

DOCUMENTATIE

TEMA 1

Nume student: Lucăcel Mălina

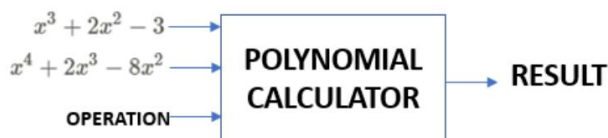
Grupa: 30221

CUPRINS

1. Obiectivul temei	3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare.....	3
3. Proiectare	4
4. Implementare	6
5. Rezultate	9
6. Concluzii.....	11
7. Bibliografie.....	11

1. Obiectivul temei

Obiectivul acestei teme a fost proiectarea si implementarea unui sistem de calcul, mai exact a unui calculator de polinoame implementat in limbajul de programare Java. Ca date de intrare avem 2 polinoame, iar ca date de iesire un singur polinom rezultat prin aplicarea operatiilor pe datele de intrare, in functie de caz.



Pe langa partea de implementare a operatiilor, proiectul beneficiaza si de o interfata “User Friendly”. Aceasta este intuitiva, fapt care o va face usor de folosit pentru orice utilizator, fie el specializat in programare sau nu.

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

Un polinom este vazut ca si o colectie de monoame. Un monom este un obiect care are un exponent(pozitiv) si un coeficient(pozitiv sau negativ). Acesta este de forma Ax^B , unde A este coeficientul, iar B este exponentul. Pentru termenul constant(termenul liber) se va seta exponentul ca fiind zero, iar pentru termenul de grad 1, vom seta exponentul ca fiind 1. Este important de precizat faptul ca, la fel ca si pentru exponenti, si coeficientii vor fi specificati(cazul in care coeficientul este +/- 1 se va specifica clar). Altfel, in aplicatie vor aparea erori.

In implementarea problemei am folosit o mapa de monoame, unde cheia va fi exponentul monomului, iar valoarea va fi monomul. Aceasta mapa a fost populata prin utilizarea unui regex (Regular Expression), mai exact `"([+-]?[0-9+)[a-z](\\^[0-9+))"`. Aceasta primeste un String(dat de utilizator in interfata grafica), il separa(coeficient si exponent), creeaza un Monom cu acestea, iar mai apoi adauga Monom-ul in mapa de polinoame.

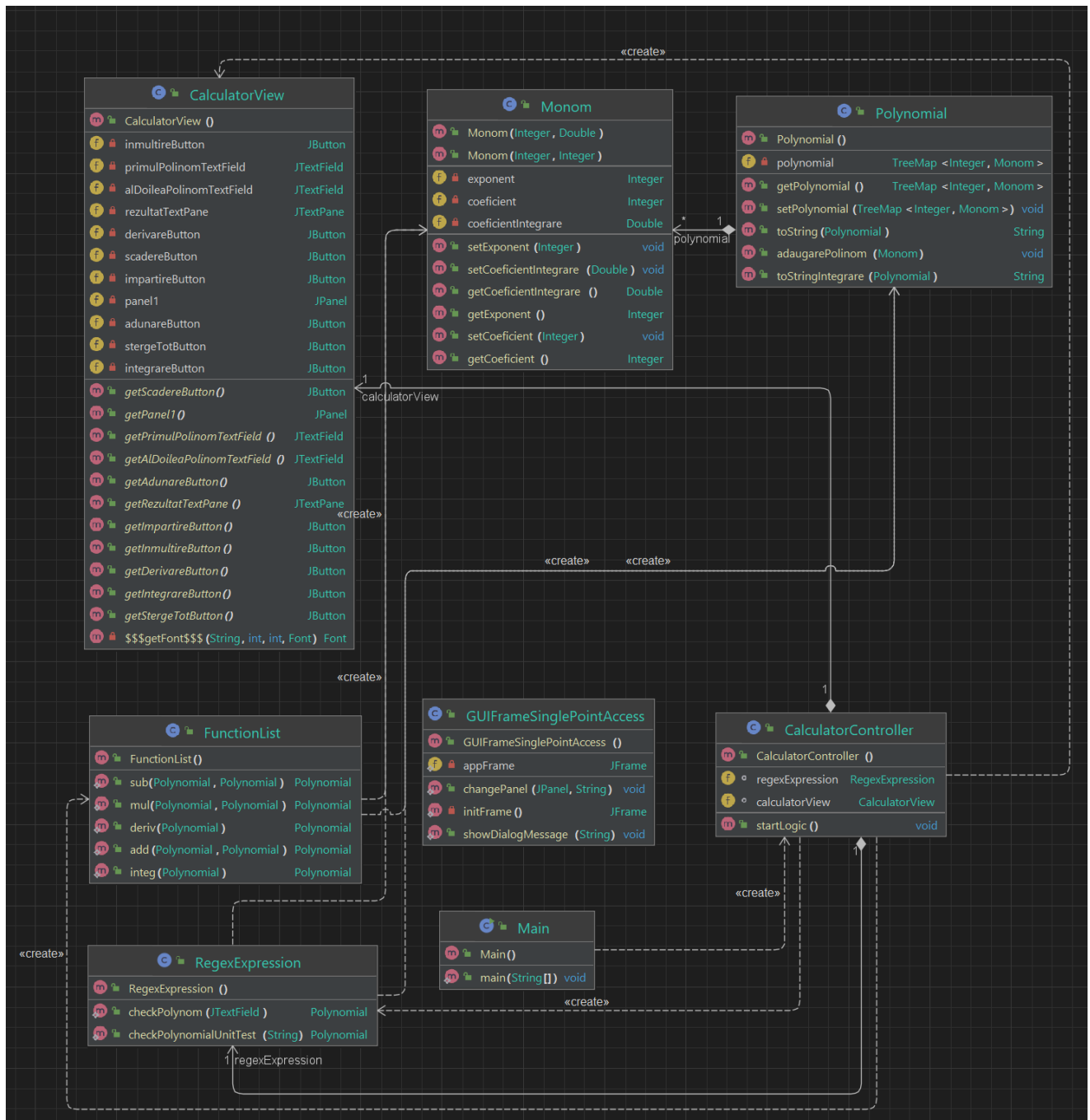
Pentru realizarea operatiilor am proiectat codul astfel:

- Suma a doua polinoame va returna un polinom
- Scaderea a doua polinoame va returna un polinom
- Inmultirea a doua polinoame va returna un polinom
- Impartirea a doua polinoame nu a fost implementata
- Integrarea unui polinom va returna un polinom
- Derivarea unui polinom va resturna un polinom

La nivelul interfetei grafice avem : 3 blocuri de text (Polinomul 1, Polinomul 2 si Rezultatul) si 7 butoane (Adunare, Scadere, Inmultire, Impartire(dezactivat), Integrare, Derivare si Sterge tot (sterge continutul tuturor blocurilor text)).

3. Proiectare

La partea de proiectare, aplicatia are urmatoarea structura:



Clasa Main este clasa principala din program. Aici apelam `calculatorController.startLogic()` . Se va porni atat interfata grafica, cat si tot ce tine de program deoarece in clasa `CalculatorController`, in metoda `startLogic()` se realizeaza interogarea pe fiecare buton in parte.

Pentru butoanele specifice operatiilor de adunare, scadere, respectiv inmultire se vor extrage din TextField-uri polinoamele pe care urmeaza sa se realizeze operatiile. Pentru butoanele specifice operatiilor de integrare si derivare se va extrage polinomul din primul TextField(Polinomul 1). Butonul caracteristic operatiei de impartire este dezactivat deoarece operatia nu a fost inca implementata.

Clasa CalculatorView implementeaza interfata grafica. Atributele clasei sunt toate componentele interfetei. "panel1" este panoul pe care se lucreaza, in care sunt afisate toate componentele pe care urmeaza sa le descriu. "primulPolinomTextField" este campul in care se introduce primul polinom, iar "alDoileaPolinomTextField" este campul in care se introduce al doilea polinom. La apasarea butoanelor "adunareButton", "scadereButton", "inmultireButton", "derivareButton", "integrareButton" se realizeaza operatiile specifice fiecaruia, rezultatul fiind afisat in campul text "rezultatTextPane". Butonul "impartireButton" este dezactivat, ca urmare nu va putea fi apasat. Butonul "stergeTotButton" sterge continutul tuturor campurilor text. In metoda CalculatorView() se regaseste tot codul pentru interfata(pozitionare, aliniament, culoare etc) .

Cu ajutorul setter-elor vom atribui o anumita valoare(data ca parametru) unui atribut corespunzator obiectului, iar cu ajutorul getter-elor vom accesa un anumit atribut corespunzatorobiect.

Clasa Polynomial primeste ca atribut "private TreeMap<Integer, Monom> polynomial = new TreeMap<>();", adica mapa in care va fi stocat polinomul. Stocarea polinomului se va face astfel: Integer – va reprezenta exponentul monomului, iar Monom va reprezenta efectiv monomul corespunzator exponentului. Este important sa folosim TreeMap deoarece aceasta este o lista ordonata care nu permite intrari duplicate. In implementarea aplicatiei am folosit constant metoda .descendingMap care ordoneaza lista in mod descrescator, facand lucrul cu polinoamele mult mai usor. Metoda toString(Polynomial p) transforma polinomul intr-un String pentru a putea fi afisat in campul text specific rezultatului. "StringBuilder" este o clasă predefinită în Java care oferă un mod eficient de a construi și manipula șiruri de caractere (Strings). Metoda toStringIntegrare(Polynomial p) va face exact acelasi lucru ca si metoda prezentata mai sus, atat ca, in acest caz avem "coeficient" de tip Double, pentru operatia de integrare.


Clasa Monom primeste ca attribute "exponent", "coeficient", respectiv "coeficientIntegrare". "exponent" va fi folosit pentru toate operatiile. "coeficient" va fi folosit pentru toate operatiile, exceptie facand operatia de integrare a unui polinom unde vom folosi "coeficientIntegrare".

Clasa RegexExpression contine metoda checkPolynom() care primeste ca parametru un camp text din care se va face citirea si se va "sparge" polinomul, dupa care se va crea cate un monom pentru fiecare termen al polinomului, pe baza caruia se va genera polinomul final de tip Polynomial. Pe langa metoda prezentata mai sus, clasa RegexExpression contine si metoda checkPolynomialUnitTest() care a fost implementata pentru teste. Spre deosebire de metoda precedenta, aceasta primeste ca parametru un String pe care va lucra.

Clasa FunctionList contine metodele add(), sub(), mul(), deriv() si integ(), metode care implementeaza operatiile de adunare, scadere, inmultire, derivare, respectiv integrare.

4.Implementare

In urmatoarele randuri voi prezenta modul in care aplicatia poate fi folosita de catre utilizator.



Se poate observa faptul ca interfata este “User Friendly”, fiind foarte intuitiva pentru utilizator.

In primele doua campuri text se vor introduce polinoamele. Acestea trebuie sa fie de forma :

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

Este important de mentionat faptul ca, in cazul in care avem coeficientul sau exponentul 0 sau 1, acesta trebuie mentionat ca fiind 0 sau 1. Altfel, vor aparea erori.

Se va apasa butonul specific operatiei pe care dorim sa o realizam, iar rezultatul va aparea in ultimul camp text.

Exemple operatii :

Polinoamele pe care se vor realiza operatiile sunt :

Polinom 1 : $1x^3 + 4x^2 + 5x^0$


Polinom 2 : $3x^2 - 2x^1 + 1x^0$

4.1 Adunare

CALCULATOR DE POLINOAME

Primul polinom:

Al doilea polinom:



Adunare	Scadere
Inmultire	Impartire
Integrare	Derivare

Rezultat:


Sterge tot

4.2. Scadere

CALCULATOR DE POLINOAME

Primul polinom:

Al doilea polinom:



Adunare	Scadere
Inmultire	Impartire
Integrare	Derivare

Rezultat:


Sterge tot

4.3. Inmultire

CALCULATOR DE POLINOAME

Primul polinom:

Al doilea polinom:



Adunare	Scadere
Inmultire	Impartire
Integrare	Derivare

Rezultat:

Sterge tot


4.4. Integrare

Integrare se face pe Polinomul 1

CALCULATOR DE POLINOAME

Primul polinom:

Al doilea polinom:



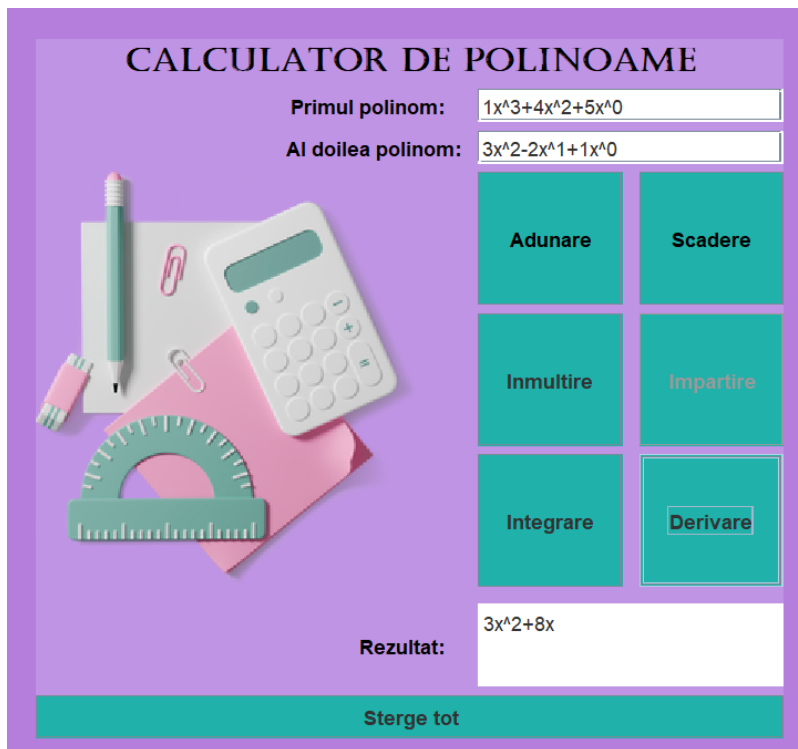
Adunare	Scadere
Inmultire	Impartire
Integrare	Derivare

Rezultat:

Sterge tot

4.5. Derivare

Derivarea se face pe Polinomul 1



The image shows a web-based application titled "CALCULATOR DE POLINOAME". It features a purple background with a central illustration of school supplies: a green pencil, a pink paperclip, a white calculator, a pink ruler, and a green protractor. The interface includes two input fields for polynomials: "Primul polinom:" with the value $1x^3+4x^2+5x^0$ and "Al doilea polinom:" with the value $3x^2-2x^1+1x^0$. To the right of these fields is a grid of six teal buttons: "Adunare", "Scadere", "Inmultire", "Impartire", "Integrare", and "Derivare". The "Derivare" button is highlighted with a white border. Below the buttons is a "Rezultat:" field displaying $3x^2+8x$. At the bottom is a teal button labeled "Sterge tot".

5. Rezultate

Testarea unitara a fost implementata cu JunitTest. Am realizat testele dupa cum urmeaza:

- Pentru testele la care asteptam un rezultat corect am folosit metoda assertEquals()
- Pentru testele la care asteptam un rezultat incorect am folosit metoda assertEquals()

Cele doua polinoame au fost initializate inaintea fiecarui test folosind adnotarea "@Before".

Metodele de test au fost marcate folosind adnotarea "@Test".

```

1 import org.example.Functions.FunctionList;
2 import org.example.Objects.Polynomial;
3 import org.example.RegExpExpression;
4 import org.junit.Before;
5 import org.junit.Test;
6 import static org.junit.jupiter.api.Assertions.assertEquals;
7 import static org.junit.jupiter.api.Assertions.assertNotEquals;
8
9 public class OperationsTest {
10     Polynomial p1 = new Polynomial();
11     Polynomial p2 = new Polynomial();
12     Polynomial p1Afisare = new Polynomial();
13     Polynomial p2Afisare = new Polynomial();
14     @Before
15     public void init()
16     {
17         p1 = RegExpExpression.checkPolynomialUnitTest( p: "1x^3+4x^2+5x^0");
18         p2 = RegExpExpression.checkPolynomialUnitTest( p: "3x^2-2x^1+1x^0");
19         p1Afisare = RegExpExpression.checkPolynomialUnitTest( p: "1x^3+4x^2+5x^0");
20         p2Afisare = RegExpExpression.checkPolynomialUnitTest( p: "3x^2-2x^1+1x^0");
21     }
22     @Test
23     public void addTestRight() {assertEquals( expected: "x^3+7x^2-2x+6", FunctionList.add(p1,p2).toString(FunctionList.add(p1Afisare,p2Afisare)));}
24
25     @Test
26     public void addTestWrong() {assertNotEquals( unexpected: "x^3-7x^2-2x+6", FunctionList.add(p1,p2).toString(FunctionList.add(p1Afisare,p2Afisare)));}
27
28     @Test
29     public void subTestRight() {assertEquals( expected: "x^3+x^2+2x+4", FunctionList.sub(p1,p2).toString(FunctionList.sub(p1Afisare,p2Afisare)));}
30
31     @Test
32     public void subTestWrong() {assertNotEquals( unexpected: "x^3-x^2+2x+4", FunctionList.sub(p1,p2).toString(FunctionList.sub(p1Afisare,p2Afisare)));}
33
34     @Test
35     public void mulTestRight() {assertEquals( expected: "3x^5+10x^4-7x^3+19x^2-10x+5", FunctionList.mul(p1,p2).toString(FunctionList.mul(p1Afisare,p2Afisare)));}
36
37     @Test
38     public void mulTestWrong() {assertNotEquals( unexpected: "3x^5-10x^4-7x^3+19x^2-10x+5", FunctionList.mul(p1,p2).toString(FunctionList.mul(p1Afisare,p2Afisare)));}
39
40     @Test
41     public void derivTestRight() { assertEquals( expected: "3x^2+8x", FunctionList.deriv(p1).toString(FunctionList.deriv(p1))); }
42
43     @Test
44     public void derivTestWrong() {assertNotEquals( unexpected: "3x^2-8x", FunctionList.deriv(p1).toString(FunctionList.deriv(p1)));}
45
46     @Test
47     public void integTestRight() {assertEquals( expected: "0.25x^4+1.33x^3+5.00x+C", actual: FunctionList.integ(p1).toStringIntegrare(FunctionList.integ(p1))+"+C");}
48
49     @Test
50     public void integTestWrong() {assertNotEquals( unexpected: "0.25x^4-1.33x^3+5.00x+C", actual: FunctionList.integ(p1).toStringIntegrare(FunctionList.integ(p1))+"+C");}
51
52     }
53 }

```

Rezultatele testelor :

Run: OperationsTest ×

Tests passed: 10 of 10 tests – 50 ms

Test Name	Duration	Status
OperationsTest	50 ms	Passed
mulTestRight	18 ms	Passed
mulTestWrong	2 ms	Passed
derivTestRight	1 ms	Passed
derivTestWrong	1 ms	Passed
addTestRight	2 ms	Passed
addTestWrong	1 ms	Passed
integTestRight	22 ms	Passed
integTestWrong	0 ms	Passed
subTestRight	2 ms	Passed
subTestWrong	1 ms	Passed

Process finished with exit code 0

Tests passed: 10 (moments ago)

6. Concluzii

Concluzionand, acest proiect m-a ajutat sa-mi aprofundez cunostiintele in limbajul de programare Java(utilizarea corecta a obiectelor, metodelor, TreeMap-urilor etc), precum si in domeniul GUI.

La capitolul dezvoltari ulterioare pot propune urmatoarele : forma algebrica a unui polinom, inmultirea unui polinom cu un scalar, impartirea cu rest.

7. Bibliografie

<https://www.javatpoint.com/java-regex>

<https://compiler.javatpoint.com/opr/test.jsp?filename=RegexExample1>

<https://www.geeksforgeeks.org/treemap-in-java/>

<https://mkyong.com/java/java-display-double-in-2-decimal-points/>