DOCUMENTATIE

TEMA 1

Pântea Marius Nicușor
GRUPA:30222

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei:

- Implementarea unui calculator de polinoame cu interfață grafică.

Obiective secundare:

- Proiectarea unei arhitecturi orientate pe obiecte. Utilizarea MVC.
- Implementarea operațiilor matematice cu polinoame (adunare, scădere, înmulțire, împărțire, derivare, integrare).
- Dezvoltarea unei interfețe utilizator intuitivă
- Asigurarea testării corectitudinii funcționalităților folosind Junit
- Utilizarea expresiilor regulate si potrivirii de modele pentru extragerea coeficienților polinomului

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

Cerințe funcționale:

- Operații matematice de bază:
- Adunare, scădere, înmulțire, împărțire a două polinoame.
- Derivarea și integrarea unui polinom.

Interfață grafică:

- Design intuitiv al interfeței pentru introducerea polinoamelor și afișarea rezultatelor.
- Posibilitatea de a selecta operația matematică dorită.

Testare:

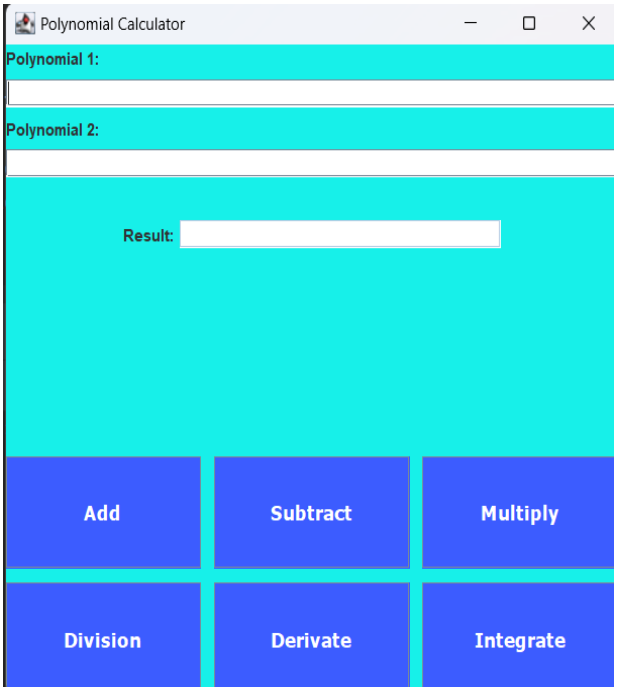
- Realizarea de teste unitare pentru fiecare operație matematică implementată.
- Asigurarea corectitudinii funcționării interfeței grafice.

Cerințe non-funcționale:

- Performanță:
  Răspuns rapid la operațiile matematice, chiar și pentru polinoame complexe.
- Uzabilitate:
  Interfața să fie ușor de utilizat și intuitivă pentru utilizatori de toate nivelurile.
- Fiabilitate:
  Minimizarea erorilor și gestionarea adecvată a excepțiilor.

Manual de utilizare:

- Utilizatorul rulează programul si deschide aplicația de calculator
- Fereastra calculatorului:



- Coeficienți pot fi introduși sub forma cx(c=coeficient) si exponentul x^e
- Utilizatorul apasă unul din butoane , cu nume explicit pentru fiecare operație
- Operația de integrare si derivare se realizează pentru polinomul1
- După ce butonul este apăsat rezultatul va apărea in câmpul lui

3. Proiectare

Diagrama UML de clase utilizata.

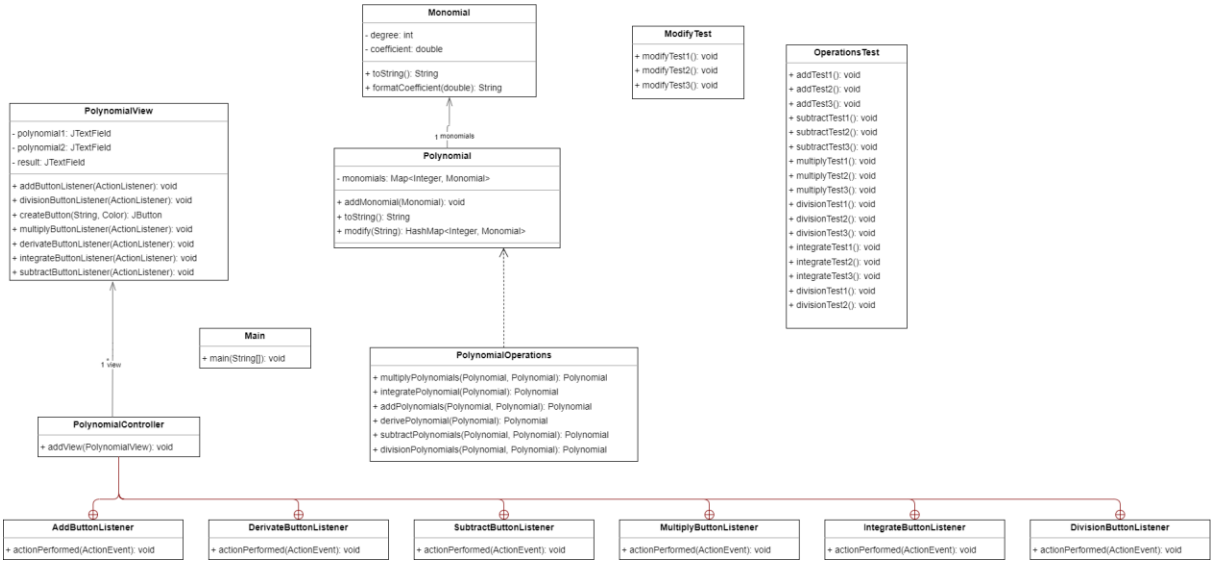
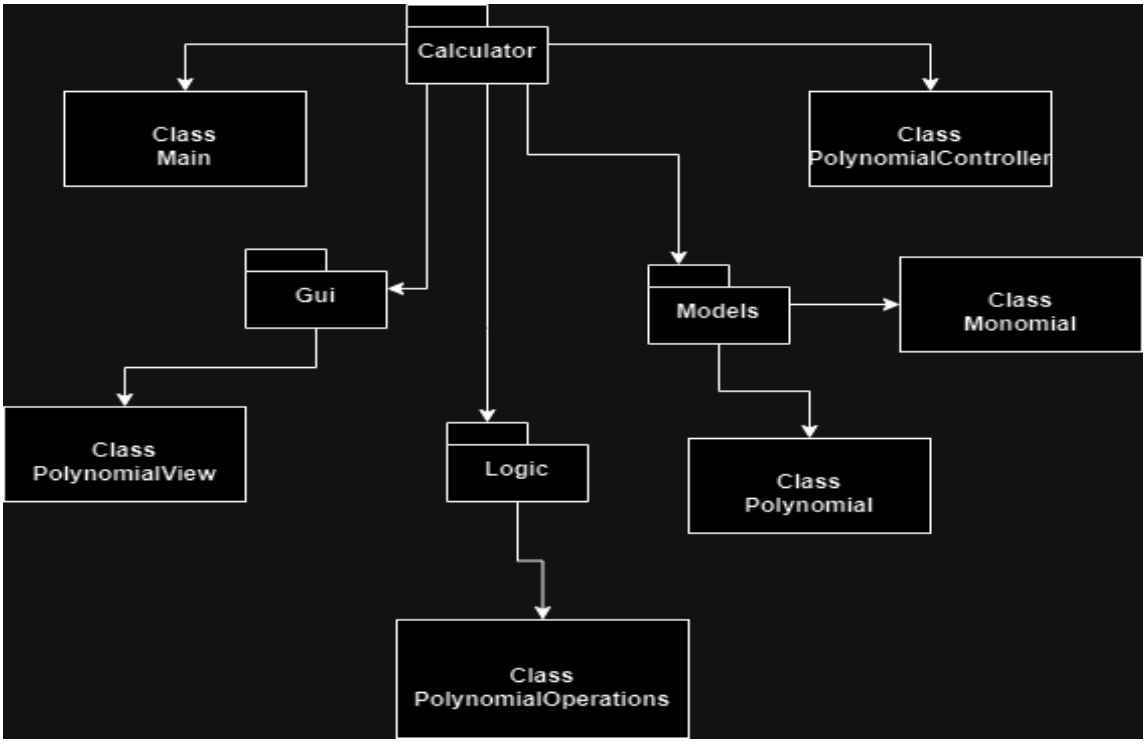


Diagrama de pachete



4. Implementare

Clase utilizate

- Monomial
- Polynomial
- PolynomialOperations
- PolynomialView
- PolynomialController
- Main

Clasa Monomial

Un monom este o expresie matematică care constă dintr-un singur termen, format dintr-un coeficient și o putere a variabilei.

Atributele clasei sunt:
-degree (gradul): reprezintă puterea variabilei în monom.
-coefficient (coeficientul): reprezintă coeficientul numeric asociat variabilei.
Constructorul clasei Monomial primește gradul și coeficientul monomului și le atribuie în variabilele corespunzătoare.

Metodele publice includ:
-getCoefficient(): returnează coeficientul monomului.
-setCoefficient(double coefficient): setează coeficientul monomului.
-getDegree(): returnează gradul monomului.
-toString(): returnează o reprezentare sub formă de șir de caractere a monomului, folosind notația matematică standard (de exemplu, "2x^3").
-formatCoefficient(double coefficient): o metodă auxiliară folosită pentru formatarea coeficientului într-un șir de caractere, eliminând zecimalele zero redundante.

Clasa Polynomial

Un polinom este format din suma sau diferența a mai multor monoame.

Atributele clasei sunt:
-monomials: Un map care stochează monoamele polinomului, cu gradul ca cheie și obiectul Monomial ca valoare.
Constructorul clasei Polynomial primește un map de monoame și îl atribuie la atributul monomials.

Metodele publice includ:
-getMonomials(): Returnează map-ul de monoame.
-addMonomial(Monomial m): Adaugă un monom la polinom. Dacă un monom cu același grad există deja în polinom, coeficientul său este adunat la coeficientul monomului existent.
-toString(): Returnează o reprezentare sub formă de șir de caractere a polinomului, ordonată în ordine descrescătoare a gradului și formată din suma monoamelor.
-modify(String polinomSursa): O metodă statică care primește un șir de caractere reprezentând un polinom și îl transformă într-un map de monoame, folosind expresii regulate pentru analiza termenilor polinomului.

Clasa PolynomialOperations

Această clasă PolynomialOperations conține diverse operații matematice care pot fi efectuate pe polinoame, precum adunare, scădere, înmulțire, împărțire, derivare și integrare.

Metodele publice includ:
-addPolynomials(Polynomial polynomial1, Polynomial polynomial2): Adună două polinoame.
-subtractPolynomials(Polynomial polynomial1, Polynomial polynomial2): Scade un polinom din altul.
-multiplyPolynomials(Polynomial polynomial1, Polynomial polynomial2): Înmulțește două polinoame.
-divisionPolynomials(Polynomial numerator, Polynomial denominator): Efectuează împărțirea unui polinom la altul.
-derivePolynomial(Polynomial polynomial): Calculează derivata unui polinom.
-integratePolynomial(Polynomial polynomial): Calculează integrala unui polinom.
Aceste operații sunt implementate folosind map-uri de monoame pentru a reprezenta polinoamele și pentru a stoca rezultatele intermediare. De asemenea, operațiile sunt gestionate în

mod corespunzător pentru a trata cazuri speciale, cum ar fi divizarea la zero sau eliminarea monoamelor cu coeficientul zero

Clasa PolynomialView

Această clasă PolynomialView este o interfață grafică simplă pentru un calculator de polinoame. Folosește biblioteca Swing pentru a crea interfața utilizatorului.

Componentele interfeței includ:
Două câmpuri de text pentru introducerea polinoamelor (polynomial1 și polynomial2).
Un câmp de text nedeteminat pentru afișarea rezultatului (result).
Șase butoane pentru efectuarea diferitelor operații cu polinoame: adunare, scădere, înmulțire, împărțire, derivare și integrare.
Metode pentru a obține polinoamele introduse și pentru a seta rezultatul afișat.
Metode pentru adăugarea de ascultători de evenimente la butoane pentru a detecta acțiunile utilizatorului.
Această clasă utilizează layout-uri și culori pentru a organiza și aranja componentele în fereastra aplicației, astfel încât să fie ușor de utilizat și să ofere o experiență plăcută utilizatorului.

Clasa PolynomialController

Clasa PolynomialController este un controler care gestionează interacțiunea între interfața grafică a calculatorului de polinoame (PolynomialView) și logica de calcul a polinoamelor (PolynomialOperations). Acesta include ascultători pentru evenimentele generate de acțiunile utilizatorului pe interfața grafică.

Metodele clasei includ:
-addView(PolynomialView view): Metodă pentru atașarea unei instanțe de PolynomialView la acest controler și pentru adăugarea de ascultători de evenimente la butoanele din interfață.
Clasele interne pentru ascultători de evenimente (AddButtonListener, SubtractButtonListener, etc.) implementează interfața ActionListener și definesc comportamentul pentru fiecare acțiune posibilă pe care utilizatorul o poate executa în interfață.
În fiecare ascultător de buton, polinoamele introduse de utilizator sunt parsate și transformate în obiecte de tip Polynomial folosind metoda modify din clasa Polynomial. Apoi, operația corespunzătoare este aplicată pe aceste polinoame folosind metodele din PolynomialOperations. Rezultatul operației este afișat în câmpul de text al interfeței grafice folosind metoda setResult.
Acest controler face legătura între interfața grafică și logica de calcul, asigurându-se că acțiunile utilizatorului sunt corect procesate și că rezultatele sunt afișate corespunzător în interfața grafică.

Main

Acesta este fișierul Main care servește drept punct de intrare în aplicația calculatorului de polinoame. Aici, se creează o instanță a PolynomialView, apoi se atașează acea instanță la un PolynomialController. Apoi, fereastra PolynomialView este făcută vizibilă.

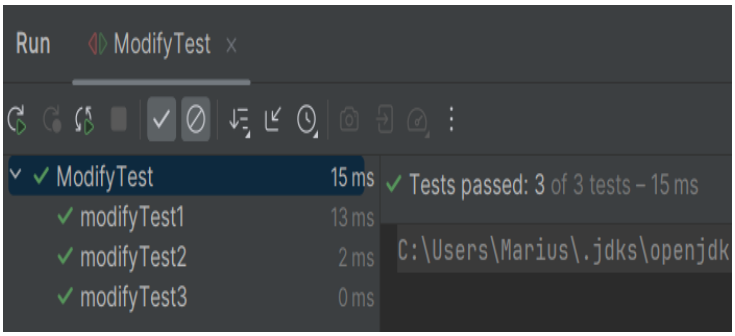
5. Rezultate

Pentru testare s-a utilizat Junit.Fiecare operatie beneficiază de cel putin 2 teste .S-au creat doua clase de test , OperationsTest(pentru a testa operatiile) , si ModifyTest(clasa unde se testeaza cele funcția modify,functie care transformă un sir de polinoame întruna map de monoame.

```
Marius1608
@Test
public void modifyTest1() {
    String polynomialString = "3x^2-2x^3+5x^4-6x+2";
    HashMap<Integer, Monomial> result = Polynomial.modify(polynomialString);

    assertEquals( expected: 3, result.get(2).getCoefficient());
    assertEquals( expected: -2, result.get(3).getCoefficient());
    assertEquals( expected: 5, result.get(4).getCoefficient());
    assertEquals( expected: -6, result.get(1).getCoefficient());
    assertEquals( expected: 2, result.get(0).getCoefficient());
}
```

```
@Test
public void modifyTest2() {
    String polynomialString = "-3x^2-2x^3+5x^4-6x-2";
    HashMap<Integer, Monomial> result = Polynomial.modify(polynomialString);

    assertEquals( expected: -3, result.get(2).getCoefficient());
    assertEquals( expected: -2, result.get(3).getCoefficient());
    assertEquals( expected: 5, result.get(4).getCoefficient());
    assertEquals( expected: -6, result.get(1).getCoefficient());
    assertEquals( expected: -2, result.get(0).getCoefficient());
}
```

```
@Test
public void modifyTest3() {
    String polynomialString = "5";
    HashMap<Integer, Monomial> result = Polynomial.modify(polynomial

    assertEquals( expected: 5, result.get(0).getCoefficient());
}
```



```
Marius1608
@Test
public void addTest1() {

    Polynomial polynomial1 = new Polynomial(new HashMap<>());
    polynomial1.addMonomial(new Monomial( degree: 2,  coefficient: 3));
    polynomial1.addMonomial(new Monomial( degree: 1,  coefficient: 2));

    Polynomial polynomial2 = new Polynomial(new HashMap<>());
    polynomial2.addMonomial(new Monomial( degree: 2,  coefficient: 1));
    polynomial2.addMonomial(new Monomial( degree: 0,  coefficient: 5));

    Polynomial result = PolynomialOperations.addPolynomials(polynomial1, polynomial2);
    assertEquals( expected: "4x^2+2x+5", result.toString());
}
```

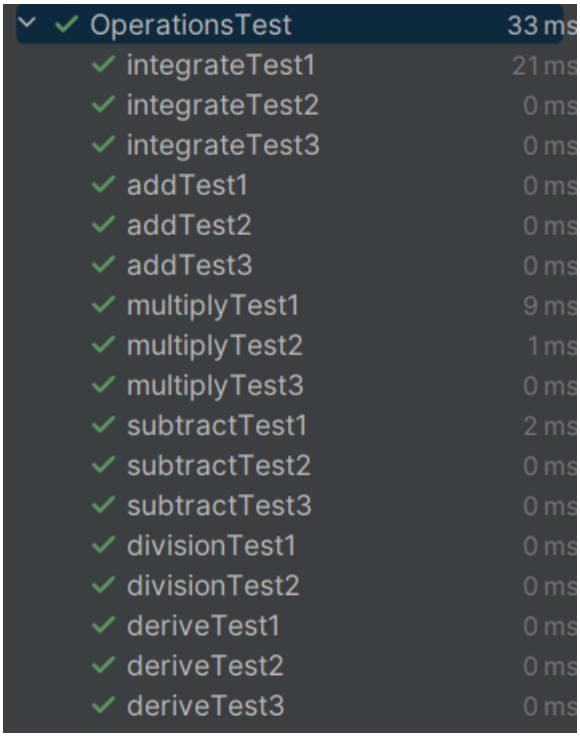
```
Marius1608
@Test
public void multiplyTest1() {

    Polynomial polynomial1 = new Polynomial(new HashMap<>());
    polynomial1.addMonomial(new Monomial( degree: 2,  coefficient: 3));
    polynomial1.addMonomial(new Monomial( degree: 1,  coefficient: 2));

    Polynomial polynomial2 = new Polynomial(new HashMap<>());
    polynomial2.addMonomial(new Monomial( degree: 1,  coefficient: 4));
    polynomial2.addMonomial(new Monomial( degree: 0,  coefficient: 1));

    Polynomial result = PolynomialOperations.multiplyPolynomials(polynomial1, polynomial2);
    assertEquals( expected: "12x^3+11x^2+2x", result.toString());
}
```

```
Marius1608
@Test
public void subtractTest1() {

    Polynomial polynomial1 = new Polynomial(new HashMap<>());
    polynomial1.addMonomial(new Monomial( degree: 2,  coefficient: 3));
    polynomial1.addMonomial(new Monomial( degree: 1,  coefficient: 2));

    Polynomial polynomial2 = new Polynomial(new HashMap<>());
    polynomial2.addMonomial(new Monomial( degree: 2,  coefficient: 1));
    polynomial2.addMonomial(new Monomial( degree: 0,  coefficient: 5));

    Polynomial result = PolynomialOperations.subtractPolynomials(polynomial1, polynomial2);
    assertEquals( expected: "2x^2+2x-5", result.toString());
}
```

```java
Marius1608
@Test
public void integrateTest1() {

    Polynomial polynomial = new Polynomial(new HashMap<>());
    polynomial.addMonomial(new Monomial( degree: 2,  coefficient: 6));
    polynomial.addMonomial(new Monomial( degree: 1,  coefficient: 4));
    polynomial.addMonomial(new Monomial( degree: 0,  coefficient: 2));

    Polynomial result = PolynomialOperations.integratePolynomial(polynomial);

    assertEquals( expected: "2x^3+2x^2+2x", result.toString());
}
```

```java
Marius1608
@Test
public void deriveTest1() {

    Polynomial polynomial = new Polynomial(new HashMap<>());

    polynomial.addMonomial(new Monomial( degree: 3,  coefficient: 6));
    polynomial.addMonomial(new Monomial( degree: 2,  coefficient: 4));
    polynomial.addMonomial(new Monomial( degree: 1,  coefficient: 2));

    Polynomial result = PolynomialOperations.derivePolynomial(polynomial);
    assertEquals( expected: "18x^2+8x+2", result.toString());
}
```

```java
Marius1608
@Test
public void divisionTest1() {
    Polynomial numerator = new Polynomial(new HashMap<>());
    numerator.addMonomial(new Monomial( degree: 2,  coefficient: 6));
    numerator.addMonomial(new Monomial( degree: 1,  coefficient: 4));
    numerator.addMonomial(new Monomial( degree: 0,  coefficient: 2));

    Polynomial denominator = new Polynomial(new HashMap<>());
    denominator.addMonomial(new Monomial( degree: 1,  coefficient: 2));


    Polynomial result = PolynomialOperations.divisionPolynomials(numerator, denominator);
    assertEquals( expected: "3x+2", result.toString());
}
```



Testele rulează cu succes. In proiect sunt mai multe teste efectuate dar cu structura asemănătoare cu cea prezentata.

6. Concluzii

Dezvoltarea unui calculator de polinoame este o experiență valoroasă, pentru o mai profundă înțelegere a conceptelor matematice fundamentale asociate cu polinoamele. Aplicând paradigma Model-View-Controller (MVC), s-a reușit separarea logica componentele aplicației noastre, facilitând astfel gestionarea și extensibilitatea proiectului. Implementarea interfeței grafice în Java cu ajutorul bibliotecii Swing a permis crearea unei interfețe intuitive și ușor de utilizat, îmbunătățind experiența de utilizare În ansamblu, proiectul a oferit nu doar posibilitatea dezvoltării abilități practice în programare, ci și o mai profundă apreciere a matematicii și a procesului de dezvoltare a software-ului.

7. Bibliografie

https://www.baeldung.com/junit-5
https://www.javatpoint.com/java-swing
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S2.pdf
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S3.pdf
https://stackoverflow.com/