

**DOCUMENTATIE TEMA 1: CALCULATOR DE  
POLINOAME**

**AGHENITEI BIANCA-IOANA**

CALCULATOARE ROMANA GRUPA 30225



**TECHNICAL UNIVERSITY**  
OF CLUJ-NAPOCA

## 1. Obiectivul temei

Se cere implementarea unei aplicatii Java cu interfata GUI care permite utilizatorului sa efectueze urmatoarele operatii asupra polinoamelor:

- adunarea a doua polinoame;
- scaderea a doua polinoame;
- inmultirea a doua polinoame;
- impartirea cu rest a doua polinoame;
- derivarea unui polinom;
- integrarea unui polinom;

Utilizatorul poate sa selecteze operatia dorita, sa introduca operanzii, iar interfata grafica va afisa rezultatul calculului.

### Obiective secundare:

1. Analiza problemei si dezvoltarea de scenarii de utilizare ()
2. Alegerea structurilor de date ()
3. Impartirea pe pachete si clase, alegerea structurii proiectului ()
4. Dezvoltarea algoritmilor de calcul ()
5. Implementarea interfetei grafice ()
6. Testare unitara ()

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

Utilizatorul va putea introduce datele de la tastatura in campuri rezervate. Se pun urmatoarele probleme:

1. interpretarea inputului de tip String (ex:  $5x^4+10x^5$ ) sub forma de date pentru a putea fi procesate si folosite in calcule: va fi folosit un Regex pentru separarea datelor de operatori;
2. programul poate implementa operatii atat cu un operand, cat si cu doi: ne dorim ca interfata grafica sa poata afisa fie un camp de introdus date, fie doua, in functie de operatia aleasa;

3.operatia de impartire returneaza doua outputuri: ne dorim ca interfata grafica sa afiseze fie doua campuri de iesire, fie unul, in functie de operatia aleasa;

Consideram urmatoarele cazuri de utilizare:

1.Utilizatorul doreste efectuarea unei operatii cu un operand sau doi. El alege operatia dorita slelectand butonul aferent;

2. Apar pe ecran numarul corespunzator de campuri de introdus date si de afisat date, cat si un buton specific operatiei, ce sugereaza efectuarea operatiei si afisarea rezultatului;

3.1 Utilizatorul introduce CORECT datele si apasa butonul, iar rezultatul este afisat in campul destinat;

3.2 Utilizatorul introduce INCORECT datele de intrare si apasa butonul, iar in campul de rezultat nu se va afisa nimic, el putand introduce in continuare date corecte;

4.1 Utilizatorul poate efectua aceeaasi operatie introducand date noi in campurile destinate. Rezultatul se va actualiza la apasarea butonului de calcul;

4.2 Utilizatorul poate selecta alta operatie, insa datele introduse deja in campuri vor ramane. Astfel el poate efectua operatii diferite cu aceiasi operanzi;

4.3 Utilizatorul paraseste operatia prin inchiderea ferestrei apasand X din coltul ei.

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

Operatiile pe polinoame presupun interactiunea cu monoamele individuale. Asadar, am ales ca structura de baza clasa Monom, care retine gradul unui monom si coeficientul lui. Utilizatorul poate introduce numai polinoame cu coeficienti intregi, insa operatiile de impartire si integrare presupun posibilitatea aparitiei coeficientilor reali. Asadar, pentru tipul coeficientului am ales float, rezultatul fiind afisat sub forma reala. Se utilizeaza principiu de incapsulare, astfel ca datele pot fi schimbate si accesate din exteriorul clasei numai prin metodele getter si setter.

Polinoamele introduse vor fi stocate intr-o structura de date de tip Polinom. Clasa polinom retine o lista (ArrayList) de obiecte de tip Monom. Am optat pentru folosirea unei colectii ArrayList pentru ca folosirea ei se preteaza operatiilor pe care le efectueam: pot fi retinute dubluri de date, iar parcurgerea ei se face foarte usor cu o instructiune for each.

Operatiile pe polinoame vor fi implementate in clasa Polinom, iar apelul lor se va face din exterior sub forma:

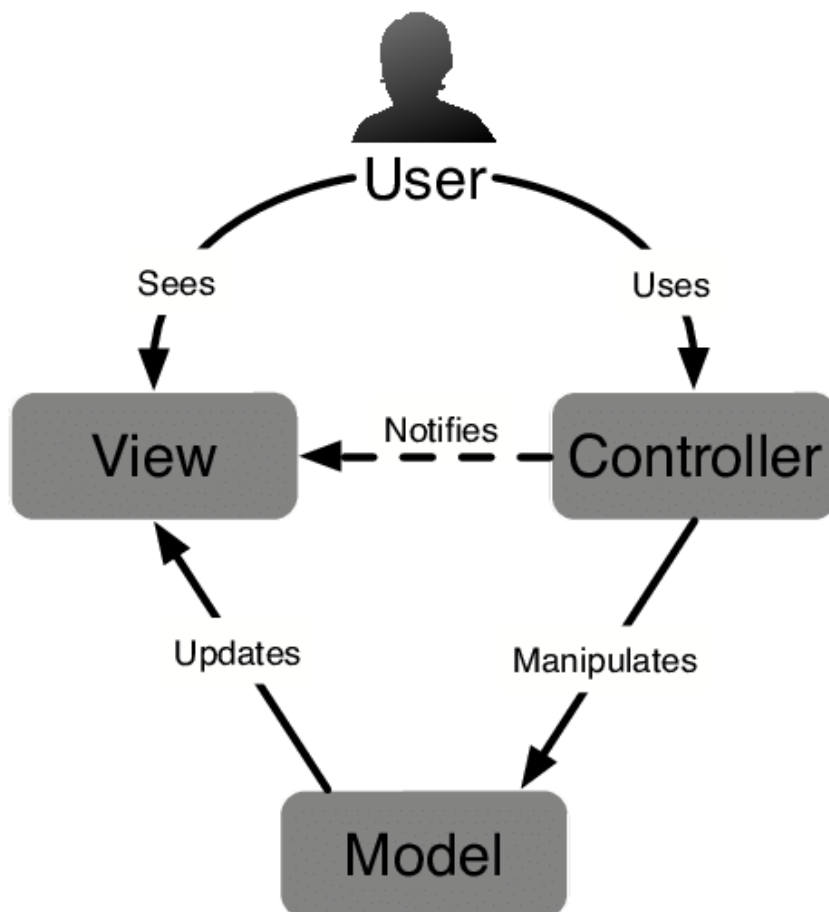
```
myPolinom.operatie1(); //pentru operatii cu un operand asupra  
lui myPolinom
```

```
myPolinom.operatie2(src) //pentru operatii cu doi operanzi intre  
myPolinom si src.
```

S-a incercat implementarea metodelor in maxim 30 de linii de cod.

Interfata grafica a fost realizata cu ajutorul Java Swing, care faciliteaza lucrul cu ferestre, butoane, campuri de introdus si afisat date, adica exact ce ne trebuie pentru acest proiect.

Pentru o buna organizare a codului am ales Model View Controller ca structura arhitecturala a proiectului. Acest lucru presupune impartirea claselor in 3 pachete separate cu nume sugestive: Model View si Controller, a caror clase interactioneaza intre ele pentru buna functionare a programului.



Pe scurt, View contine tot ce tine de interfata grafica. Utilizatorul trimite un request la Controller, in cazul nostru datele de procesat si operatia pe care vrea sa o efectueze. Controllerul directioneaza datele catre model si cere un rezultat. Astfel, Modelul este cel care se ocupa de procesarea informatiilor, iar clasele in care se efectueaza calculele

le-am inclus in pachetul Model. Controllerul primeste inapoi rezultatul si cere afisarea lui in View.

## 4. Implementare:

### 1. Clasa Monom

- retine coeficientul si gradul unui monom;
- implemeteaza metoda `Monom schimbaSemn()` , care schimba semnul coeficientului unui monom si retine rezultatul in this. Aceasta va fi utila in functia de scadere a polinoamelor.
- suprascrie clasa `toString` pentru o afisare frumoasa si inteligibila a rezultatului in interfata grafica.

### 2. Clasa Polinom

- retine un `ArrayList` de monoame (`listaMonoame`), cele din care este compus polinomul;
- implementeaza metoda `Polinom creeazaPolinom(String input)`, care primeste ca parametru un `String` si se foloseste de regex pentru a identifica structuri de tip `Monom`. Se creeaza un nou `Polinom`, iar monoamele gasite sunt adaugate la lista de monoame aferenta;
- implementeaza metoda `String adunaPolinom(Polinom src)`, care primeste ca parametru un `Polinom` si efectueaza adunarea acestura cu `Polinomul` retinut in this. Rezultatul este suprascris in this. Se parcurge lista de monoame din ambele polinoame si unde se gaseste coicienta de grad, coeficientii se aduna. Exista posibilitatea ca in polinomul parametru sa existe monoame de grade noi, care tebuie si ele adaugate la suma. Pentru a realiza asta, la final am parcurs intreg polinomul si in cazul in care am intalnit monoame de grad care nu mai exista in rezultat, le-am retinut intr-o lista pentru a le adauga la final.
- implementeaza medoda `String scadePolinom(Polinom src)`, analoaga cu cea pentru adunare, cu exceptia feptului ca monoamele din parametru care nu exista in rezultat la final vor trebui adaugare cu semn schimbat.

-implementeaza metoda `String deriveazaPolinom()`, care deriveaza polinomul stocat in this si il suprascrie cu rezultatul. Am tinut cont si de faptul ca o constanta derivata da 0;

- implementeaza metoda `String integreazaPolinom()`, care integreaza polinomul stocat in this si il suprascrie cu rezultatul.

-implementeaza metoda `String inmultestePolinom(Polinom src)` , care inmulteste polinomul primit a parametru cu cel din this si suprascrie rezultatul in this.

-implemeneaza metoda `Polinom impartePolinom(Polinom src)`, care imparte cu rest polinomul din this la polinomul trimis ca parametru. Restul este returnat de functie, iar catul suprascrie polinomul din this.

Algoritmul de impartire functioneaza in felul urmator:

**cat timp** gradul deimpartitului < gradul impartitorului

- adauga la cat monomul care inmultit cu cel mai mare monom din deimpartit da cel mai mare monom din impartitor

- inmulteste o copie a impartitorului cu noul monom adaugat la cat

- scade din deimpartit acest polinom obtinut la pasul anterior

**repeta**

La finalul executiei, in this va ramane restul, iar catul va fi returnat.

Intrucat valoarea impartitorului ar fi fost schimbata la fiecare iteratie, se impune folosirea unei copii a impartitorului. Am avut grija ca copia sa fie un nou obiect, care contine o noua lista de monoame, cu noi monoame, nu doar o referinta spre aceeasi lista de monoame. Pentru aceasta, am implementat un nou constructor in clasa Monom care

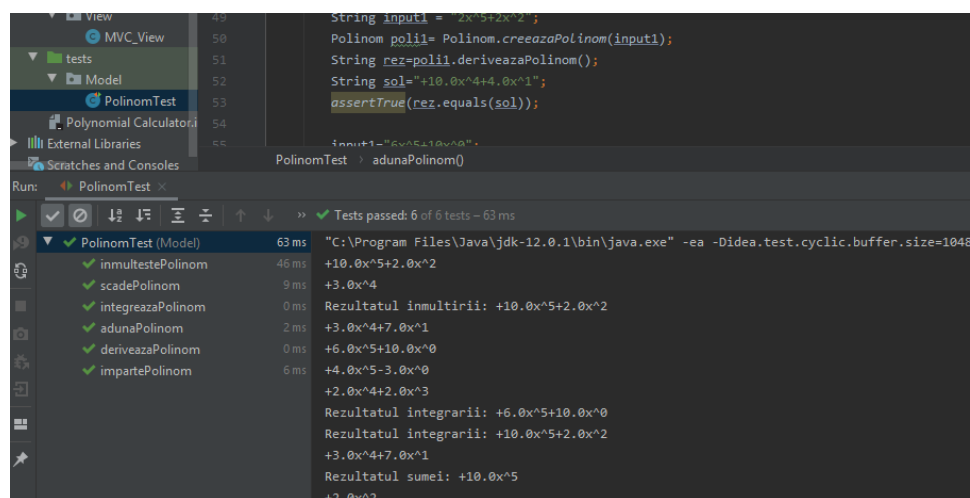
primește ca parametru un monom și creează un nou obiect cu aceleași valori ale variabilelor de instanță.

Pentru găsirea monomului de grad maxim dintr-un polinom am găsit soluția de a implementa metoda `Monom find_biggest()` care returnează monomul căutat itărind prin toată lista de monoame a polinomului din `this`.

## 5. Rezultate

Rezultatele operațiilor din clasa `Polinom` au fost testate folosind unitatea de testare `JUnit3`. Am creat teste pentru fiecare dintre funcțiile implementate, iar pentru fiecare operație am adăugat câte două teste, în care am încercat să includ și cazuri speciale.

Inițial, nu toate testele au trecut, dar cu această ocazie am găsit greseliile din algoritmi și le-am îndreptat, astfel că la final toate testele au trecut cu succes.



```
String input1 = "2x^5+2x^2";
Polinom poli1= Polinom.creeazaPolinom(input1);
String rez=poli1.deriveazaPolinom();
String sol="+10.0x^4+4.0x^1";
assertTrue(rez.equals(sol));

input1="6x^5+10x^0";
PolinomTest > adunaPolinom()
```

Run: PolinomTest x

Tests passed: 6 of 6 tests - 63 ms

Test Name	Duration
PolinomTest (Model)	63 ms
inmultestePolinom	46 ms
scadePolinom	9 ms
integreazaPolinom	0 ms
adunaPolinom	2 ms
deriveazaPolinom	0 ms
impartePolinom	6 ms

Console Output:

```
"C:\Program Files\Java\jdk-12.0.1\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048
+10.0x^5+2.0x^2
+3.0x^4
Rezultatul inmultirii: +10.0x^5+2.0x^2
+3.0x^4+7.0x^1
+6.0x^5+10.0x^0
+4.0x^5-3.0x^0
+2.0x^4+2.0x^3
Rezultatul integrarii: +6.0x^5+10.0x^0
Rezultatul integrarii: +10.0x^5+2.0x^2
+3.0x^4+7.0x^1
Rezultatul sumei: +10.0x^5
+2.0x^2
```

## 6. Concluzii

În urma finalizării acestui proiect consider că am fixat unele dintre noțiunile programării orientate pe obiect, m-am familiarizat cu crearea unei interfețe grafice cu `Jswing` folosind arhitectura `Model-View-Controller` și am fost introdusă unor noțiuni cu totul noi pentru



mine, precum folosirea unui Regex, a unei unitati de testare sau a platformei BitBucket.

Desi la crearea claselor si a metodelor nu am intalnit probleme, implementarea metodelor de calcul al polinoamelor mi-au dat de furca, intrucat lucrul cu colectii nu imi era familiar. M-am intampinat mai ales cu problema crearii unei clone a unui polinom: desi faceam un nou obiect de tip polinom, lista de monoame asociata pointa la aceeaasi locatie, iar schimbarea unei liste cauza schimbarea celilalte. Ma bucur ca am fixat aceasta notiune importanta.

De asemenea, aceasta a fost prima mea aplicatie dupa modelul MVC si desi la inceput mi se parea inutil sa existe atatea pachete si clase “in plus”, pana la finalizarea proiectului am inteles cat de avantajos este sa ai totul bine organizat, mai ales intr-un proiect amplu.

Scrierea “de mana” a interfetei grafice a fost un aspect nou pentru mine, fiind obisnuita cu un tool de tip “drag and drop”. Desi nu mi s-a parut la fel de usor, ma bucur am putut sa analizez si partea de cod din spate. M-am familiarizat cu layouturi si paneluri, aspecte despre care nu stiam mai nimic. La final ma pot lauda ca pot sa scriu o interfata grafica simpla, lucru care mi se pare un skill important pentru un programator in Java.

Unitatea de testare initial mi se parea foarte greu de realizat, dar in urma vizionarii mai multor clipuri despre asta, am ajuns la concluzia ca e cel mai usor aspect al temei si este chiar utila. Probabil in viitor voi apela singura la solutia aceasta pentru verificarea metodelor.

Din pacate timpul nu mi-a permis sa adaug mai mult proiectului, insa m-am gandit la mai multe posibilitati de imbunatatire ulterioara a lui:

- imbunatatirea regexului, pentru a permite introducerea mai “user-friendly” a datelor, de exemplu sa permita omiterea coeficientului 1;

-tratarea mai multor exceptii, atat la calcul, cat si la introducerea datelor;

-permiterea introducerii datelor cu coeficienti reali;

-imbunatatiri la interfata grafica: imagini, culori;

-adaugarea unei melodii pe fundal;

In concluzie, pot spune ca m-am distrat realizand acest proiect si sunt multumita cu rezultatul final. Sunt fericita ca am fixat niste notiuni importante de OOP, care ma vor ajuta la dezvoltarea in viitor a unor proiecte mai complexe.