

DOCUMENTATIE

TEMA *1*

NUME STUDENT: Ureche Simona Elena
GRUPA: 30224

CUPRINS

1.	Obiectivul temei.....	3
2.	Analiza problemei, modelare, scenarii, cazuri de utilizare	4
3.	Proiectare	5
4.	Implementare	9
5.	Rezultate	13
6.	Concluzii.....	14
7.	Bibliografie	14

1. Obiectivul temei

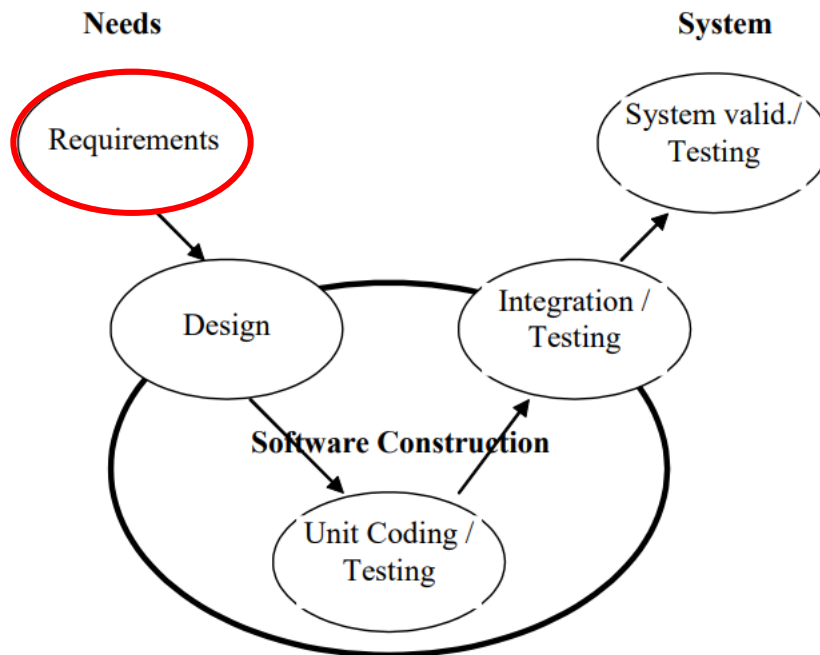
(i) Obiectivul principal

Dezvoltarea unei aplicații cu interfață grafică dedicată manipulării și efectuării operațiilor esențiale asupra polinoamelor. Aceste operații includ adunarea, scăderea, înmulțirea, împărțire, derivarea și integrarea polinoamelor.

(ii) Obiective secundare

Obiectiv secundar	Descriere	Capitol
<i>Cerințe funcționale/nonfuncționale</i>	Definesc comportamentul și capacitățile sistemului. Acestea pot include operațiuni specifice, funcționalități, servicii sau caracteristici pe care sistemul trebuie să le ofere pentru a satisface nevoile utilizatorilor și a atinge obiectivele stabilite. Scenariile oferă o perspectivă detaliată asupra modului în care utilizatorii și sistemul colaborează pentru a realiza sarcini.	2
<i>Proiectarea orientată pe obiecte(POO)</i>	Prezentarea arhitecturii generale a aplicației, evidențiind modul în care conceptele OOP au fost aplicate pentru a organiza și structura codul.	3
<i>Împărțirea în pachete și clase</i>	Modul în care clasele și alte resurse sunt grupate în pachete logice pentru a organiza și gestiona proiectul.	3
<i>Structuri de date și algoritmi</i>	Utilizate în cadrul aplicației pentru a rezolva diferite probleme sau sarcini.	3
<i>Interfete grafice definite</i>	Folosirea tehnicii Model View Controller pentru realizarea unui Graphical User Interface.	3
<i>Descrierea claselor și metodelor</i>	Semnăturile și descrierea claselor și metodelor relevante din cadrul calculatorului de polinoame.	4
<i>Implementarea interfeței utilizator</i>	Modul în care interfața utilizator a fost proiectată și implementată pentru a facilita interacțiunea utilizatorului cu manipularea polinoamelor.	4
<i>Testarea unitară</i>	Scenarii de testare a operațiilor cu polinoame, folosind ca funcționalitatea Junit Test.	5

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



Scenariu de Utilizare: Adăugare Polinoame

Actor Principal: Utilizatorul

Cerințe Funcționale:

- Calculatorul de polinoame trebuie să permită utilizatorilor să introducă polinoame.
- Calculatorul de polinoame trebuie să permită utilizatorilor să selecteze operația matematică.
- Calculatorul de polinoame trebuie să adauge două polinoame

Cerințe Non-Funcționale:

- Calculatorul de polinoame trebuie să fie intuitiv și ușor de utilizat de către utilizator.
- Calculatorul de polinoame ar trebui să ofere rezultatele într-un timp rezonabil, chiar și pentru polinoame complexe sau operații extinse.
- Aplicația ar trebui să funcționeze fără erori și să ofere rezultate precise în toate situațiile, fără să se blocheze sau să se comporte imprevizibil.

Scenariul Principal de Succes:

- Utilizatorul introduce cele două polinoame în interfața grafică.
- Utilizatorul selectează operația de "adunare".
- Utilizatorul apasă butonul "calculează".
- Calculatorul de polinoame efectuează adunarea celor două polinoame și afișează rezultatul.

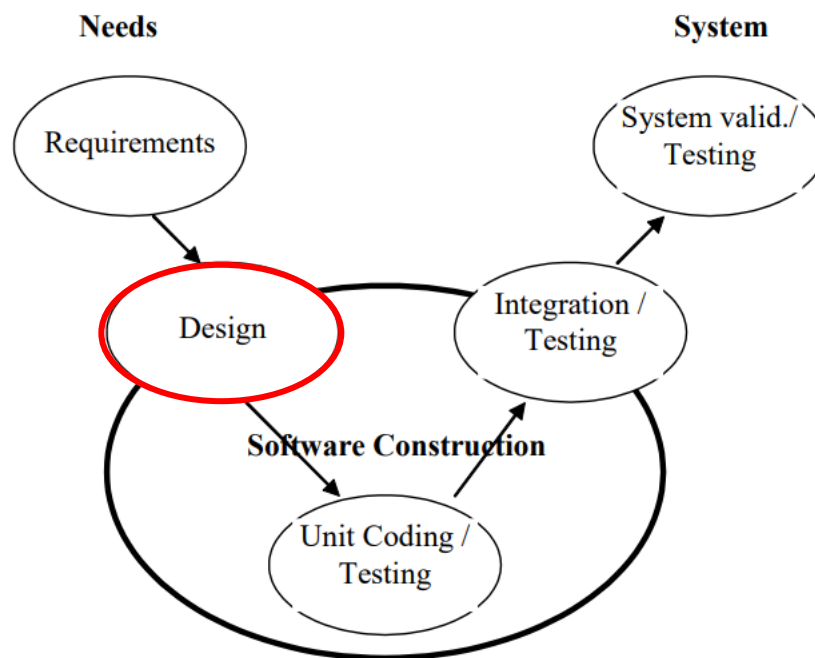
Secvență Alternativă: Polinoame Incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu două sau mai multe variabile).
- Scenariul revine la pasul 1.

3. Proiectare

(i) Decizii de proiectare

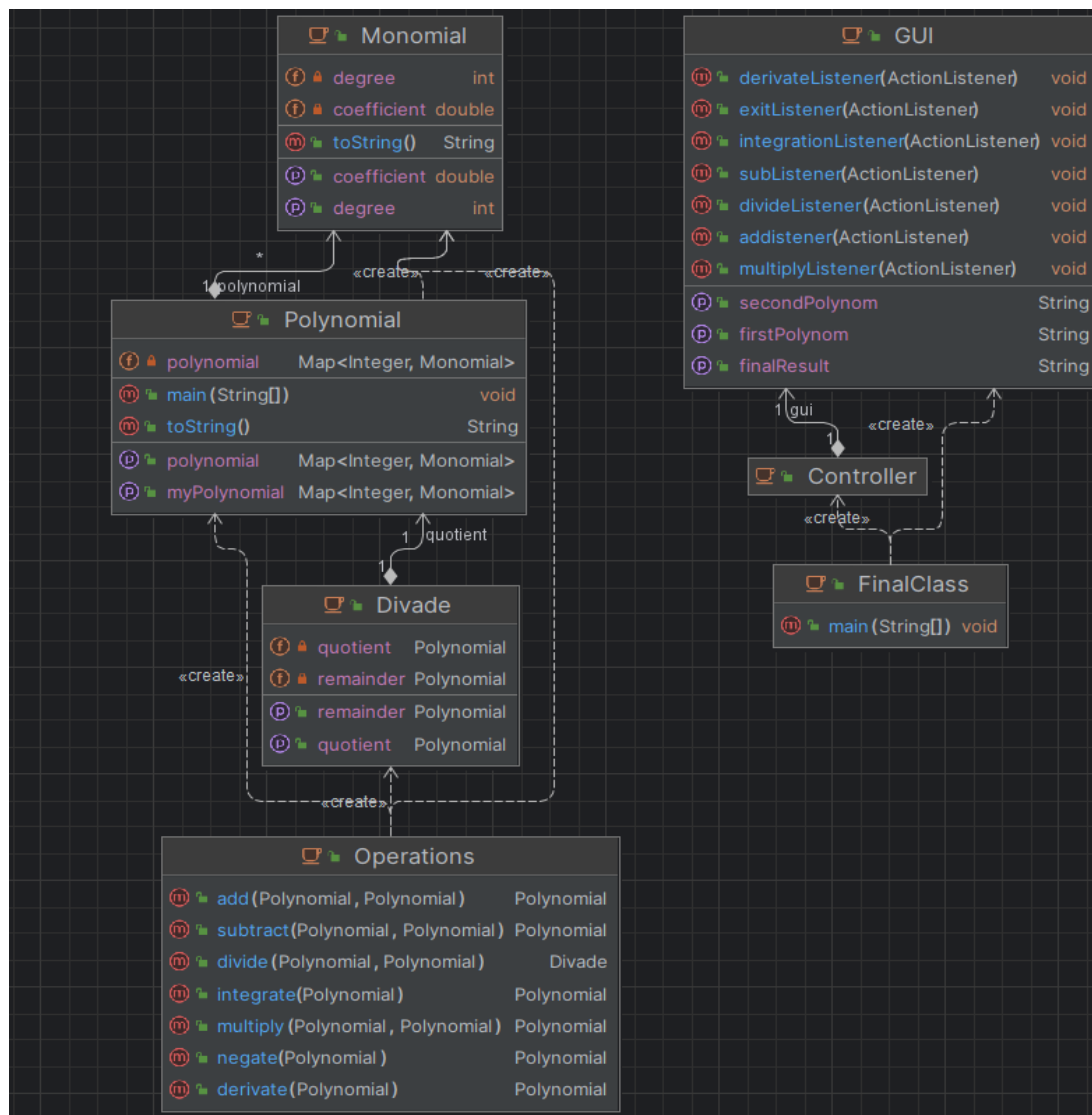
Aplicația pentru calculatorul de polinoame este structurată folosind conceptele de programare orientată pe obiecte (OOP) pentru a organiza și structura codul într-un mod coerent și eficient. Vom separa atât componentele logice cât și modelul de date de interfață de control, ceea ce face mai ușoară gestionarea și întreținerea codului.



(ii) Împărțirea în pachete

- Pachetul GUI: Aici se va afla clasa “GUI”, care gestionează interfața grafică a aplicației.
- Pachetul Logic: Acest pachet conține clasa “Operations”, care conține implementările operațiilor matematice pe polinoame, și clasa “Controller”, care servește drept intermediar între interfața grafică și operațiile efective.

- Pachetul Model: Aici vor fi definite clasele 'Polynomial' și 'Monomial', care modelează structura și comportamentul polinoamelor și monoamelor.



(iii) Împărțirea în clase

- Clasa GUI: Această clasă gestionează aspectele legate de interfața grafică a aplicației. Ea definește elementele vizuale și interacțiunile utilizatorului cu acestea, precum și logica necesară pentru a interpreta și a răspunde la acțiunile utilizatorului.

- Clasa Operations: Această clasă conține implementările operațiilor matematice asupra polinoamelor. Aici sunt definite metodele pentru adunare, scădere, înmulțire, împărțire, derivare și integrare, utilizate pentru manipularea și calculul polinoamelor.

- Clasa Controller: Această clasă acționează ca un intermediar între interfața grafică și logica de afaceri. Ea gestionează comunicarea și transmiterea datelor între interfața grafică și operațiile efective, asigurându-se că acțiunile utilizatorului sunt interpretate corect și că rezultatele sunt afișate în mod corespunzător.
- Clasa Monomial: Această clasă definește structura și comportamentul unui monom, inclusiv gradul și coeficientul acestuia. Este utilizată pentru a reprezenta fiecare componentă individuală a unui polinom.
- Clasa Polynomial: Această clasă reprezintă un polinom ca o colecție de monoame. Ea conține metode pentru a efectua operații pe polinoame, precum și pentru a gestiona și manipula componente individuale.

(iv) Structuri de date

```
Map<Integer, Monomial> polynomial;
```

- Map: Este o interfață în Java care stochează perechi de cheie-valoare. În acest caz cheile sunt de tip `Integer`, reprezentând gradele monomilor, iar valorile sunt obiecte de tip `Monomial`, reprezentând monomul corespunzător gradelor respective.
- Integer: Aceasta este cheia map-ului și reprezintă gradul monomului în polinom. Fiecare grad este unic și este asociat cu un singur monom.
- Monomial: Este clasa care reprezintă un monom în cadrul polinomului. Aceasta conține doi membri: coeficientul și gradul monomului. De asemenea mai exista

(v) Interfața grafică

- Bucățița de proiect cu care utilizatorul va interacționa. Vor fi afișate 3 câmpuri de tip text care permit utilizatorului să introducă polinoamele sau să vizualizeze rezultatele. De asemenea sunt afișate și 7 butoane (addButton, subButton, multiplyButton, divideButton, derivateButton, integrationButton, exitButton), care permit utilizatorului să efectueze diferite operații matematice pe polinoame dar și să închidă fereastra interfeței grafice. Fiecare buton are un text asociat și este configurat pentru a răspunde la evenimente de click.

Polynom Calculator

—

□

×

Polynomial Calculator

First Polynomial =

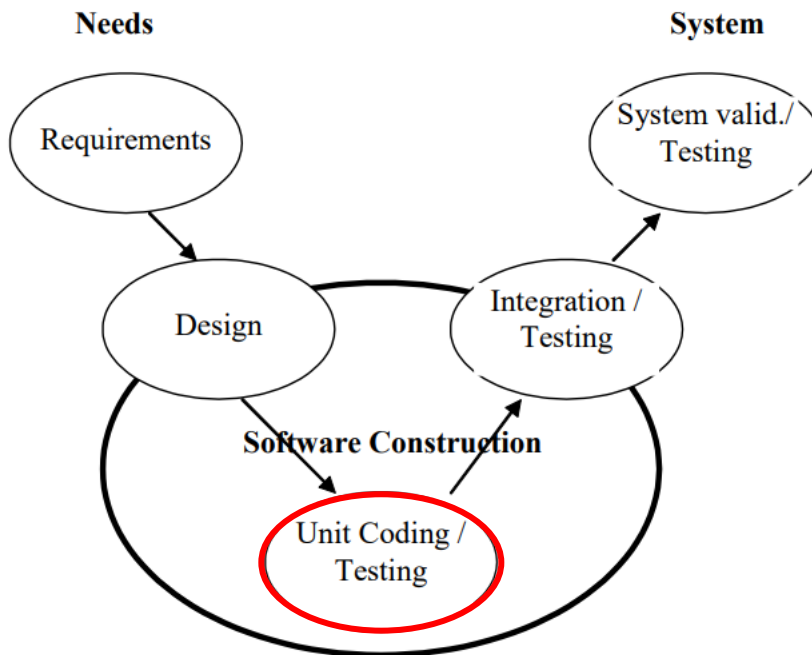
Second Polynomial =

Add	Divide
Substract	Derivate
Multiplicate	Integrate

Final Result =

Exit

4. Implementare



(i) Descrierea claselor și metodelor

(i.1) Clasa GUI

Clasa "GUI" este responsabilă pentru crearea și gestionarea interfeței grafice a aplicației. Aceasta conține următoarele metode:

- Metoda "getFirstPolynom()" : returnează șirul de caractere din câmpul text asociat primului polinom.

- Metoda "getSecondPolynom()" : returnează șirul de caractere din câmpul text asociat celui de-al doilea polinom.

- Metoda "setFinalResult(String result)" : setează textul rezultatului final în câmpul text corespunzător.

- Metodele "addlistener", "subListener", "multiplyListener", "divideListener", "derivateListener", "integrationListener", "exitListener": adaugă ascultători pentru evenimentele de acțiune generate de butoanele pentru operațiile matematice și butonul de ieșire.

(i.2) Clasa Monomial

Clasa "Monomial" reprezintă un monom din algebra polinomială și conține următoarele metode și constructor:

- Metoda "getDegree()": returnează gradul monomului.
- Metoda "setDegree(int degree)": setează gradul monomului la valoarea specificată.
- Metoda "getCoefficient()": returnează coeficientul monomului.
- Metoda "setCoefficient(double coefficient)": setează coeficientul monomului la valoarea specificată.
- Metoda "toString()": returnează o reprezentare sub formă de șir de caractere a monomului, în formatul "coeficientx^grad".

(i.3) Clasa Polynomial

Clasa "Polynomial" reprezintă un polinom și include următoarele metode și constructor:

- Constructorul "Polynomial(String inputValue)" este responsabil pentru parsarea unui șir de intrare care reprezintă un polinom și transformarea sa într-o structură de date internă pentru utilizarea ulterioară în calcul. Aplicând o expresie regulată, constructorul identifică monomiile din șirul de intrare, folosind un model care acoperă toate formele posibile ale unui monom, inclusiv coeficienții, exponențele și variabila x.

```
21 public Polynomial(String inputValue) {
22     Map<Integer, Monomial> result = new HashMap<>();
23
24     Pattern p = Pattern.compile("((-?\\d+(?=[x]))?(-?[x])(\\^((-?\\d+)?)|((-?[x])|((-?\\d+))))");
25     inputValue = inputValue.replaceAll(regex: "\\s", replacement: "");
26     inputValue = inputValue.replaceAll(regex: "\\*", replacement: "");
27     Matcher m = p.matcher(inputValue);
28     double coefficient = 0;
29     int exponent = 0;
30
31     //search for each monomial potential using the pattern
32     while (m.find()) {
33         if (m.group(3) != null && m.group(2) != null) { //the monomial of the form ax^b
34
35             exponent = (m.group(5) != null ? Integer.parseInt(m.group(5)) : 1);
36             coefficient = Integer.parseInt(m.group(2));
37         } else if (m.group(3) != null && m.group(2) == null) { ... } else if (m.group(3) == null && m.group(2) == null) { //the cons
45             coefficient = Integer.parseInt(m.group(4));
46         }
47         result.put(exponent, new Monomial(exponent, coefficient));
48         coefficient = 0;
49         exponent = 0;
50     }
51     this.polynomial = result;
52 }
```

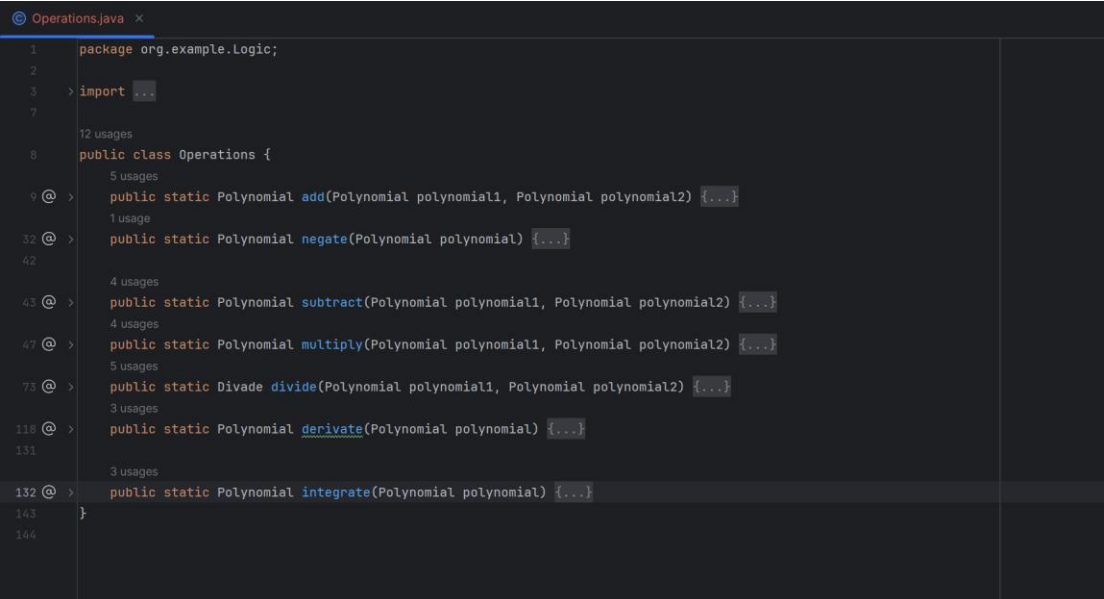
- Metoda "getMyPolynomial()" : returnează map-ul de monoame care reprezintă polinomul.

- Metoda "setPolynomial(Map<Integer, Monomial> polynomial)": setează map-ul de monoame al polinomului la valoarea specificată.
- Metoda "toString()" : returnează o reprezentare sub formă de șir de caractere a polinomului.

(i.4) Clasa Operations

Clasa "Operations" conține diverse operații matematice efectuate asupra polinoamelor și include următoarele metode:

- Metoda "add(Polynomial polynomial1, Polynomial polynomial2)": returnează suma a două polinoame.
- Metoda "negate(Polynomial polynomial)": returnează negativul unui polinom dat.
- Metoda "subtract(Polynomial polynomial1, Polynomial polynomial2)": returnează diferența dintre două polinoame; folosește metoda „negate” pentru a nega cel de-al doilea polinom, iar apoi face suma.
- Metoda "multiply(Polynomial polynomial1, Polynomial polynomial2)": returnează produsul dintre două polinoame.
- Metoda "divide(Polynomial polynomial1, Polynomial polynomial2)": returnează atât câtul, cât și restul împărțirii.
- Metoda "derivate(Polynomial polynomial)": returnează derivata a unui polinom



```

1 package org.example.Logic;
2
3 > import ...
7
12 usages
8 public class Operations {
9     5 usages
10     @ > public static Polynomial add(Polynomial polynomial1, Polynomial polynomial2) {...}
11     1 usage
32 @ > public static Polynomial negate(Polynomial polynomial) {...}
42
43 @ > public static Polynomial subtract(Polynomial polynomial1, Polynomial polynomial2) {...}
44     4 usages
47 @ > public static Polynomial multiply(Polynomial polynomial1, Polynomial polynomial2) {...}
48     5 usages
73 @ > public static Divide divide(Polynomial polynomial1, Polynomial polynomial2) {...}
74     3 usages
118 @ > public static Polynomial derivate(Polynomial polynomial) {...}
131
132 @ > public static Polynomial integrate(Polynomial polynomial) {...}
143
144 }

```

(i.5) Clasa Divide

Clasa "Divide" este o clasă simplă care reprezintă rezultatul unei operații de împărțire a doi polinoame și include următoarele componente:

- Constructorul "Divade(Polynomial quotient, Polynomial remainder)".
- Metoda "getQuotient()": returnează câtul rezultat din împărțire.
- Metoda "getRemainder()": returnează restul rezultat din împărțire.

(i.6) Clasa Controller

Clasa "Controller" este responsabilă de gestionarea interacțiunilor între interfața grafică și operațiile definite în logică. Ea include următoarele componente:

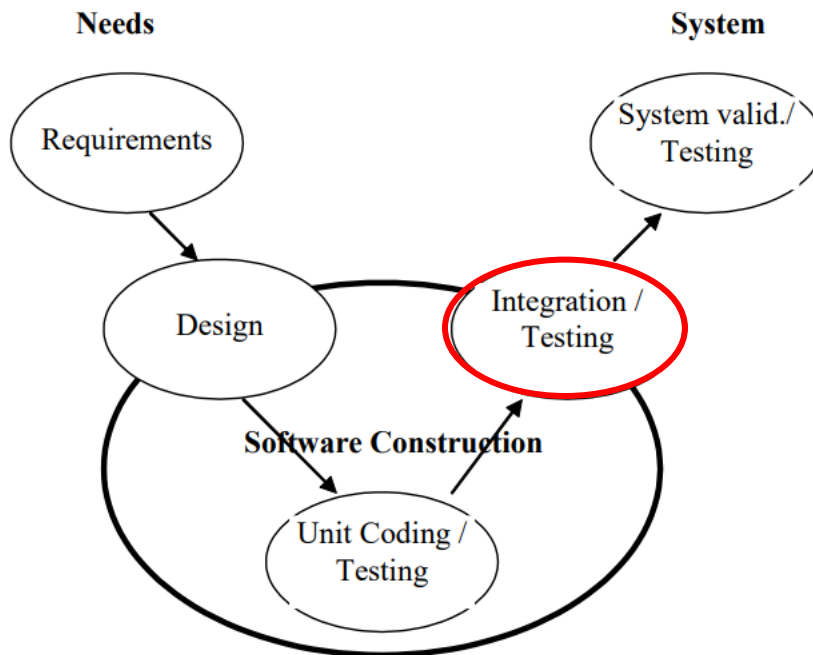
- Clasele interne ActionListener (AddListener, SubListener, MultiplyListener, DeriveListener, IntegrateListener, DivideListener, ExitListener): aceste clase interne implementează interfața ActionListener pentru a asculta evenimentele generate de butoanele din interfața grafică. Fiecare clasă ascultă un anumit tip de acțiune (adunare, scădere, înmulțire, derivare, integrare, împărțire, ieșire) și execută operația corespunzătoare.

```

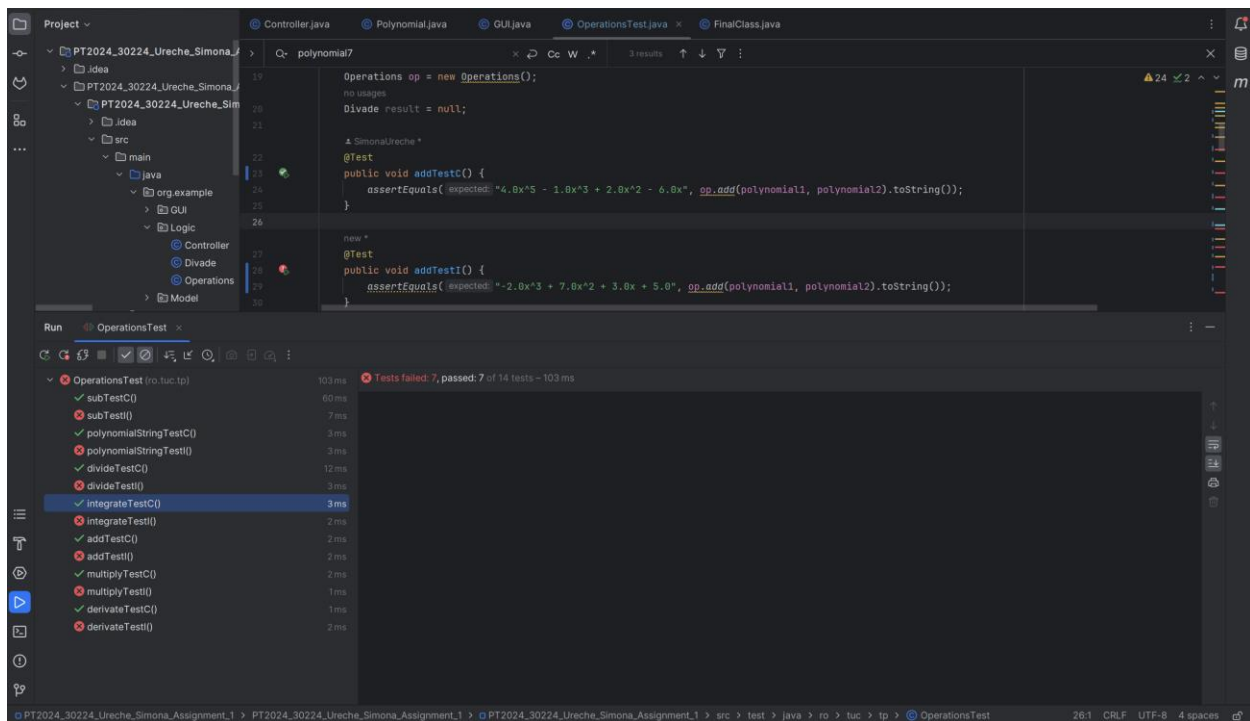
10 public class Controller {
11     17 usages
12     private GUI gui;
13
14     1 usage
15     @ public Controller(GUI gui) {
16         this.gui = gui;
17         gui.addListener(new AddListener());
18         gui.subListener(new SubListener());
19         gui.multiplyListener(new MultiplyListener());
20         gui.divideListener(new DivideListener());
21         gui.derivateListener(new DeriveListener());
22         gui.integrateListener(new IntegrateListener());
23         gui.exitListener(new ExitListener());
24     }
25
26     1 usage
27     > public class AddListener implements ActionListener {...}
28
29     1 usage
30     public class SubListener implements ActionListener {
31         @Override
32         public void actionPerformed(ActionEvent e) {...}
33     }
34
35     1 usage
36     > public class MultiplyListener implements ActionListener {...}

```

5. Rezultate



Fișierul "OperationsTest.java" conține o suită de teste pentru operațiile definite în logică. Aceste teste verifică corectitudinea operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare a polinoamelor. Fiecare test are ca scop compararea rezultatelor obținute prin apelarea metodelor din clasa "Operations" cu rezultatele așteptate.



- Am realizat 14 teste, 7 pentru a demonstra corectitudinea operatiilor, iar 7 gresite intentionat.

6. Concluzii

Lucrând la acest proiect, am avut ocazia să învăț și să experimentez diverse aspect noi cum ar fi lucrul cu expresii REGEX pentru a analiza și procesa polinoamele din șiruri de caractere, realizarea de teste unitare folosind Junit, dar și îmbunătățirea lucrului cu structura de date de tip Map.

Eventuale îmbunătățiri s-ar putea aduce la nivelul interfeței grafice dar și la nivelul algoritmului de parsare al polinomului, astfel încât să poată gestiona mai multe formate de intrare a polinomului. De asemenea, s-ar putea adauga suport pentru operații avansate precum factorizarea și determinarea rădăcinilor

7. Bibliografie

- Swing : <https://docs.oracle.com/javase/tutorial/uiswing/index.html>
- Junit : <https://www.baeldung.com/junit-5>
- Regex : <https://regexr.com/>
- Name-Conventional: <https://google.github.io/styleguide/javaguide.html>
- UML: <https://www.jetbrains.com/help/idea/class-diagram.html>