DOCUMENTATIE

TEMA *1*

*Calculator Polinomial*

NUME STUDENT: Macavei Alina - Maria

GRUPA: 30224

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Obiectiv principal:

Implementarea unui calculator polinomial capabil să efectueze operații aritmetice și algebrice de bază pe polinoame.

Obiective secundare:

* Analiza problemei și definirea cerințelor funcționale și non-funcționale.
* Modelarea structurii calculatorului polinomial.
* Proiectarea arhitecturii orientate pe obiecte a aplicației.
* Implementarea funcțiilor și interfețelor necesare pentru operațiile cu polinoame.
* Testarea și validarea corectitudinii funcționalităților**.**

1. **Analiza problemei, modelare, scenarii, cazuri de utilizare**
   1. **Cadrul de cerințe**

Proiectul Calculatorului Polinomial este destinat să ofere utilizatorilor o platformă simplă și interactiva pentru a efectua operații matematice pe polinoame. Principalele cerințe funcționale ale aplicației sunt:

1. Utilizatorul trebuie să poată introduce două polinoame sub formă de șiruri de caractere.
2. Utilizatorul trebuie să poată selecta operația matematică pe care dorește să o efectueze asupra polinoamelor.
3. Aplicația trebuie să ofere rezultatul corect al operației selectate.
   1. **Cazuri de utilizare**

Cazul de utilizare: Adunarea / Scaderea / Inmultirea / Impartirea / Derivarea / Integrarea polinoamelor

1. Utilizatorul introduce două polinoame.
2. Utilizatorul selectează operația de adunare.
3. Aplicația afișează rezultatul adunării (etc) polinoamelor.

# 3.Proiectare

Proiectarea acestui proiect implică definirea arhitecturii și a relațiilor dintre diferitele componente ale aplicației. Mai jos sunt detaliate aspectele relevante ale proiectului:

**3**

**3.1 Arhitectura OOP**

Proiectul este structurat în mai multe pachete, fiecare conținând clase și funcționalități relevante:

* **DataModels**: Acest pachet conține clase care se ocupă de extragerea și manipularea datelor introduse sub formă de șiruri de caractere.
* **BusinessLogic**: Aici sunt definite clasele care modelează conceptele matematice precum monomul și polinomul, și oferă operații de bază pentru acestea.
* **GUI**: Pachetul GUI conține clasele care definesc interfața grafică a aplicației.
* **Operations**: Acest pachet include clase care implementează operații matematice pe polinoame.

**3.2 Diagramă UML de Clase**

Diagrama UML de clase arată structura și relațiile dintre clasele din proiect. Mai jos este o reprezentare simplificată a acestei diagrame:

A screenshot of a computer

Description automatically generated

**4**

**3.3 Interfețe și Algoritmi**

* **Interfețe**: Nu sunt definite interfețe specifice în cadrul proiectului meu .
* **Algoritmi**: Codul conține implementări ale algoritmilor necesari pentru operațiile matematice pe polinoame, cum ar fi adunarea, scăderea, înmulțirea, derivarea si integrarea .

**4.Implementare**

Mai jos este o descriere detaliată a claselor și a implementării interfeței utilizator:

**Clasele și Metodele importante:**

1. **Monomial:**
   * **Câmpuri:**
     + **coefficient** (double): Coeficientul monomului.
     + **power** (int): Puterea monomului.
     + **monom** (String): Reprezentarea sub formă de șir a monomului.
   * **Metode Importante:**
     + **Monomial(String monom)**: Constructor care primește un șir de caractere reprezentând monomul și inițializează coeficientul și puterea.
     + **setCoefficients()**: Metodă privată care extrage coeficientul și puterea din șirul monomului.
     + **getCoefficient()**: Returnează coeficientul monomului.
     + **getPower()**: Returnează puterea monomului.
     + **toString()**: Returnează o reprezentare sub formă de șir a monomului.
2. **Polynomial:**
   * **Câmpuri:**
     + **polynom** (HashMap<Integer, Double>): Reprezentarea polinomului sub formă de perechi de forma (putere, coeficient).

5

* + - **monomialList** (List<Monomial>): Lista de monomi care compun polinomul.
  + **Metode Importante:**
    - **Polynomial(List<Monomial> monomialList)**: Constructor care primește o listă de monomi și inițializează polinomul.
    - **monomialToPolynomial()**: Metodă care convertește lista de monomi în reprezentarea polinomială folosind un **HashMap**.
    - **hashToArray(HashMap<Integer, Double> hashMap)**: Metodă care convertește **HashMap**-ul de perechi de putere-coeficient într-un vector de monomi sortat.
    - **toString(HashMap<Integer, Double> polynom)**: Returnează o reprezentare sub formă de șir a polinomului dat.

**Implementarea Interfeței Utilizator:**

Interfața utilizator a fost implementată folosind Java Swing pentru a crea o fereastră grafică interactivă. Interfața include următoarele elemente:

* Două câmpuri de text pentru introducerea polinoamelor.
* Un câmp de text pentru afișarea rezultatelor operațiilor.
* Buton pentru fiecare operație matematică: adunare, scădere, înmulțire, împărțire, calculul restului la împărțire, derivare și integrare.
* Buton pentru ștergerea conținutului câmpurilor de text.
* Clasa **ActionHandler** gestionează evenimentele de la butoane și implementează logica de procesare a operațiilor pe polinoame.

Această implementare permite utilizatorului să introducă două polinoame, să selecteze operația dorită și să obțină rezultatul în fereastra de afișare.

6

# 5.Rezultate

# În această implementare, am adăugat o clasă de test numită OperationsTest, care conține metode de testare pentru fiecare operație matematică din clasa Operations. Aceste metode de

# testare utilizează aserțiile JUnit assertEquals pentru a verifica dacă rezultatele obținute sunt în conformitate cu rezultatele așteptate.

# 6.Concluzii

Calculatorul Polinomial reprezintă o aplicație utilă și eficientă pentru manipularea și efectuarea de operații matematice pe polinoame.

Pe parcursul dezvoltării acestui proiect, am întâmpinat și am rezolvat diverse provocări și aspecte complexe, cum ar fi:

* Implementarea corectă a algoritmilor pentru adunare, scădere, înmulțire, derivare, integrare, împărțire și calculul restului la împărțirea a două polinoame.
* Procesarea corespunzătoare a intrărilor utilizatorului pentru a asigura manipularea corectă a polinoamelor.
* Testarea riguroasă a funcționalității aplicației pentru a asigura corectitudinea și eficiența operațiilor matematice.

Prin implementarea acestui proiect, am acumulat o înțelegere mai profundă a conceptelor matematice și a programării orientate pe obiecte. În concluzie, proiectul Calculatorului Polinomial reprezintă o experiență valoroasă care a contribuit la îmbunătățirea abilităților de programare și înțelegerea matematică.

**Posibile dezvoltări ulterioare**

În viitor, proiectul Calculatorului Polinomial ar putea fi îmbunătățit prin adăugarea următoarelor funcționalități:

* Interfață grafică mai avansată, cu opțiuni suplimentare pentru vizualizare și manipulare
* Suport pentru polinoame cu coeficienți complecși sau de ordin superior.
* Implementarea unui sistem de salvare și încărcare a polinoamelor și a rezultatelor operațiilor.
* Validarea datelor de intrare

# 7.Bibliografie

1. Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.
2. What are Java classes? - [www.tutorialspoint.com](http://www.tutorialspoint.com)

7