

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
CATEDRA CALCULATOARE

DOCUMENTAȚIE

TEMA 1

CALCULATOR DE POLINOAME

CIOBAN FABIAN-REMUS
30223

CUPRINS

1.	Obiectivul temei.....	3
2.	Analiza problemei, modelare, scenarii, cazuri de utilizare	3
3.	Proiectare.....	4
4.	Implementare.....	5
5.	Rezultate.....	6
6.	Concluzii	6
7.	Bibliografie.....	6

1. Obiectivul temei

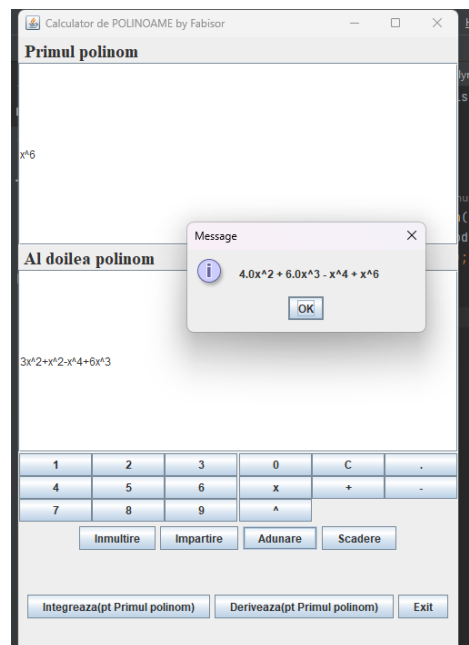
Obiectivul acestei teme este de a proiecta si de a implementa, un program care proceseaza operatiile cu polinoame. Polinoamele sunt de o singura variabila, noi fiind nevoiti efectiv sa implementam adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea polinoamelor. Altfel spus un calculator de polinoame cu o interfata grafica “User Friendly”, care sa poata sa fie utilizat cu usurinta de orice utilizator..

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

Un polinom reprezinta o expresie, formata din mai multe monoame, care, la randul lor sunt construite cu ajutorul unor coeficienti si exponenti. Exista o multime de metode prin care se poate realiza un calculator de polinoame. Eu, am ales sa folosesc Regex.

Modul prin care se introduc informatiile, modul in care sunt salvate in memorie polinoamele, modul in care sunt sortate in ordinea gradelor, aspectul interfetei grafice poate sa difere de la programator la programator, modul de implementare stand la latitudinea fiecaruia.

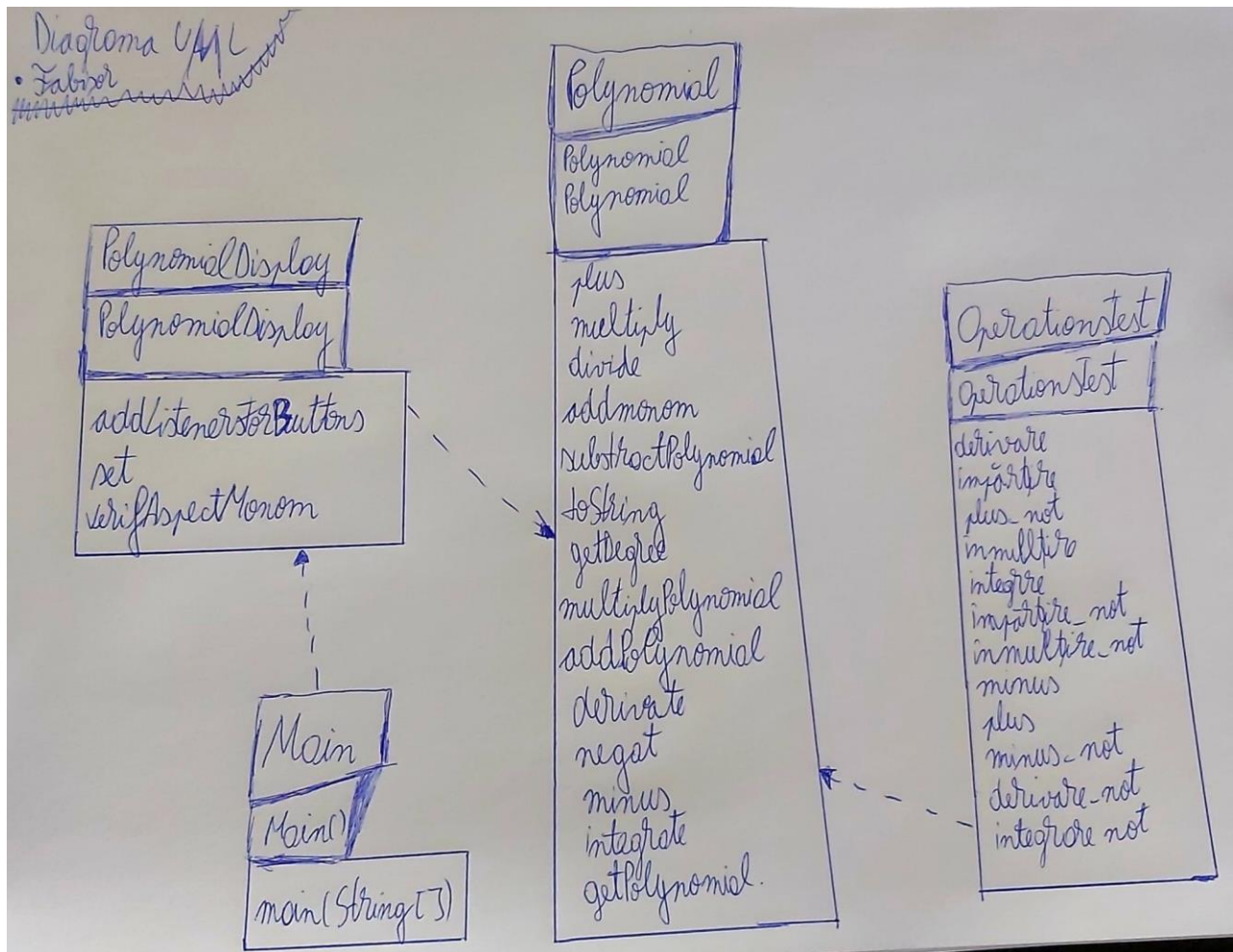
Pentru scenariile de utilizare, calculatorul functioneaza perfect, atata timp cat utilizatorul introduce corect polinomul/conditiile.



3. Proiectare

Diagrama UML

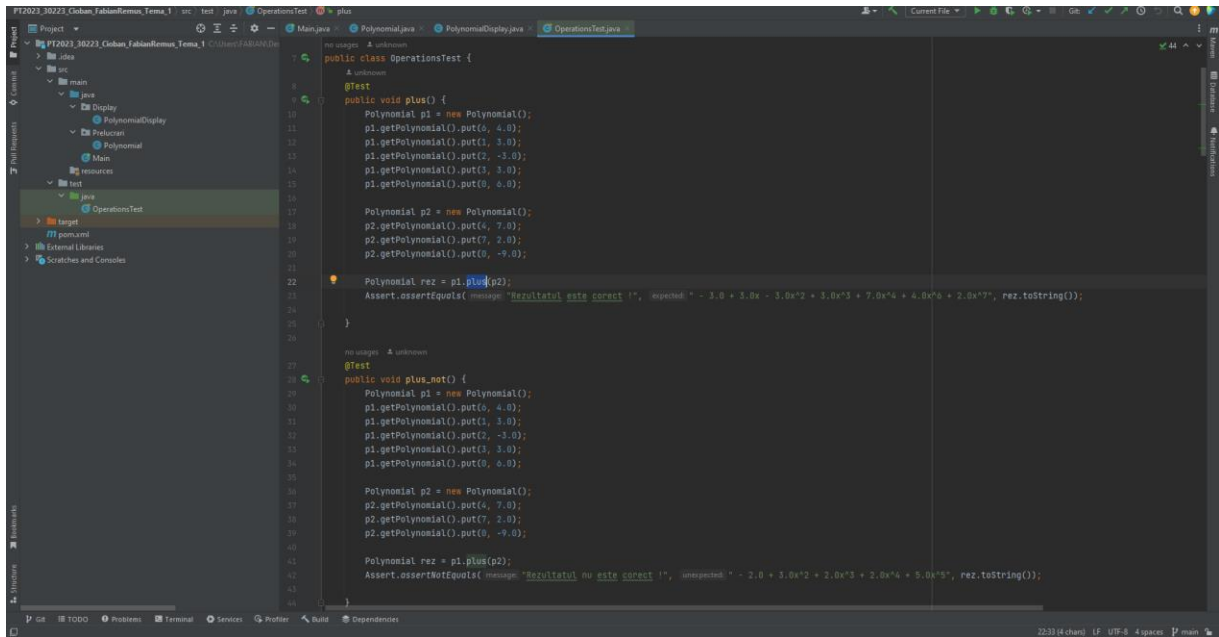
Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele și specificații pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al cărui fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.



La ce revine analiza problemei? Este foarte simplu, avem nevoie de un calculator rapid de polinoame deoarece operațiile cu polinoame dacă vei sta să le faci pe hartă îți vor lua foarte mult timp. Cum ar fi ca tu să fii nevoit doar să introduci polinoamele și apăsând operația să ți se furnizeze rezultatul? Ei bine acum este posibil.

5. Rezultate

Pentru rezultate, am implementat o clasa de test cu numele OperationsTest unde am facut teste pentru toate functionalitatile calculatorului cu ajutorul unei testari unitare implementate cu Junit, mai exact cu ajutorul metodei din Assertions, assertEquals, assertNotEquals unde am dat polinomul rezultat in urma operatiilor si rezultatul calculat de mine pe hartie.



```
public class OperationsTest {  
    @Test  
    public void plus() {  
        Polynomial p1 = new Polynomial();  
        p1.getPolynomial().put(4, 4.0);  
        p1.getPolynomial().put(1, 3.0);  
        p1.getPolynomial().put(2, -3.0);  
        p1.getPolynomial().put(3, 3.0);  
        p1.getPolynomial().put(0, 6.0);  
  
        Polynomial p2 = new Polynomial();  
        p2.getPolynomial().put(4, 7.0);  
        p2.getPolynomial().put(7, 2.0);  
        p2.getPolynomial().put(0, -9.0);  
  
        Polynomial rez = p1.plus(p2);  
        Assert.assertEquals("Rezultatul este corect !", expected: " - 3.0 + 3.0x - 3.0x^2 + 3.0x^3 + 7.0x^4 + 4.0x^6 + 2.0x^7", rez.toString());  
    }  
  
    @Test  
    public void plus_not() {  
        Polynomial p1 = new Polynomial();  
        p1.getPolynomial().put(4, 4.0);  
        p1.getPolynomial().put(1, 3.0);  
        p1.getPolynomial().put(2, -3.0);  
        p1.getPolynomial().put(3, 3.0);  
        p1.getPolynomial().put(0, 6.0);  
  
        Polynomial p2 = new Polynomial();  
        p2.getPolynomial().put(4, 7.0);  
        p2.getPolynomial().put(7, 2.0);  
        p2.getPolynomial().put(0, -9.0);  
  
        Polynomial rez = p1.plus(p2);  
        Assert.assertNotEquals("Rezultatul nu este corect !", unexpected: " - 2.0 + 3.0x^2 + 2.0x^3 + 2.0x^4 + 3.0x^5", rez.toString());  
    }  
}
```

6. Concluzii

In concluzie, acest proiect m-a invatat si m-a ajuta sa folosesc Regex, si cum sa leg clasele intre ele pentru o functionare cat mai corecta a modelului si pentru a putea implementa cat mai riguros controlul aplicatiei. . Viitori utilizatori ar putea fii elevii de clasa a 12-a deoarece acest calculator i-ar ajuta destul de mult.

7. Bibliografie

- 1.YouTube
- 2.Wikipedia
- 3.W3School
- 4.Oracle