezvoltare ulterioară:

- calcularea valorii polinomului într-un punct  $x$  dat de către utilizator;
- calcularea derivatei de ordin  $n$  prin apelarea repetată a derivatei de ordin 1 deja implementată;
- calcularea seriei Taylor într-un punct dat avem nevoie doar de derivata de ordin  $n$ ;
- calcularea rădăcinilor pentru polinoame de grad 2;
- calcularea rădăcinilor pentru polinoame de grad 3 folosind algoritmul lui Horner sau relațiile lui Viète;
- ridicarea la putere a unui polinom.

## 8. Bibliografie

- <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>
- <https://stackoverflow.com/>
- <http://www.mkong.com/tutorials/junit-tutorials/>
- <http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>
- <http://www.coned.utcluj.ro/~marcel99/PT/>
- [http://www.coned.utcluj.ro/~salomie/PT\\_Lic/](http://www.coned.utcluj.ro/~salomie/PT_Lic/)