Universitatea Tehnică din Cluj-Napoca
Facultatea de Automatică și Calculatoare

DOCUMENTAȚIE

TEMA NUMĂRUL 1

Nume student: Maria-Magdalena Creț

Grupa 30223

Cuprins

1 Obiectivul temei **3**
1.1 Obiectivele secundare ale acestei teme sunt: 3

2 Analiza problemei, modelare, scenarii, cazuri de utilizare **4**
2.1 Analiza problemei și cum se realizează operațiile matematice pe polinoame . . 4
2.2 Modelarea problemei pentru scrierea codului Java 6

3 Proiectare **8**

4 Implementare **9**

5 Rezultate **16**

6 Concluzii **17**

7 Bibliografie (Webografie) **19**

1 Obiectivul temei

Obiectivul principal a acestei teme este acela de a se proiecta și implementa un calculator care rezolvă operațiile de polinoame cu o interfata grafică realizată cu GUI care permite utilizatorului să introducă polinoame, respectiv să se selecteze operația matematica dorită pentru acestea. Operațiile sunt: adunare, scădere, inmulțire și impărțire, iar pentru un singur polinom se găsesc și operațiile de derivare și integrare. Acestea trebuie efectuate pe baza codului implementat în Java, iar rezultatul pentru fiecare dintre operațiile menționate mai sus trebuie să apară pe ecran, în interfața grafică realizată. Pentru implementarea interfeței grafice există posibilitatea alegerii între Java Swing sau JavaFX.

Cerința acestei teme este următoarea:

• Utilizați un design de programare orientat pe obiecte (utilizați încapsularea, definiți clase adecvate: de exemplu, Polinom și, opțional, Monomial).

• Utilizați Map pentru modelarea polinomului.

• Folosiți foreach în loc de for(int i=0...).

• Implementați o interfață grafică cu utilizatorul, utilizând Java Swing sau JavaFX.

• Implementați operațiile pentru polinoame (operații menționate mai sus)

• Implementați clase cu maximum 300 de linii (cu excepția claselor UI) și metode cu maxim 30 de linii.

• Utilizați convențiile de denumire Java.

• Documentație de bună calitate care se adresează tuturor secțiunilor din documentație șablon.

• Bună organizare a codului sursă.

• Folosiți expresii regulate și potrivirea modelului pentru extragerea coeficienților polinomului

• Utilizați Junit pentru testare

1.1 Obiectivele secundare ale acestei teme sunt:

1. Asigurarea unei interfețe grafice prietenoase și ușor de folosit pentru utilizator, care să permită introducerea și vizualizarea polinoamelor într-un mod ușor și intuitiv. Acest lucru realizându-se după modelul MVC (Model View Controller).
2. Implementarea operațiilor matematice pentru polinoame într-un mod eficient astfel încât să se ofere o performanță bună a aplicației, indiferent de numărul de monoame a polinoamelor introduse sau de gradul acestora.
3. Realizarea claselor necesare funcționării calculatorului de polinoame.
4. Realizarea unei bune organizări a codului, a claselor, pentru a putea face ca acest cod sa fie cât mai ușor și accesibil de reutilizat sau modificat în caz de nevoie
5. Documentarea temei, redarea informațiilor corespunzătoarea realizării acesteia, pentru ca aplicație sa fie cât mai ușor de înteles și utilizat de orice persoană.

6. Folosirea expresiilor regulate (Regular Expression) și a potrivirii modelului (Pattern matching) pentru extragerea coeficienților și a puterii fiecărui monom din polinoame. Înțeledera acestor Regular Expression și învățarea utilizării lor.

2 Analiza problemei, modelare, scenarii, cazuri de utilizare

2.1 Analiza problemei și cum se realizează operațiile matematice pe polinoame

Se vor reprezenta mai jos operațiile cerute pe polinoame din punct de vedere matematic pentru a putea înțelege mai ușor cum se realizează acestea ulterior în cod.

1. Adunarea a doua polinoame:

Pentru două polinoame de forma:

$P1(x) = 3x^3 + 2x^2 + 5x - 1$
$P2(x) = 6x^2 + 2x + 3$

Operația de adunare se realizează în următorul mod:
Idee: Termenii de același grad se adună, urmând a se aduna apoi la polinomul rezultat, termenii lăsați liberi.

Se va obține pentru cele două polinoame de mai sus:
$P(x) = (3x^3 + 2x^2 + 5x - 1) + (6x^2 + 2x + 3)$
$P(x) = 3x^3 + (2x^2 + 6x^2) + (5x + 2x) - 1 + 3$
$P(x) = 3x^3 + (2x^2 + 6x^2) + (5x + 2x) - 1 + 3$
$\Rightarrow \mathrm{P(x)} = 3\mathrm{x}^3 + 8x^2 + 7x + 2$

2. Scăderea a doua polinoame:

Pentru două polinoame de forma:

$P1(x) = 3x^3 + 2x^2 + 5x - 1$
$P2(x) = 6x^2 + 2x + 3$

Operația de scădere se realizează în următorul mod:
Termenii de același grad se scad, urmând a se scadea apoi la polinomul rezultat, termenii lăsați liberi. Este adunarea primului polinom cu cel de-al doilea cu semn schimbat. (Așa cum se va observa implementarea în cod pentru a refolosi codul de la metoda de adunarea a polinoamelor).

Se va obține pentru cele două polinoame de mai sus:

$P(x) = (3x^3 + 2x^2 + 5x - 1) - (6x^2 + 2x + 3)$
$\Rightarrow \mathrm{P(x)} = 3\mathrm{x}^3 - 4x^2 + 3x - 4$

3. Înmulțirea a doua polinoame:

Pentru două polinoame de forma:

$P1(x) = 2x^2 + 3$
$P2(x) = 3x - 1$

Operația de înmulțire se realizează în următorul mod:
Idee: Se înmulțește termen cu termen.

Se va obține pentru cele două polinoame de mai sus:

$P(x) = (2x^2 + 3) * (3x - 1)$
$\Rightarrow \mathrm{P(x)} = 6\mathrm{x}^3 - 2x^2 + 9x - 3$

4. Împărțirea a doua polinoame:

Pentru două polinoame de forma:

$P1(x) = 2x^2 + x + 2$
$P2(x) = 3x - 1$

Operația de împărțire se realizează în următorul mod:
Observație: După schema lui Horner.

Se va obține pentru cele două polinoame de mai sus:

$$\Rightarrow \begin{array}{r|l} & \frac{2}{3}x + \frac{5}{9} \\ \cline{2-2} 3x - 1 & 2x^2 + x + 2 \\ \cline{1-1} -2x^2 + \frac{2}{3}x & \\ \cline{1-1} \frac{5}{3}x + 2 & \\ \cline{1-1} -\frac{5}{3}x + \frac{5}{9} & \\ \cline{1-1} \frac{23}{9} & \end{array}$$

5. Derivarea unui polinom:

Pentru un polinom de forma:

$P(x) = 4x^4 + 2x^3 + 2x - 5$

Idee: Coeficientul fiecarui monom se va înmulți cu gradul monomul, scăzându-se ulterior gradul fiecarui monom din dericare, cu 1. Termenii considerati constante(monom de grad 0) vor fi 0, adică se elimină din polinomul rezultat.

$$P'(x) = \frac{\mathrm{d}P(x)}{\mathrm{d}x}$$

$\Rightarrow$ P'(x) = $16x^3 + 6x^2 + 2$

5. Integrarea unui polinom:

Pentru un polinom de forma:

$P(x) = 2x^3 + 3x^2 + 2x - 5$

Idee: Coeficientul fiecarui monom se va împărți cu gradul monomul la care se adauga 1. Se actualizeaza apoi gradul monomului cu +1. Termenii considerati constante(monom de grad 0) vor avea la integrare gradul egal cu 1.

$\Rightarrow \int (2x^3 + 3x^2 + 2x - 5)\, dx = \frac{2}{4}x^4 + \frac{3}{3}x^3 - 5x + C = \frac{1}{2}x^4 + x^3 - 5x + C$

2.2 Modelarea problemei pentru scrierea codului Java

Din definiția matematică înțelegem că un polinom constituie o expresie construită dintr-una sau mai multe variabile și constante.Fiecare polinom este realizat dintr-unul sau mai multe monoame (putem considera astfel că am împărțit acest polinom în blocuri componente pentru a ușura realizarea operațiilor) Calculatorul va functiona cu polinoame într-o singură variabilă (o considerăm variabila x), ca de exemplu : x^2+5x-7. Nu se vor realiza operații pe polinoame cu mai multe variabile, ca de exemplu : $x^4+5xyz+2z-1$. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile in acel monom.

Așadar pentru a simplifica rezolvarea problemei și pentru a crea o structură cât mai ușor de analizat, unde se pot realiza într-un mod accesibil și rapid modificări, s-a ales organizarea codului Java în pachete(packages):

- polynomial calculator package: se integreaza clasele care conțin implementarea codului. Clasele sunt:
  - Polynomial , care conține constructori pentru un obiect de tipul Polynomial și metoda toString() pentru afișarea unui obiect de tipul Polynomial sub formă de string;
  - Monom, o clasă în care s-a realizat metoda toString() pentru fiecare dintre monoamele pe care le conține un polinom, pentru a ușura realizarea codului;
  - PolynomialOperation, clasa în care s-au realizat operațiile matematice cerute pentru polinoame;

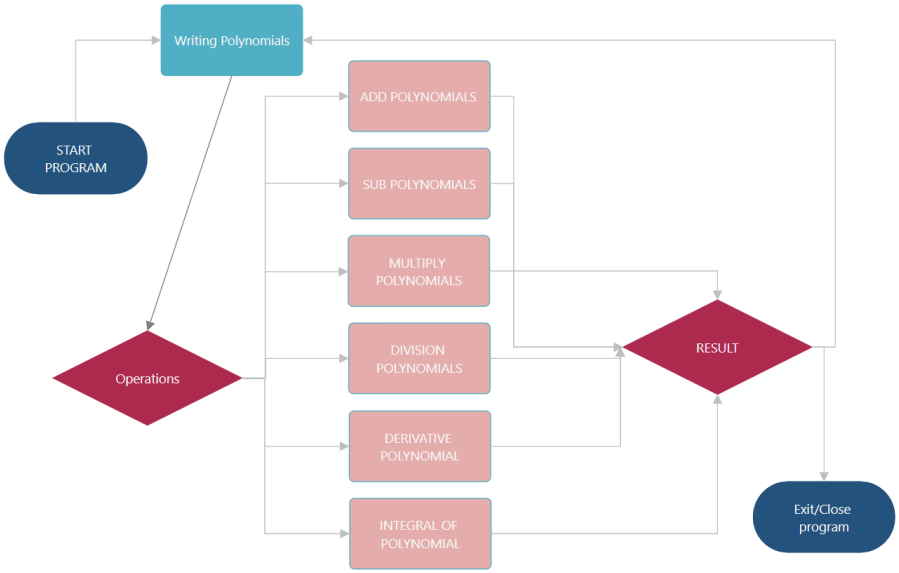

Figura 1: Diagramă pentru crearea unui model(plan), pentru realizarea interfeței grafice corespunzătoare temei curente

  - PolynomialRead, clasa în care s-a implementat metoda readPolynomial, care primește un polinom introdus de la tastatura de tipul String și din care, cu ajutorul utilizării unor Regular Expression (regex) se extrage coeficientul și gradul unui monom, se seteaza apoi monomul respectiv și se introduce într-un polinom pentru a se putea realiza operațiile pe acesta;

- PolynomialCalculatorGUI: se poate considera ca fiind un pachet pentru că acesta conține clasa PolynomialCalculatorGUI, unde s-a implementat codul pentru design-ul aplicației, dar este de fapt, mai mult de atât, un Swing UI Design. Pentru interfața grafică s-a ales implementarea cu Java Swing.

Observații suplimentare:

În scrierea monoamelor nu s-a folosit simbolul "*" care ar fi putut înseamna înmultire, ci coeficientul o să fie lipit de x, iar puterea o să fie reprezentată prin simbolul "^",având următoarea structură: (de exemplu) 2x^3+x-3 La rularea programul se va deschide o interfață grafică în care se vor putea realiza următorii pași:

1. Se va tasta primul polinom în field-ul "Polynomial1" (de la tastatură);
2. Se va tasta al doilea polinom în field-ul "Polynomial2" (de la tastatură);
3. Pentru a efectua operația dorită, se accesează unul dintre butoanele care activează operația respectivă, iar în field-ul "Result" o sa obtinem rezultatul efectiv. În field-ul acesta nu se poate tasta nimic. S-au realizat constrângeri pentru introducerea datelor(de exemplu nu se pot introduce în field-uri altfel de date, în afară de "." sau numere de la 0 la 9, variabila "x" și simbolul "^". Acest lucra realizându-se pe baza unui regex utilizat în metoda verifyInput().

4. Se poate repeta acest proces ori de câte ori se dorește, iar la finalizarea procesului se poate inchidem interfața grafică prin apasarea tastei X din dreptul ferestrei sus. Aceasta fereastră se poate mări sau micșora după preferințe. Conține un titlu și un aspect prietenos utilizatorului.

3 Proiectare

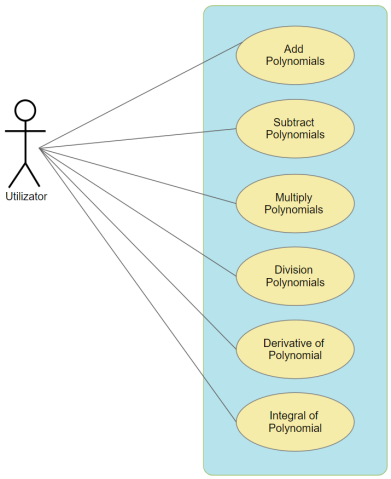

Figura 2: Diagrama Use-Case

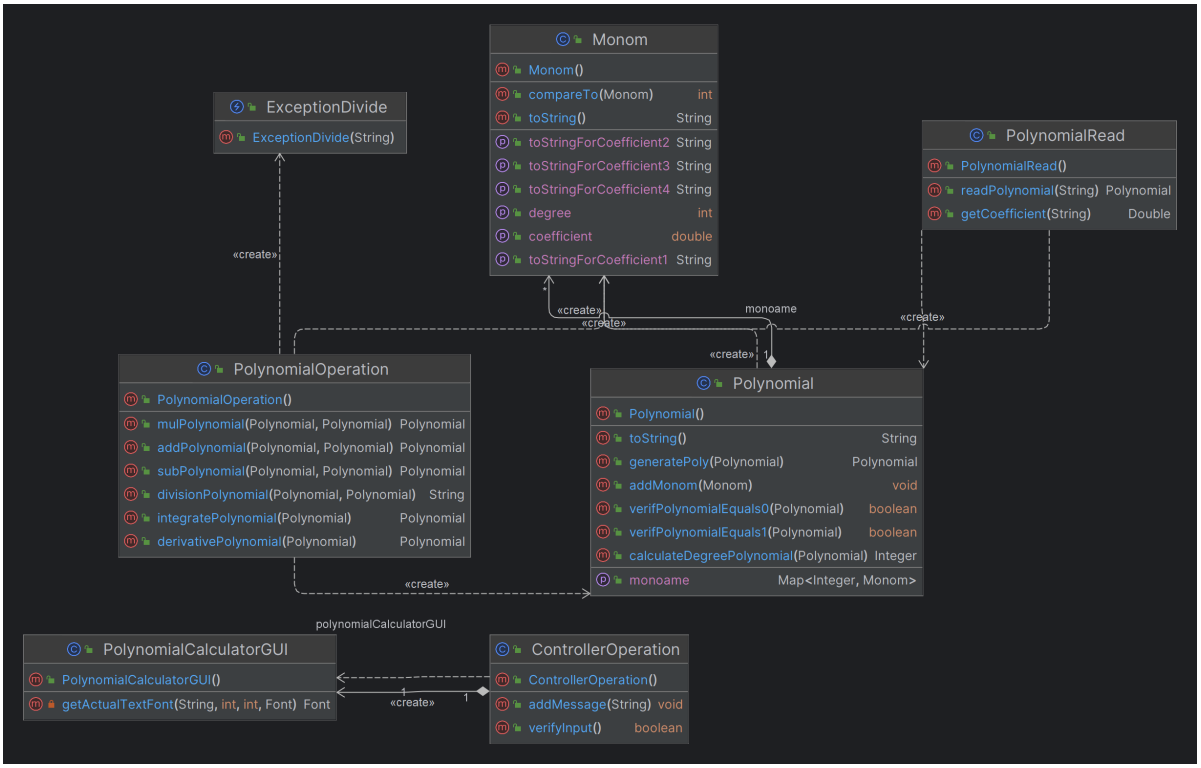

Figura 3: Diagrama UML pentru clase

4 Implementare

Implementarea codului s-a realizat structural, pe baza a 4 pachete: exception_package, polynomial_calculator, test_junit_polynomial și pachetul de Swing UI Designer: PolynomialCalculatorGUI, este singurul la care nu se respectă convenția de nume pentru pachetele Java(litere mici și spații pentru despărțirea între cuvinte), deoarece acesta conține o clasa în care se realizeaza partea de UI Design, cu același nume.
Pachetele sunt structurate altfel:

(a) Pachetul exception_package conține o clasa ExceptionDivide, care extinde superclasa Throwable, pentru toate erorile și excepțiile limbajului Java. Această clasă conține un constructor super(), pentru a putea transmite un mesaj atunci când are loc excepția.

(b) Pachetul polynomial_calculator, poate fi considerat și cel mai important sau pachetul principal al proiectului, deoarece el conține toate clasele care realizeaza operațiile pe polinoame, cât și celelalte metode importante, ajutătoare. Clasele din acest pachet sunt: *Monom*, *Polynomial*, *PolynomialOperation* și clasa *PolynomialRead*. Mai jos se vor găsi explicații pentru fiecare dintre clasele menționate anterior.

(c) Pachetul test_junit_polynomial, acesta conține clasa de teste unitare, *TestJunitP*, unde se realizează testele unitare pentru majoritatea operațiilor realizate pe polinoame(in special, pentru operațiile matematice cerute)

(d) *PolynomialCalculatorGUI* este pachetul care conține o clasă cu același nume, unde se implementează codul din limbajul Java cu Swing pentru crearea interfeței grafice a aplicației.

Clasele sunt cele numite mai sus, iar informațiile importante despre acestea sunt următoarele:

(a) Clasa *Monom*: conține argumentele necesare reprezentării unui monom *grad* și *coeficient*, un constructor fără argumente utilizat în clasa Polynomial spre exemplu, gettere(returnează argumentele) și setere(seteaza valoarea argumentelor), acestea ne permit să preluam dintr-un monom al unui polinom gradul și coeficientul și, respectiv, să le modificăm în funcție de cerințele proiectului. În această clasă, o metodă foarte importantă este toString(), care permite scrierea pe ecran a monomului sub forma unui String(sau șir de caractere), pentru a fi ușor de vizualizat de utilizator. S-au introdus în acestă metodă simboluri precum *, care reprezintă înmulțirea.

```
1 package polynomial_calculator_package;
2
3 public class Monom implements Comparable<Monom>{
4     private double coefficient;
5     private int degree;
6
7     public Monom() {
8         this.coefficient = 0.0;
9         this.degree = 0;
```

```
10      }
11      public double getCoefficient() {
12          return coefficient;
13      }
14      public void setCoefficient(double coefficient) {
15          this.coefficient = coefficient;
16      }
17      ..........................................................

18
19          public String toString() {
20          if(this.coefficient == 1)
21          {
22              return getToStringForCoefficient1();
23          }
24          else if(this.coefficient > 0) {
25              return getToStringForCoefficient2();
26          }
27          else if(this.coefficient == -1){
28              return getToStringForCoefficient3();
29          }
30          else if(this.coefficient < 0){
31              return getToStringForCoefficient4();
32          }
33          else {
34              return 0 + "";
35          }
36      }
37  }
```

(b) Clasa *Polynomial*: conține o mapă de monoame *private Map¡Integer, Monom¿ monoame = new TreeMap¡Integer, Monom¿(Collections.reverseOrder());*. S-a ales această variantă de mapă pentru ca stocarea monoamelor se se realizez în ordine crescătoare(sortate) după grad. De asemenea această clasă, conține metoda toString(), suprascrisă (@Override), pentru a putea afișa pe ecran(în fereastra aplicației), un polinom sub forma de String. Se mai găsește si metoda addMonom, care ne permite să inserăm într-un polinom nou creat, instanțiat de clasa POlynomial, un monom, cu gradul și coeficientul său. Acest monom se adaugă în mapa de monoame a unui polinom, folosind gradului monomul și cheie. Valorile monomului pot fi considerați coeficienții.

```
1   public void addMonom(Monom monom) {
2           monoame.put(monom.getDegree(), monom);
3       }
```

(c) Clasa *PolynomialOperation*: Aici se realizează operațiile pe polinoame, operațiile care sunt cele de adunare, scădere, înmulțire și împărțire. Pentru un singur polinom, ales pentru interfața grafică ca fiind primul (Polynomial1), se realizează și

operațiile de derivare și integrare.
Exemplu de operații:
Operația de Adunare a doua polinoame:
Observație! Scăderea s-a realizat apelând operația de adunare, dar cel de-al doilea polinom s-a înmulțit cu -1, astfel că s-a reutilizat codul, scăpând de doua metode redundante.

```
1  public class PolynomialOperation {
2      public static Polynomial addPolynomial(Polynomial
           polynomial1, Polynomial polynomial2) {
3          Polynomial polynomial3 = new Polynomial();
4          for (int degree : polynomial1.getMonoame().keySet())
               {
5              Monom monom = new Monom();
6              Double coefficientPoly1 = polynomial1.getMonoame
                   ().get(degree).getCoefficient();
7              Double coefficientPoly3 = 0.0;
8              if (polynomial2.getMonoame().containsKey(degree))
                    {
9                  coefficientPoly3 = coefficientPoly1 +
                       polynomial2.getMonoame().get(degree).
                       getCoefficient();
10             } else {
11                 coefficientPoly3 = coefficientPoly1;
12             }
13             if (coefficientPoly3 != 0.0) {
14                 monom.setCoefficient(coefficientPoly3);
15                 monom.setDegree(degree);
16                 polynomial3.addMonom(monom);
17             }
18         }
19         for (int degree : polynomial2.getMonoame().keySet())
               {
20             Monom monom = new Monom();
21             if (polynomial1.getMonoame().containsKey(degree)
                   == false) {
22                 monom.setCoefficient(polynomial2.getMonoame()
                       .get(degree).getCoefficient());
23                 monom.setDegree(polynomial2.getMonoame().get(
                       degree).getDegree());
24                 polynomial3.addMonom(monom);
25             }
26         }
27         return polynomial3;
28     }
```

Operațiile de Derivare și Integrare pentru un singur polinom:
La integrare, constanta C a fost scrisă în metoda *actionPerformed* din clasa interfeței GUI, odată cu crearea butonului pentru această operație. O posibilă îmbunătățire

ar fi aceea ca la integrare să se returneze direct un String, acest lucru realizându-se prin convertirea tipului de date Polinom, la String, prin metoda suprascrisă toString() și să se returneze direct un String de forma: *return polynomialCopy + "+C"*. Însă s-a păstrat prima varianta, polinomul se convertește în String doar în clasa GUI, din motivul că se poate dori folosirea acestui rezultat de la integrală sub forma *Polynomial*.

```
1     public static Polynomial derivativePolynomial(Polynomial
         polynomial){
2
3          Polynomial polynomialCopy = new Polynomial();
4          for(int degree:polynomial.getMonoame().keySet()){
5               Monom monom = new Monom();
6               monom.setDegree(degree-1);
7               Double degreeCopy = monom.getDegree()*0.1;
8               monom.setCoefficient(polynomial.getMonoame().get(
                   degree).getCoefficient()*degree);
9               polynomialCopy.addMonom(monom);
10         }
11         return polynomialCopy;
12    }
13    public static Polynomial integratePolynomial(Polynomial
         polynomial){
14
15         Polynomial polynomialCopy = new Polynomial();
16         for(int degree:polynomial.getMonoame().keySet()){
17              Monom monom = new Monom();
18              monom.setDegree(degree+1);
19              monom.setCoefficient(polynomial.getMonoame().get(
                   degree).getCoefficient()/(degree+1.0));
20              polynomialCopy.addMonom(monom);
21         }
22         return polynomialCopy;
23    }
```

(d) Clasa *PolynomialRead*: În acestă clasă se găsește metoda readPolynomial, care primește inputul sub forma String introdus de la tastatură în program și îl modifică, parsând astfel, cu ajutorul unui regex, tot ce este în fața lui x, coeficientului de tipul Double, iar tot ce este după simbol de putere este considerat grad, fiind de tipul int. Acest regex utilizat simplifică foarte mult secționarea caraterelor și atribuirea lor diverselor argumente (în acest caz, grad și coeficient). Se atribuie treptat în funcție de caractere, valori gradelor și argumentelor unui monom (se seteaza cu setterele realizate în clasa Monom), urmând ca monomul realizat să fie introdus într-un polinom nou inițializat. Condiția de oprire este cea din while, pe baza Matcher-ului *matcher.find()*, când această conditie este nulă, înseamnă că am ajuns la finalul Stringului dat ca input și s-a completat polinomul cu toate monoamele respective. Clasa Matcher în Java face parte din pachetul java.util.regex și funcționează în colaborare cu clasa Pattern pentru operațiuni de potrivire a mode-

lului pe text folosind expresii regulate.

```
1  public class PolynomialRead {
2      private static final String CONST_REGEX =   "
          ([\\+-]?[1-9]*(\\.[0-9][0-9]*)?x\\^-?[2-9]+)
          |([\\+-]?[1-9]*(\\.[0-9][0-9]*)?x)
          |([\\+-]?[0-9]+(\\.[0-9][0-9]*)?)";
3      public static Polynomial readPolynomial(String
          inputPolynomial){
4          final Pattern pattern = Pattern.compile(CONST_REGEX,
              Pattern.MULTILINE);
5          final Matcher matcher = pattern.matcher(
              inputPolynomial);
6          Polynomial polynomial = new Polynomial();
7          while (matcher.find()) {
8              String groupPoly = matcher.group(0);
9              Monom monomPoly = new Monom();
10             Double coefficient = 0.0;
11             Integer degree = 0;
12             if(groupPoly.contains("x")) {
13                 coefficient = getCoefficient(groupPoly);
14                 if(groupPoly.contains("^")){
15                     degree = Integer.parseInt(groupPoly.
                          substring(groupPoly.indexOf("x") + 2))
                          ;
16                 }
17                 else{
18                     degree = 1;
19                 }
20             }
21             else{
22                  coefficient = Double.parseDouble((groupPoly)
                        );
23                  degree = 0;
24             }
25             monomPoly.setCoefficient(coefficient);
26             monomPoly.setDegree(degree);
27             polynomial.addMonom(monomPoly);
28         }
29         return polynomial;
30     }
```

(e) Clasa *TestJunitP*: se vor menționa la *Rezultate* caracteristicile acestei clase.

(f) Clasa *PolynomialCalculatorGUI*: S-a realizat o interfață grafică prietenoasă, în nuanțe verzi, ușor de utilizat. Are doua field-uri pentru introducerea celor doua polinoame și un field pentru afișarea rezultatului operației dorite. Sunt 6 butoane pentru selecția operației dorite. S-au implementat doua metode: verifyInput, care verifică ce caractere s-au introdus și pe baza unui regex, în cazul în care s-au in-

trodus caractere invalide se va afișa o eroare pe fereastră cu mesajul respectiv. S-a implementat și o metodă pentru verificarea field-ului gol, în cazul în care se introduce un polinom într-un field, iar în cel de-al doilea field nu se introduce niciun polinom se va afișa de astfel un mesaj de eroare pe ecran. De asemenea pentru cazurile speciale la împărțire *împărțirea la 0* și *împărțirea la un polinom cu un grad mai mare decât însuși deîmpărțitul*, se afișează mesaje de eroare pe ecran. Odată ce se afișează un mesaj de eroare pentru un input greșit, field-ul în care s-a introdus inputul greșit se curăță automat, iar field-ul de result, cu rezultat operației realizate anterior (dacă s-a realizat vreo operație), se curăță de asemenea.

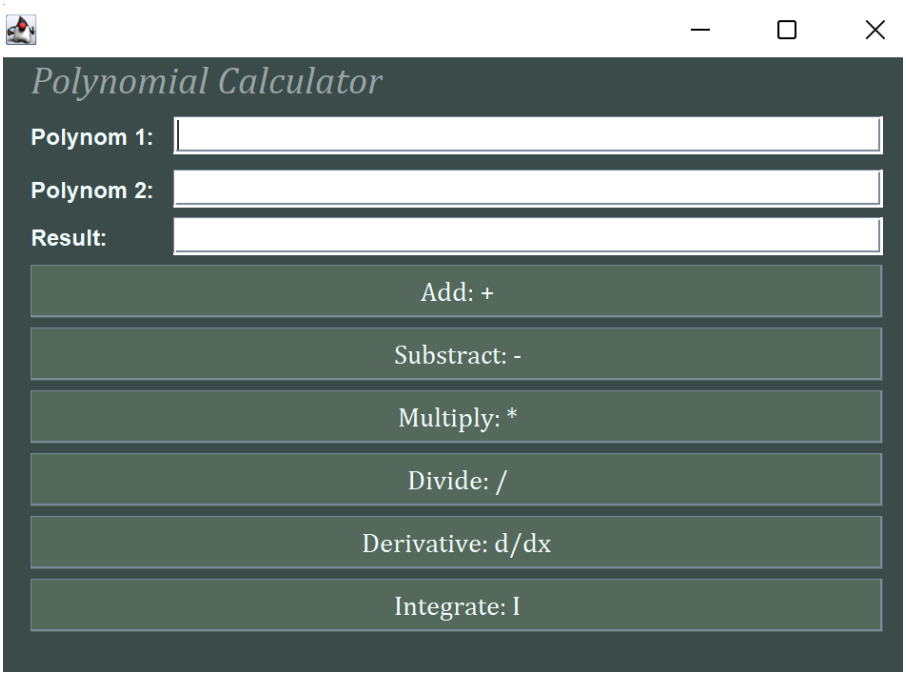

Figura 4: Interfața grafică pentru calculatorul de polinoame

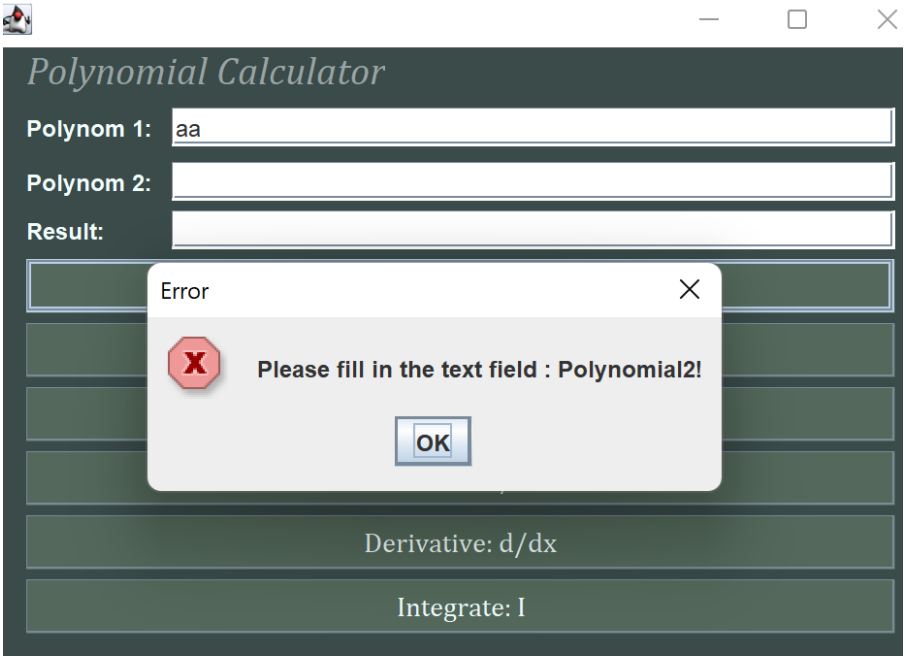

Figura 5: Afișarea unei eroare pentru introducerea caracterelor invalide

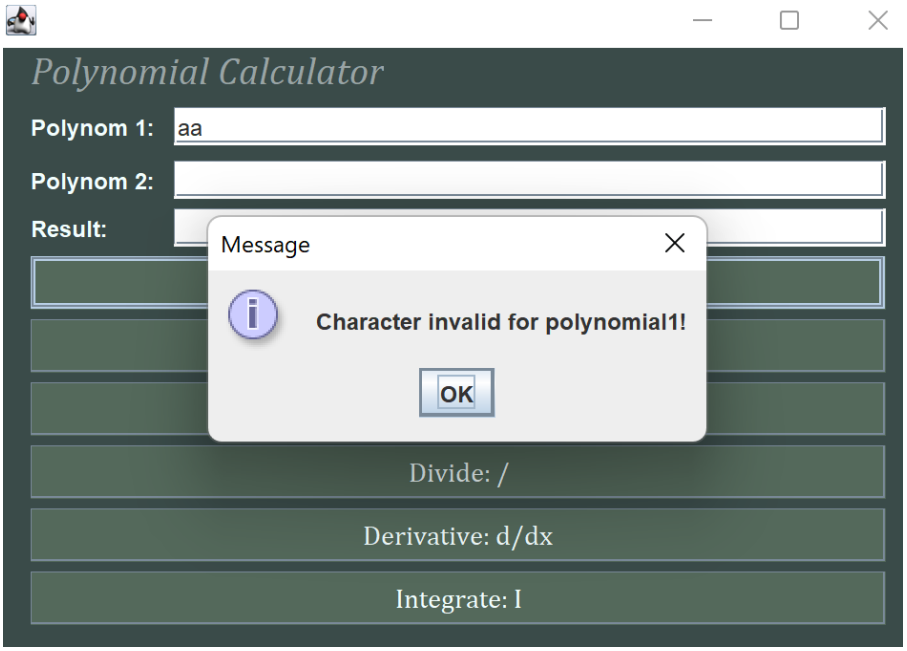

Figura 6: Afișarea unei eroare pentru field gol

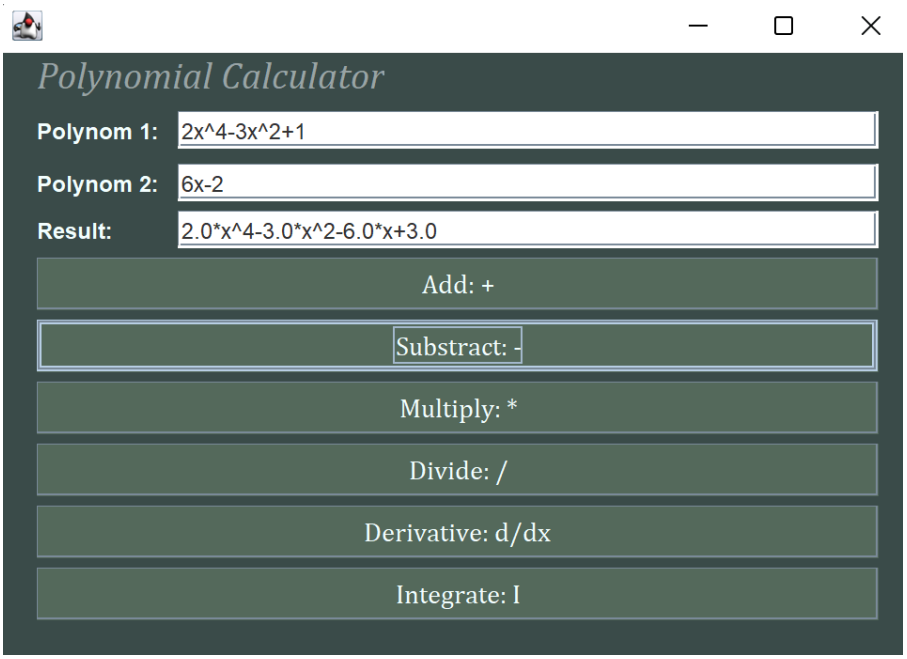

Figura 7: Afișarea unui rezultat pentru operația de scădere între două polinoame

5 Rezultate

Rezultatele se afisează în ultimul JTextField din interfață *resultTextField*. Pe baza informațiilor primite în câmpurile *Polynomial1* și *Polynomia2* se creează polinoamele, cu ajutorul metodei de citit String-urile introduse de la tastatatură *readPolynomial*. La apăsarea unui buton denumit după operația matematică dorită se realizează operația corespunzătoare, rezultatul fiind observat în căsuța de *Result*. S-a creat o metodă specială (verifyInput), care verifică dacă s-au introdus altfel de date, în afară de cele corecte pentru citirea polinomul (caractere precum litere, altele în afară de x, simboluri necorespunzătoare, introducerea a mai multor semne operaționale fără logică), cât și o metodă care să verifice dacă s-a introdus un polinom în cele doua căsuțe odată cu apăsarea butonului de operație. Se afișeaza mesaje de eroare în fereastră în oricare din cazuri.
Exemplu:
Un input este de forma: 2x^3+x-3, iar rezultatul care se va afișa este sub forma: c1*x^p1+c2*x^p2+...+a, unde c1, c2,... sunt considerați coeficienți, iar p1, p2,... grade/puteri ale monoamelor, a este monomul de grad 0. S-au ales la afișare parantezele ( ) pentru cazul în care se dorește introducere unor puteri negative, la afișare se vor observa mai ușor de către utilizator.
Sub reprezentarea matematică un polinom arată de forma:

$$P(x) = a_n x^n + \ldots + a_1 x + a_0$$

S-a creat o clasă de testare unitară Java, Clasa *TestJunitP*: În această clasă s-au implementat teste unitare pentru metodele create anterior în celelalte clase. S-a folosit Junit, fiind cea mai populară metodă de teste unitare pentru Java, unde sunt folosite dependențe Maven, adăugate în fișierul pom.xml. Exemplu:

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.9.2</version>
    <scope>test</scope>
</dependency>
```

S-au utilizat adnotări corespunzătoare modului de afișare și de realizare a testelor dorit: @AfterEach – denotes that the annotated method will be executed after each test method (previously @After) @BeforeAll – denotes that the annotated method will be executed before all test methods in the current class (previously @BeforeClass) @AfterClass)
De asemenea s-au utilizat aserții (assertions), care sunt utilizate pentru a verifica premisele corectitudinii codului în timpul dezvoltării și testării. Acestea permit programatorilor să introducă expresii care trebuie să fie adevărate într-o anumită etapă a programului și să genereze o excepție în cazul în care aceste premise nu sunt îndeplinite.
Exemplu de test cu Junit: Exemplu de test pentru adunare. S-au introdus doua polinoame, s-a inițializat rezultatul așteptat, iar pe baza metodei din clasa PolynomialOperation s-a obtinut rezultatul, care după verificare s-a constat că este la fel, deci testul s-a realizat cu SUCCES.

```
1           @org.junit.Test
2           public void testAddNumber1() throws ExceptionDivide {
3               System.out.print("Test Number 1 for Add Operation:\n"
                    );
4
5               Polynomial polynomial1 =  PolynomialRead.
                    readPolynomial("6x^4+3x^2-5x-8");
6               Polynomial polynomial2 =  PolynomialRead.
                    readPolynomial("2x^-2+5x-6");
7
8               Polynomial waitingResult = PolynomialRead.
                    readPolynomial("6x^4+3x^2+2x^-2-14");
9               Polynomial obtainedResult = PolynomialOperation.
                    addPolynomial(polynomial1,polynomial2);
10
11              boolean testVerify = obtainedResult.toString().equals
                    (waitingResult.toString());
12              Assert.assertTrue(testVerify);
13              numberOfTestsSuccess = numberOfTestsSuccess + 1;
14          }
15      }
```

La final pe baza adnotării @AfterClass și pe baza unei metode implementate s-a afișat un mesaj care conține numărul de teste încercate și numărul de teste realizate cu succes. —*They were executed X tests of which Y tests successfully*—

6 Concluzii

În cele din urmă, se poate spune că, deși, crearea unui aplicației pentru un calculator de polinoame poate părea la prima vedere o sarcină complicată, prin utilizarea tehnicilor de programare orientate pe Obiect, se poate obține o aplicație eficientă și utilă (mai ales când ne putem referi la o categorie de persoane care nu știu să rezolve corect operațiile pe polinoame, precum niște elevi de liceu pentru care subiectul este la început de predare și au nevoie de un mod constant de verificare sau validare a calculelor).
Se mai poate spune că este o aplicație utilă pentru o categorie largă de persoane, astfel că atât elevii de liceu, cât și studenții de la universitățiile de inginerie sau matematică pot apela la ajutorul acesteie pentru a calcula operațiile dorite. Totodată pentru un student la Calculatoare este un start în ceea ce înseamna proiectarea unui design cât mai corect, înțelegerea unor principii de programare și învățarea scrierii unui cod cât mai corect cu putință.
Utilizarea unui system de build precum Maven, în utilizarea arhitecturii MVC, dezvolta o aplicație ușor de menținut.

Îmbunătățiri ulterioare pot fi:

- Adăugarea de funcționalități suplimentare: extinderea funcționalității aplicației

prin adăugarea de operații suplimentare cu polinoame, cum ar fi: cel mai mare divizor comun a două polinoame, cel mai mic multiplu comun a două polinoame, evaluarea într-un punct dat sau găsirea rădăcinilor.

- Interfață grafică îmbunătățită: pentru a fi mai intuitivă și ușor de utilizat. Adăugarea unor opțiuni de personalizare, cum ar fi selectarea culorilor temei, prezența unor butoane cu cifre pentru a face diferită introducerea polinomului (de data aceasta, nu numai de la tastatură).
- O testare mai riguroasă a aplicației pe un număr mai mare de cazuri.
- O posibilă îmbunătățire priviind organizarea codului.

7 Bibliografie (Webografie)

(a) Cursuri OOP Anul2, Semestrul 1

(b) https://dsrl.eu/courses/pt/materials/lectures/

(c) https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Polynomial.java.html

(d) https://stackoverflow.com

(e) https://regex101.com/

(f) https://www.overleaf.com/learn/latex

(g) https://libguides.eur.nl/overleaf/lists-tables-images-labelling