

# **DOCUMENTATIE**

## **TEMA 1**

### **Calculator de Polinoame**

NUME STUDENT: BUDACĂ ALEXANDRA  
GRUPA: 30224

## CUPRINS

|   |    |
|---|----|
| 1. Obiectivul temei.....  | 3  |
| 2. Analiza problemei, modelare, scenarii, cazuri de utilizare ..... | 3  |
| 3. Proiectare.....  | 8  |
| 4. Implementare .....   | 11 |
| 5. Rezultate .....  | 14 |
| 6. Concluzii.....   | 15 |
| 7. Bibliografie.....  | 15 |

## 1. Obiectivul temei

Obiectivul principal al temei cu numarul 1 este implementarea unui calculator, ce realizeaza operatii pe polinoame, si are o interfata grafica usor de folosit pentru a permite utilizatorului introducerea polinoamelor, selectarea operatiei dorite si vizualizarea rezultatului.

Obiectivele secundare ale acestei teme sunt:

- Structurarea proiectului in pachete: GUI, DataModels, BusinessLogic
- Realizarea arhitecturii grafice folosind JavaSwing
- Realizarea claselor necesare functionarii calculatorului de polinoame (clasa Operations din BusinessLogic; clasele Monomial, Polynomial din DataModels)
- Utilizarea notiunilor de programare orientate pe obiect, respectand paradigmele acesteia
- Implementarea operatiilor in clasa Operations (adunare, scadere, inmultire, impartire, derivare, integrare)
- Folosirea expresiilor regulate (regex) pentru recunoasterea polinomului (pattern matching)
- Testara unitara folosind JUnit

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

In analiza problemei am pornit de la cerintele problemei, iar primul pas a fost intelegerea problemei, pentru a putea defini ulterior domeniul acesteia. Astfel am putut analiza problema si sa o descompun in subprobleme individuale pentru construirea solutiei finale.

Pentru acest calculator de polinoame trebuie sa intelegem, in primul rand, ce este un monom, ce este un polinom, operatiile care se pot efectua asupra acestora, cum functioneaza operatiile si cum le putem executa.

Asadar, pentru inceput vom construi clasele Monomial si Polynomial, apoi clasa Operations care contine metodele corespunzatoare operatiilor efectuabile asupra polinoamelor : adunare, scadere, inmultire, impartire, derivare, integrare. Mai apoi vom construi interfata grafica si functionalitatile acesteia. In final, ne ramane de testat fiecare operatie in parte, pentru a asigura corectitudinea.

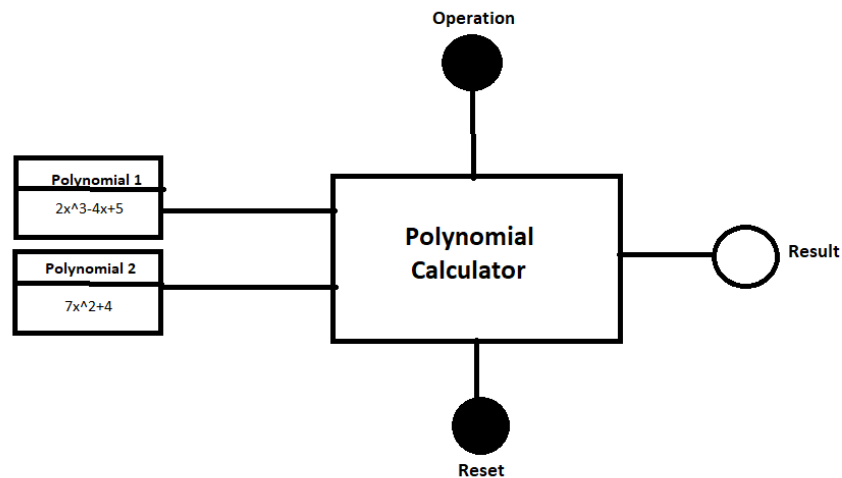
Ca si cerinte functionale ale aplicatiei, am identificat urmatoarele :

- Utilizatorul trebuie sa introduca minim un polinom pentru operatiile de derivare si integrare, si 2 polinoame pentru restul operatiilor
- Utilizatorul trebuie sa poata selecta operatia dorita
- La apasarea butonului de egal ("="), rezultatul ar trebui sa apara pe ecran
- Coeficientii polinomelor la introducere ar trebui sa fie numere reale, insa la aplicarea operatiilor de impartire si integrare, rezultatul va putea avea coeficienti de tip real
- Tratarea erorilor in urma utilizarii incorecte a aplicatiei: utilizatorul poate introduce polinoame invalide; utilizatorul introduce prea multe sau prea putine polinoame pentru operatia selectata

Cerintele non-functionale identificate sunt :

- Complexitatea design-ului interfetei grafice
- Flexibilitatea aplicatiei

Ulterior, am creat diagrama de use-case a aplicatiei pentru a proiecta o viziune asupra proiectului.

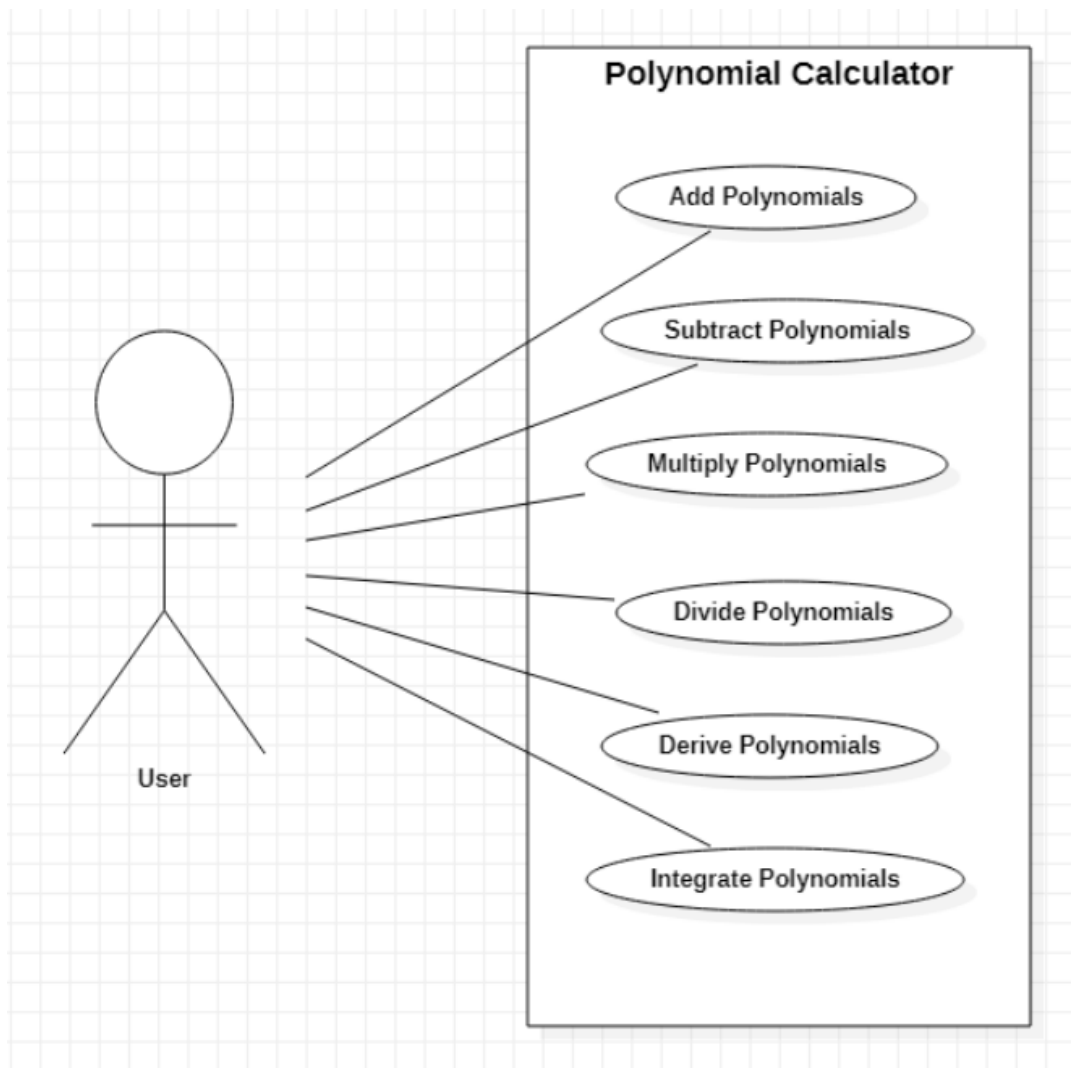


Calculatorul de polinoame este privit ca o cutie neagra, care primeste ca intrari cele 2 polinoame, si operatia care se doreste a se efectua, dupa care pe iesire se va afisa rezultatul. De asemenea, este prevazut cu un buton de Reset, ca la finalul efectuarii unei opeatii, utilizatorul sa poata introduce alte 2 polinoame pentru a efectua operatii.

### Modelare scenarii :

Exista multe scenarii de ulilizare ale aplicatiei, asadar trebuie luate in considerare toate cazurile posibile, in special cele care ar putea merge prost. Acestea trebuie gandite din perspectiva utilizatorului de aplicati, nu al proiectantului. In cazul acesarii unor scenarii "interzise", utilizatorul ar trebui informat de acest lucru cu un mesaj.

Mai jos este prezentata interactiunea ulilizatorului cu aplicatia.



### Cazuri de utilizare :

Exista 2 cazuri diferite de utilizare. Cele care implica operatia pe un polinom (derivare, integrare), si cele care implica operatia pe 2 polinoame (resul operatiilor).

- Operatii care se efectueaza pe 2 polinoame (adunare, scadere, inmultire, impartire)

Scenariul de succes: Utilizatorul realizeaza operatia dorita.

→ 1. Utilizatorul introduce in interfata grafice 2 polinoame

→ 2. Utilizatorul selecteaza una din urmatoarele operatii (adunare, scadere, inmultire, impartire).  
Polinomul impartitor in cazul impartirii este diferit de 0!

→ 3. Calculatorul de polinoame realizeaza operatia selectata si afiseaza rezultatul

Scenariul de esec (alternativ celui de success): Polinoamele introduse sunt incorecte!

→ Utilizatorul introduce unul sau doua polinoame incorect.

→ Eroarea corespunzatoare este semnalata ("Cannot divide by 0!" || "This is not a polynomial!")

→ Scenariul se intoarce la pasul 1.

➤ Operatii care se efectueaza pe un polinom (derivare, integrare)

Scenariul de succes: Utilizatorul realizeaza operatia dorita.

→ 1. Utilizatorul introduce in interfata grafice cel putin un polinom. In cazul introducerii a doua polinoame calculatorul va afisa rezultatul operatiei pentru primul polinom introdus.

→ 2. Utilizatorul selecteaza una din urmatoarele operatii (derivare, integrare)

→ 3. Calculatorul de polinoame realizeaza operatia selectata si afiseaza rezultatul

Scenariul de esec (alternativ celui de success): Polinoamele introduse sunt incorecte!

→ Utilizatorul introduce unul sau doua polinoame incorect.

→ Eroarea corespunzatoare este semnalata. ("This is not a polynomial!")

→ Scenariul se intoarce la pasul 1

### 3. Proiectare

Am ales sa fac implemenarea cu modelul de pachete GUI → DataModels → BusinessModel ceea ce presupune ca in prima faza vor fi implemenate cele 3 pachete mentionate anterior, in plus adaugandu-se pachetul pentru testarea unitara.

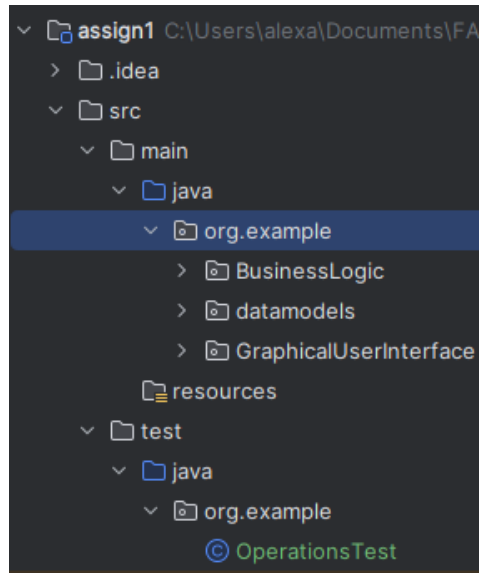
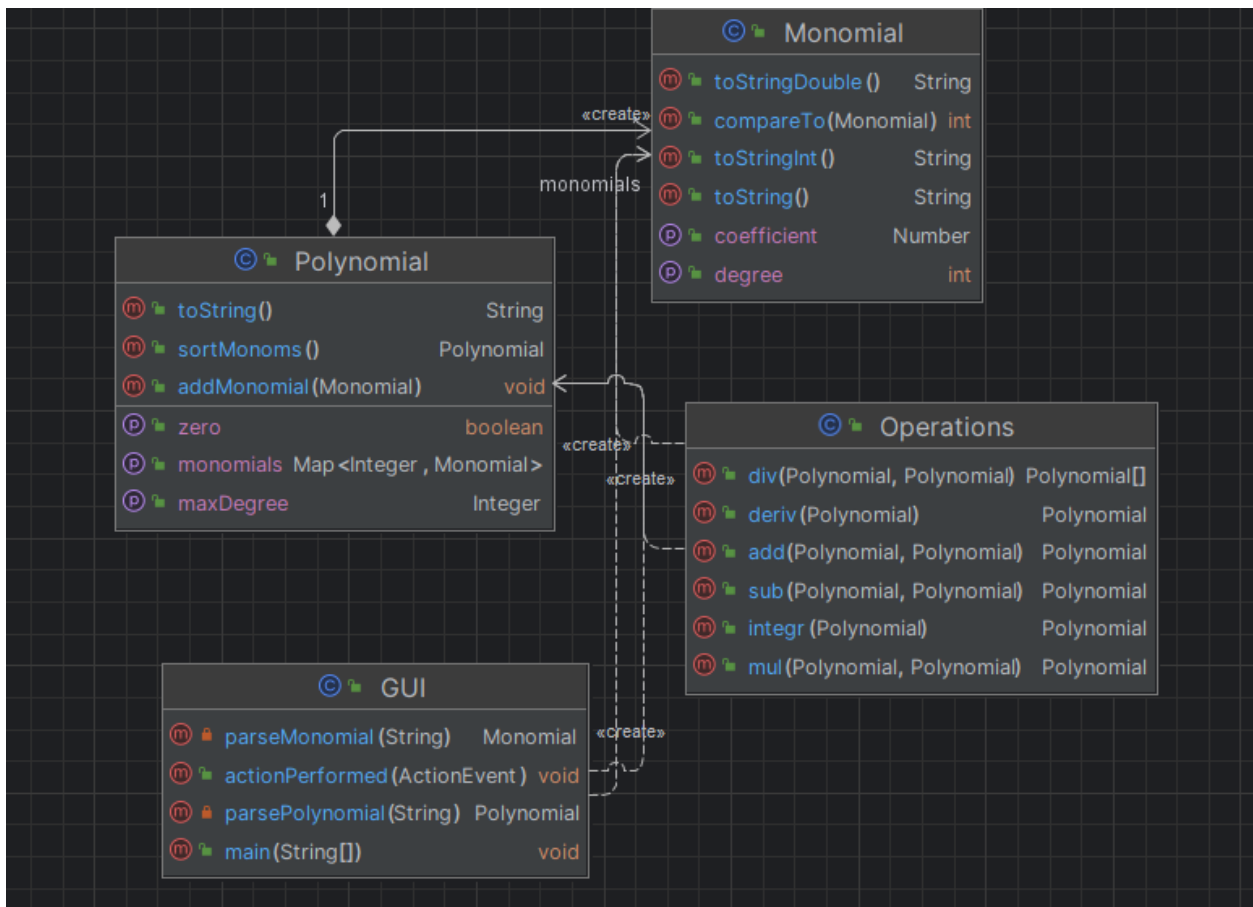


Diagrama UML de clase





**Clasa Monomial** este clasa fundamentala a aplicatiei si defineste structura de monom. Un monom este un singur termen dintr-un polinom si contine un coeficient si un grad. (ex:  $2x^3$ )

Pentru coeficient am ales tipul Number pentru ca vom avea nevoie atat de coeficienti intregi (Integer), cat si reali(Double). Asadar, pentru a ne usura munca l-am facut la inceput de tip Number, superclasa pentru Integer si Double.

In aceasta clasa se gasesc 3 metode de toString, astfel metodele toStringInteger, respectiv toStringDouble sunt metode de afisare pentru tipul specificat, iar metoda principala toString decide pe care din cele doua sa le foloseasca. Astfel, vom apela direct metoda toString, indiferent de tipul coeficientului.

Tot aici se regaseste si metoda compareTo, intrucat aceasta clasa implementeaza interfata Comparable, pentru a putea sorta monoamele descrescator, in functie de grad.

**Clasa Polynomial** este clasa care sta la baza definirii operatiilor, Un polinom este o lista de monoame, asadar acestei clase i-am atasat un Map pentru a retine monoamele din care este alcatuit.

Aceasta clasa implementeaza metodele: toString() - afiseaza polinomul de tip String; sortMonoms() – sorteaza Map-ul care contine momoamele, in ordinea descrescatoare a gradului; addMonomials() – adauga un obiect de tip Monom in Map-ul polinomului ; zero() – verifica daca polinomul ests zero; maxDegree() – calculeaza gradul maxim din Map-ul de monoame si il returneaza.

**Clasa Operations** implementeaza operatiile propriu-zise: adunare, scadere, inmultire, impartire, derivare, integrare.

- Adunarea a doua polinoame:

$$P(X) + Q(X) = (a_n + b_n) * X^n + (a_{n-1} + b_{n-1}) * X^{n-1} + \dots + (a_1 + b_1) * X + (a_0 + b_0)$$

- Scaderea a doua polinoame:

$$P(X) - Q(X) = (a_n - b_n) * X^n + (a_{n-1} - b_{n-1}) * X^{n-1} + \dots + (a_1 - b_1) * X + (a_0 - b_0)$$

- Inmultirea a doua polinoame:

$$P(X) = 3 * X^2 - X + 1$$

$$Q(X) = X - 2$$

The result of multiplying the two polynomials is:

$$P(X) * Q(X) = 3 * X^3 - X^2 + X - 6 * X^2 + 2 * X - 2 = 3 * X^3 - 7 * X^2 + 3 * X - 2$$

- Pentru operatia de impartire a fost folosit urmatorul algoritim

```
function n / d is
  require d ≠ 0
  q ← 0
  r ← n           // At each step n = d × q + r

  while r ≠ 0 and degree(r) ≥ degree(d) do
    t ← lead(r) / lead(d)    // Divide the leading terms
    q ← q + t
    r ← r - t × d

  return (q, r)
```

- Derivarea unui polinom :

$$\frac{d}{dx}(a_n * X^n + a_{n-1} * X^{n-1} + \dots + a_1 * X + a_0) = n * a_n * X^{n-1} + (n-1) * a_{n-1} * X^{n-2} + \dots + a_1$$

- Integrarea unui polinom

$$\int a_n * X^n + a_{n-1} * X^{n-1} + \dots + a_1 * X + a_0 = \int a_n * X^n dx + \int a_{n-1} * X^{n-1} dx + \dots + \int a_1 * X dx + \int a_0 dx$$

where:

$$\int a_n * X^n dx = a * \frac{X^{n+1}}{n+1} + C$$

**Clasa GUI** contine implementarea interfetei grafice – butoanele de operatii, textField - urile de input si rezultat, fereastra care apare pe ecran(panel) si etichetele puse pe interfata (label).

In aceasta clasa, cele 2 metode importante sunt cele de parseMonomial si parsePolynomial care transforma dintr-un String intr-un obiect de tip Monomial, respectiv Polynomial, folosindu-se de expresii regulate (regex).

## 4. Implementare

In acest capitol voi incerca sa descriu cele mai importante metode din fiecare clasa, intrucat in capitolele anterioare am descris in mare modul de functionare al metodelor.

### Clasa Monomial

Cele mai importante metode din aceasta clasa sunt cele de toStringInteger() si toStringDouble(), care functioneaza similar, asadar voi descrie doar una din cele 2.

➔**toStringDouble()**: aceasta metoda verifica mai intai daca coeficientul este zero. Daca este zero, returneaza un String null, semnificand absenta termenului. In caz contrar, se verifica gradul termenului si se formeaza sirul corespunzator:

- Pentru un termen de gradul zero, se returneaza coeficientul formatat la doua zecimale.
- Pentru un termen de gradul unu, se formeaza un sir care contine coeficientul si, daca este cazul, litera "x".
- Pentru termeni cu grade mai mari decat unu, se formeaza un sir care contine coeficientul, litera "x" si exponentul.

➔ **toString()** : verifica ce tip de toString() sa foloseasca

## Clasa Polynomial

Aceasta clasa contine de asemenea un toString() care se foloseste de cel al clasei Monomial, insa cea mai importanta clasa este addMonomial()

- ➔ **addMonomial()** : adauga un monom la un polinom. In cadrul acestei metode, se parcurge fiecare monom din map-ul monomials, care stocheaza monomii polinomului.
- Daca un monom din map are acelasi grad ca si monomul pe care incercam sa-l adaugam, atunci coeficientii celor doua monomi sunt adunati. Acest lucru se face prin preluarea coeficientului monomului din map, adaugandu-i coeficientul monomului pe care dorim sa-l adaugam si actualizand coeficientul monomului din map cu valoarea noua.
  - Daca nu exista niciun monom cu acelasi grad in map, inseamna ca monomul pe care dorim sa-l adaugam este nou si trebuie pur si simplu adaugat in map-ul monomials.

## Clasa Operations

Aceasta clasa contine toate metodele pentru operatii, si voi prezenta o scurta descriere pentru fiecare.

- ➔ **add()** : aduna doua polinoame. Parcurge monomii celui de-al doilea polinom si ii adauga pe fiecare in primul polinom utilizand metoda addMonomial().
- ➔ **sub()** : scade doua polinoame. Parcurge monomii celui de-al doilea polinom si ii adauga cu semnul schimbat (negati) in primul polinom, folosind metoda addMonomial()
- ➔ **mul()** : inmulteste doua polinoame, Parcurge fiecare monom din primul polinom si fiecare monom din al doilea polinom si inmulteste coeficientii si aduna gradele
- ➔ **div()** : imparte polinomul D (deimpartit) la I (impartitor). Impartirea este efectuata folosind metoda clasica de impartire a polinoamelor. Se realizeaza impartirea termen cu termen si se ajusteaza coeficientii si gradele in consecinta. Metoda returneaza un array cu doua polinoame: rezultatul impartirii (C) si restul impartirii (R).
- ➔ **deriv()** : deriveaza un polinom. Parcurge fiecare monom din polinom si calculeaza coeficientul si gradul derivatei pentru fiecare monom

- ➔ **integr()** : integreaza un polinom. Parcurge fiecare monom din polinom si calculeaza coeficientul si gradul integralei pentru fiecare monom

## **Clasa GUI**

Clasa GUI creaza interfata grafica a aplicatiei si gestioneaza interactiunilor utilizatorului. Mai departe voi prezenta metodele acesteia.

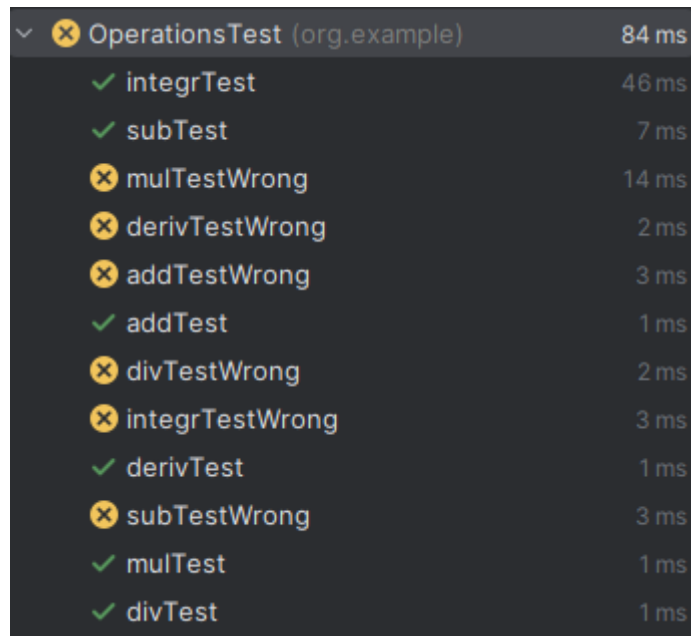
- ➔ **constructorul GUI()**: Initializeaza toate elementele interfetei grafice, cum ar fi ferestre, campuri de text, butoane si etichete.
- ➔ **actionPerformed(ActionEvent e)**: Implementeaza logica pentru actiunile butoanelor. Verifica ce buton a fost apasat si apeleaza metodele corespunzatoare din clasa Operations pentru a efectua operatiile matematice adecvate asupra polinoamelor introduse de utilizator.
- ➔ **parsePolynomial(String input)**: Aceasta metoda primeste un sir de caractere reprezentand un polinom si il analizeaza pentru a extrage monomii. Utilizeaza expresii regulate pentru a gasi monomii si apoi ii adauga intr-un obiect de tipul Polynomial.
- ➔ **parseMonomial(String input)**: Aceasta metoda primeste un sir de caractere reprezentand un monom si il analizeaza pentru a extrage coeficientul si gradul monomului. Utilizeaza expresii regulate pentru a gasi coeficientul si gradul monomului si apoi creeaza un obiect de tipul Monomial cu aceste valori.

## 5. Rezultate

Pentru partea de testare am folosit framework-ul JUnit. Am testat fiecare metoda din clasa Operations pentru a ma asigura ca functionalitatea este cea dorita. Rezultatele au fost cele asteptate, in cazul in care polinoamele introduse au fost corecte. Am reusit sa tratez si cazurile in care pot sa apara erori in introducerea polinoamelor si astfel rezultatele sa nu fie cele dorite.

Mai jos sunt prezentate rezultatele testarii operatiilor folosind JUnit si se poate observa cum testele au avut success, adica rezultatul asteptat corespunde cu cel furnizat de operatie.

Pe langa acestea am efectuat si pentru fiecare operatie un test cu rezultatul gresit, astfel incat am schimbat un semn, sau un coeficient pentru a demonstra functionalitatea. Aceste teste au la finalul numelui "Wrong" pentru deosebire.

A screenshot of a JUnit test runner window showing the results for a class named 'OperationsTest' from the package 'org.example'. The total execution time is 84 ms. The tests are listed with green checkmarks for successful tests and yellow 'x' icons for failed tests. The failed tests are 'mulTestWrong', 'derivTestWrong', 'addTestWrong', 'divTestWrong', and 'integrTestWrong'. The successful tests are 'integrTest', 'subTest', 'addTest', 'derivTest', 'subTestWrong', 'mulTest', and 'divTest'.

|                              |       |
|------------------------------|-------|
| OperationsTest (org.example) | 84 ms |
| ✓ integrTest                 | 46 ms |
| ✓ subTest                    | 7 ms  |
| ✗ mulTestWrong               | 14 ms |
| ✗ derivTestWrong             | 2 ms  |
| ✗ addTestWrong               | 3 ms  |
| ✓ addTest                    | 1 ms  |
| ✗ divTestWrong               | 2 ms  |
| ✗ integrTestWrong            | 3 ms  |
| ✓ derivTest                  | 1 ms  |
| ✗ subTestWrong               | 3 ms  |
| ✓ mulTest                    | 1 ms  |
| ✓ divTest                    | 1 ms  |

## 6. Concluzii

Acest proiect mi-a oferit oportunitatea de a invata sau reaminti multe informatii utile. In primul rand, am inteles procesul de creare a unei interfete grafice, operatiile cu polinoame si implementarea algoritmului de impartire a doua polinoame. In al doilea rand, am invatat sa lucrez cu proiecte Maven, sa utilizez JUnit pentru testare unitara si sa folosesc Regex.

Acest calculator de polinoame poate fi dezvoltat ulterior prin adaugarea mai multor operatii si polinoame, devenind astfel util pentru elevii de liceu sau studentii de la facultatile de Matematica, care se confrunta frecvent cu polinoamele si doresc sa inteleaga operatiile sau sa-si verifice cunostintele.

Desi dezvoltarea acestei aplicatii nu a fost lipsita de provocari si obstacole, am reusit sa le depasesc si sa invat din ele. Pe parcursul proiectului, am realizat ca pentru fiecare problema exista o solutie, iar o analiza atenta a cerintelor si proiectarea corecta sunt esentiale pentru definirea unui model de date precis. Preluarea input-ului utilizatorului a fost, de asemenea, o provocare, care a necesitat o abordare meticuloasa.

Exista mereu loc pentru imbunatatiri, iar aceasta aplicatie poate fi dezvoltata si optimizata. De la design-ul interfetei grafice pana la adaugarea de noi functionalitati, exista posibilitatea de a face operatiile cu polinoame mai interactive si mai usor de utilizat.

## 7. Bibliografie

1. <https://dsrl.eu/courses/pt/>
2. <https://users.utcluj.ro/~igiosan/>
3. <https://www.baeldung.com/java-check-double-integer>
4. <https://www.baeldung.com/java-hashmap-sort>