**DOCUMENTATIE**

**TEMA 1**

**Nume prenume: Hruban Andrada-Bianca**

**Grupa: 30226**

Cuprins

[1. Obiectivul temei 3](#_Toc130905600)

[1.1. Obiectiv Principal: 3](#_Toc130905601)

[1.2. Obiective Secundare: 3](#_Toc130905602)

[2.Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc130905603)

[Diagramă USE CASE 5](#_Toc130905604)

[3.Proiectare 5](#_Toc130905605)

[3.1 Structuri de date 5](#_Toc130905606)

[3.1. Diagrama de clase 6](#_Toc130905607)

[3.2 Algoritmi 6](#_Toc130905608)

[4.Implementare 7](#_Toc130905609)

[Clasa Polynomial 7](#_Toc130905610)

[Clasa Operation 8](#_Toc130905611)

[Clasa Pair 8](#_Toc130905612)

[Clasa GUI si App 9](#_Toc130905613)

[Clasa VerificareOperatii 9](#_Toc130905614)

[5.Rezultate 10](#_Toc130905615)

[6.Concluzii si Dezvoltări Ulterioare 14](#_Toc130905616)

[7.Bibliografie 15](#_Toc130905617)

# 1.Obiectivul temei

## Obiectiv Principal:

Proiectul vizează implementarea unui sistem de procesare a polinoamelor univariabile cu coeficienți întregi. Acesta facilitează efectuarea operațiilor precum adunarea, scăderea, înmulțirea și împărțirea între două polinoame, cât și derivarea sau integrarea unui polinom singular. Sistemul se distinge prin datele de intrare (unul sau două polinoame) și datele de ieșire (rezultatele operațiilor). Implementarea proiectului asigură funcționalitatea și accesibilitatea pentru o gamă largă de utilizatori.

## Obiective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii | Prezentarea funcționalității prin prezentarea problemei și a unor diagrame Use Case | 3 |
| Crearea diagramei UML | Prezentarea claselor, metodelor si interfețelor | 3 |
| Alegerea structurilor de date | Structurile de date folosite și motivul folosirii acestora. | 4 |
| Implementarea de algoritmi | Folosiți pentru operții | 4 |
| Împărțirea pe clase | Prezentarea fiecărei clase a proiectului. | 4 |
| Alegere design interfață și implemenatrea | Implementare posibilă în java swing sau FX | 4 |
| Testarea unitară | Verificarea operațiilor implementate | 5 |

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Polinoamele sunt o parte importantă a matematicii și se bazează pe conceptul de expresie algebraică construită dintr-una sau mai multe variabile și constante, folosind doar operații de adunare, scădere, înmulțire și ridicare la putere constantă pozitivă întreagă. În implementarea clasei Polynomial, un polinom este reprezentat ca o colecție de termeni, fiecare termen fiind format dintr-un coeficient și un exponent al variabilei. Această reprezentare permite utilizatorului să efectueze operații matematice uzuale, cum ar fi adunarea, scăderea, înmulțirea, derivarea și integrarea polinoamelor

În general, polinoamele sunt folosite pentru a modela fenomenele din viața reală, cum ar fi creșterea plantelor, viteza unei mașini sau fluctuațiile piețelor financiare. În programare, clasele cum ar fi Polynomial permit programatorilor să efectueze operațiuni matematice complexe pe polinoame cu ușurință și precizie.

P(x) = an \* xn + an-1 \* xn-1+ ... + a1\* x + a0,

unde ai sunt coeficientii iar n este gradul polinomului.

Această reprezentare constituie forma canonică a polinoamelor cu o singură variabilă.

**2.1 Cadrul de cerinte functionale:**

* Programul trebuie să permită preluarea unui polinom sub forma unui șir de caractere, folosind regex, care să fie stocat ca un obiect de tip Polynomial.
* Programul trebuie să ofere metoda de afișare a unui polinom sub forma unui șir de caractere, cu termenii ordonați descrescător după puterea termenului.
* Programul trebuie să ofere metoda de returnare a gradului polinomului.
* Programul trebuie să ofere metoda de returnare a coeficientului principal al polinomului.
* Utilizatorul poate introduce două polinoame în câmpurile corespunzătoare de pe interfața grafică
* Utilizatorul poate selecta una dintre următoarele operații: adunare, scădere, înmulțire, împărțire, derivare sau integrare
* Sistemul va afișa rezultatul operației selectate într-un câmp de afișare

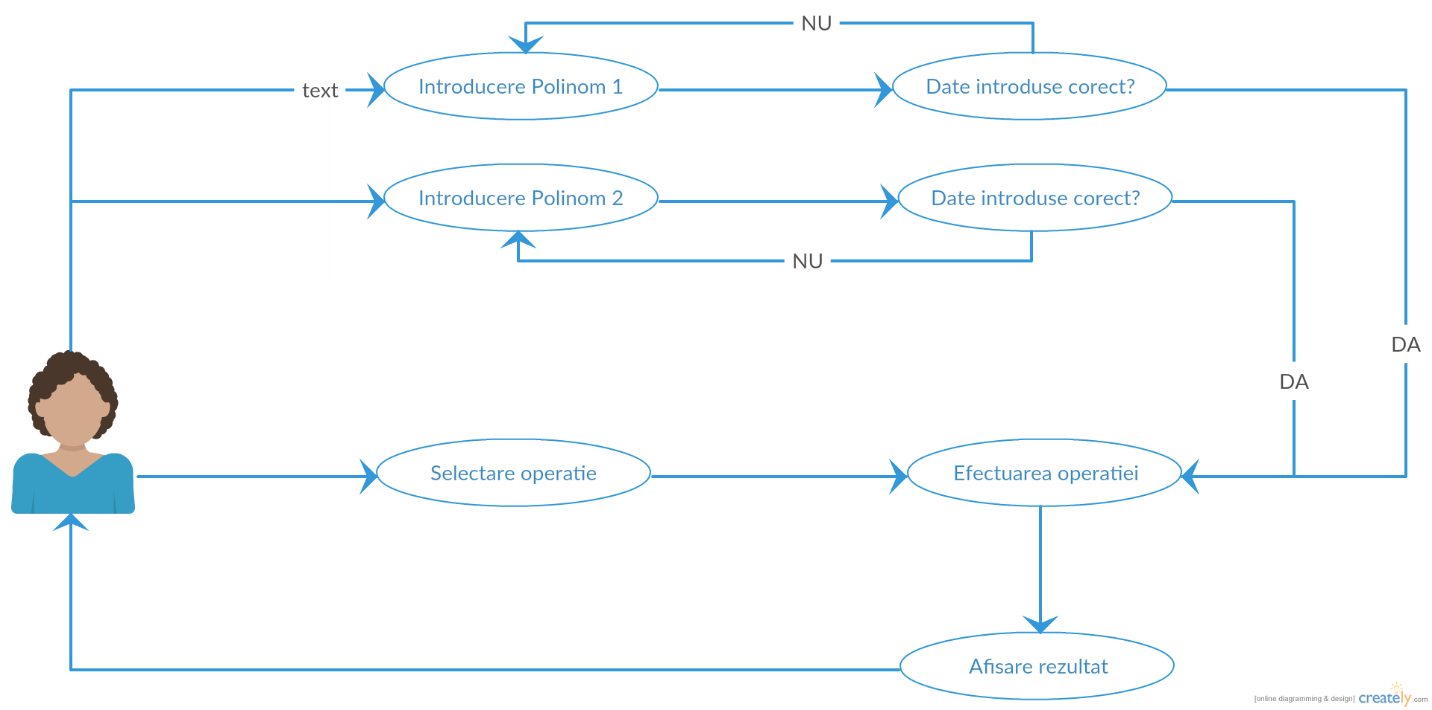
**2.2 Cadrul de cerinte non-funcționale:**

* Programul trebuie să fie scris în limbajul de programare Java.
* Programul trebuie să aibă o eficiență sporită, în special în ceea ce privește timpul de rulare.
* Programul trebuie să aibă o interfață ușor de utilizat și intuitivă.
* Programul trebuie să fie testat corespunzător și să prezinte un grad ridicat de fiabilitate și stabilitate.
* Interfața grafică trebuie să fie intuitivă și ușor de folosit
* Sistemul trebuie să fie eficient și să ofere un timp de răspuns rapid
* Sistemul trebuie să fie robust și să nu permită introducerea de date eronate de către utilizator

**2.3 Use-case-ul :**

1. Utilizatorul deschide aplicația.
2. Interfața grafică se încarcă și se afișează în fereastra.
3. Utilizatorul introduce polinoamele în câmpurile corespunzătoare.
4. Utilizatorul selectează operația dorită: Adunare, Scădere, Înmulțire, Împărțire, Derivare sau Integrare.
5. Aplicația verifică dacă datele introduse sunt valide și afișează un mesaj de eroare dacă nu sunt.
6. Dacă datele introduse sunt valide, aplicația realizează operația selectată asupra celor două polinoame.
7. Rezultatul operației este afișat în câmpul corespunzător.
8. Utilizatorul poate apăsa butonul "Reset" pentru a șterge datele introduse și rezultatul afișat.
9. Utilizatorul poate închide aplicația.

### Diagramă USE CASE



# Proiectare

## 3.1 Structuri de date

În domeniul informaticii, o structură de date reprezintă o abordare sistematizată de organizare și stocare a informațiilor într-un calculator, astfel încât acestea să poată fi utilizate eficient. Adesea, selecționarea adecvată a unei structuri de date va facilita implementarea unui algoritm eficient.

Proiectarea OOP a aplicației este bazată pe utilizarea claselor pentru a reprezenta obiecte de tip polinom. Clasa Polynomial reprezintă un polinom și include o hartă pentru a stoca coeficienții și puterile termenilor. Clasa Operation include metode pentru a efectua operații aritmetice pe polinoame, precum adunare, scădere, înmulțire, împărțire, derivare și integrare.

Interfața grafică a utilizatorului (GUI) este implementată prin intermediul clasei GUI. Aceasta include câmpuri de text pentru a introduce polinoame, butoane pentru a selecta operația aritmetică și o etichetă pentru a afișa rezultatul. GUI folosește clasele Polynomial și Operation pentru a efectua operațiile aritmetice și a afișa rezultatele.

Pentru a stoca polinoamele și rezultatele intermediare, s-a folosit o hartă (Map<Integer, Double>) în care cheile reprezintă puterile termenilor și valorile reprezintă coeficienții lor.

Map: Map este o interfață din Java care face parte din pachetul java.util. Ea reprezintă o colecție de perechi cheie-valoare, unde fiecare cheie este unică. Map permite stocarea, accesarea, modificarea și ștergerea valorilor în funcție de cheile asociate.

TreeMap: TreeMap este o clasă ce implementează interfața Map și se regăsește în același pachet java.util. TreeMap stochează perechile cheie-valoare într-o structură de arbore binară de căutare echilibrată, ceea ce permite operații eficiente de adăugare, ștergere și căutare. Un avantaj al TreeMap față de alte implementări ale interfeței Map este că păstrează ordinea naturală a cheilor sau o ordine definită de un comparator furnizat. Acest lucru face ca TreeMap să fie potrivit atunci când este necesară ordonarea datelor pe baza cheilor.

Map-ul este implementat sub forma unei TreeMap, pentru a păstra termenii polinomului ordonați în ordine descrescătoare după putere.

HashMap este o colecție neordonată și neclasificată, adică nu există o ordine specifică în care elementele sunt stocate. În schimb, elementele sunt stocate folosind un algoritm de dispersie bazat pe cheia fiecărui element. Aceasta permite o căutare eficientă a unui element după cheia sa.

In clasa Operation, pentru adunare, scadere si inmultire, se utilizeaza harti (Map) pentru a stoca termenii polinoamelor rezultate. In metoda divide se utilizeaza o pereche (Pair) de polinoame pentru a returna atat catul, cat si restul impartirii.

## Diagrama de clase

Graphical user interface

Description automatically generated

## Algoritmi

Fiecare operație în parte presupune implementarea unui algoritm special pentru realizarea ei.

În clasa Operation, pentru adunare, scădere și înmulțire, se parcurg map-urile termenilor fiecărui polinom și se adaugă, scad sau înmulțesc termenii cu aceeași putere, astfel încât să se obțină polinomul rezultat. Pentru împărțire, se folosește algoritmul de împărțire a două polinoame, prin scăderea repetată a unui multiplu al împărțitorului din dividend, până când gradul restului devine mai mic decât gradul împărțitorului. Pentru derivare, se aplică formula derivatelor de polinoame. Pentru integrare, se aplică formula de integrare a polinoamelor.

Acestea sunt aspectele principale ale proiectării OOP a aplicației Polynomial Calculator, care utilizează clase, interfețe, structuri de date și algoritmi pentru a realiza operații cu polinoame...

# Implementare

În realizarea problemei am decis ca am nevoie de 5 clase: Polynomial, Operation, Pair, GUI, VerificareOperatii, App.

### Clasa Polynomial

Această clasă Java, numită **Polynomial**, reprezintă un polinom folosind un **Map<Integer, Double>**. Cheia map-ului reprezintă exponentul (puterea), iar valoarea asociată este coeficientul termenului. Clasa conține metode pentru a obține termenii polinomului, pentru a prelua un polinom dintr-un șir de caractere folosind regex, pentru a converti un polinom într-un șir de caractere și pentru a obține gradul și coeficientul termenului principal al polinomului.

Metodele din clasa **Polynomial** sunt:

1. **Polynomial(Map<Integer, Double> terms)**: Constructorul care primește un map de termeni ca argument și inițializează polinomul cu aceștia.
2. **Polynomial()**: Constructorul fără argumente care inițializează polinomul ca unul gol, adică un polinom cu coeficienții egali cu zero.
3. **getTerms()**: Această metodă returnează termenii polinomului ca un map.
4. **preiaRegex(String s)**: Această metodă extrage termenii polinomului din șirul de caractere **s** folosind regex și adaugă termenii în map.
5. **toString()**: Această metodă converteste polinomul într-un șir de caractere în formatul corespunzător (ex. "2x^3 - 5x^2 + 3x - 7").
6. **getDegree()**: Această metodă returnează gradul polinomului (adică exponentul cel mai mare).
7. **getLeadingCoefficient()**: Această metodă returnează coeficientul termenului principal (termenul cu cel mai mare exponent) al polinomului.

### Clasa Operation

Această clasă Java, numită **Operation**, conține metode statice pentru a efectua operații matematice pe polinoame, cum ar fi adunarea, scăderea, înmulțirea, împărțirea, derivarea și integrarea. Clasa folosește clasa **Polynomial** pentru a reprezenta polinoamele.

Metodele din clasa **Operation** sunt:

1. **add(Polynomial p1, Polynomial p2)**: Această metodă adună două polinoame, **p1** și **p2**, și returnează un nou polinom rezultat.
2. **subtract(Polynomial p1, Polynomial p2)**: Această metodă scade polinomul **p2** din polinomul **p1** și returnează un nou polinom rezultat.
3. **multiply(Polynomial p1, Polynomial p2)**: Această metodă înmulțește două polinoame, **p1** și **p2**, și returnează un nou polinom rezultat.
4. **divide(Polynomial dividend, Polynomial divisor)**: Această metodă împarte polinomul **dividend** la polinomul **divisor** și returnează un obiect **Pair<Polynomial, Polynomial>** care conține câtul și restul împărțirii.
5. **derivative(Polynomial p)**: Această metodă calculează derivata polinomului **p** și returnează un nou polinom rezultat.
6. **integrate(Polynomial p)**: Această metodă calculează integrala polinomului **p** și returnează un nou polinom rezultat. Rețineți că această metodă calculează integrala indefinită, deci nu se adaugă o constantă arbitrară la rezultat.

Metodele folosesc obiecte **Map** pentru a reprezenta termenii polinoamelor și efectuează operațiile matematice pe acestea.

### Clasa Pair

Această clasă Java, numită **Pair**, este o clasă de perechi generice, care conține două obiecte de tipuri diferite sau de același tip. Clasa **Pair** este utilă atunci când doriți să returnați două obiecte dintr-o metodă sau să stocați două obiecte împreună.

Metodele din clasa **Pair** sunt:

1. **Pair(A first, B second)**: Constructorul care primește două obiecte, **first** de tipul A și **second** de tipul B, și le inițializează în instanța clasei Pair.
2. **getFirst()**: Această metodă returnează primul obiect din pereche de tipul A.
3. **getSecond()**: Această metodă returnează al doilea obiect din pereche de tipul B.
4. **toString()**: Această metodă converteste perechea într-un șir de caractere într-un format specific pentru operația de împărțire a polinoamelor: "Quotient [first] Remainder [second]", unde [first] și [second] sunt reprezentări în șir de caractere ale obiectelor stocate în pereche.

Clasa **Pair** poate fi folosită în diverse scenarii, cum ar fi stocarea unui rezultat și a unui rest în cazul împărțirii polinoamelor, cum este utilizat în clasa **Operation**.

### Clasa GUI si App

Această clasă Java, numită **GUI**, creează o interfață grafică pentru un calculator de polinoame. Acesta conține câmpuri text pentru introducerea a două polinoame, etichete pentru afișarea informațiilor și butoane pentru efectuarea diferitelor operații matematice. Rezultatul operației selectate va fi afișat în interfața grafică.

Metodele din clasa **GUI** sunt:

1. **initialize()**: Această metodă inițializează interfața grafică și componente, precum și adaugă acțiuni pentru butoane.
2. **performOperation(Operation2 operation)**: Această metodă primește o operație de tip **Operation2** și execută operația matematică corespunzătoare pentru polinoamele introduse în câmpurile text. Apoi, afișează rezultatul în eticheta **resultDisplayLabel**.
3. **resetFields()**: Această metodă golește câmpurile text și resetează eticheta **resultDisplayLabel** la mesajul inițial.

Clasa **GUI** conține, de asemenea, o enumerare privată **Operation2** care conține tipurile de operații disponibile în aplicație: **ADD**, **SUBTRACT**, **MULTIPLY**, **DIVIDE**, **DERIVE** și **INTEGRATE**.

Clasa **App** are o metodă **main.** Această metodă creează o instanță a clasei **GUI** și setează vizibilitatea ferestrei la **true**. Acest cod va porni aplicația și va afișa fereastra calculatorului de polinoame.

### Clasa VerificareOperatii

Cel mai folosit utilitar pentru testarea unitară a claselor Java este JUnit.

În clasa **VerificareOperatii**, au fost create teste pentru a verifica funcționalitatea metodelor de operații polinomiale din clasa **Operation**. Următoarele teste au fost realizate în clasa:

1. **setUp**: Această metodă este anotată cu **@Before**, ceea ce înseamnă că va fi executată înainte de fiecare test. În această metodă, au fost inițializate două polinoame, **p1** și **p2**, care vor fi folosite în toate metodele de testare.
2. **adunareTest**: Acest test verifică funcționalitatea metodei **add**. Polinoamele **p1** și **p2** au fost adunate, iar rezultatul a fost comparat cu valoarea așteptată sub formă de șir de caractere.
3. **scadereTest**: Acest test verifică funcționalitatea metodei **subtract**. Polinoamele **p1** și **p2** au fost scăzute, iar rezultatul a fost comparat cu valoarea așteptată sub formă de șir de caractere.
4. **inmultireTest**: Acest test verifică funcționalitatea metodei **multiply**. Polinoamele **p1** și **p2** au fost înmulțite, iar rezultatul a fost comparat cu valoarea așteptată sub formă de șir de caractere.
5. **impartireTest**: Acest test verifică funcționalitatea metodei **divide**. Polinoamele **p1** și **p2** au fost împărțite, iar rezultatul a fost comparat cu valoarea așteptată sub formă de șir de caractere.
6. **derivareTest**: Acest test verifică funcționalitatea metodei **derivative**. Derivata polinomului **p1** a fost calculată, iar rezultatul a fost comparat cu valoarea așteptată sub formă de șir de caractere.
7. **integrareTest**: Acest test verifică funcționalitatea metodei **integrate**. Integrala polinomului **p1** a fost calculată, iar rezultatul a fost comparat cu valoarea așteptată sub formă de șir de caractere.

Prin efectuarea acestor teste, s-a asigurat că metodele de operații polinomiale funcționează corect și oferă rezultatele așteptate.

# 5.Rezultate

Pentru interfata grafica GUI putem verifica cu cate un calcul operatiile matematice.

Verificare adunare:

Table

Description automatically generated

Verificare scadere:

Table

Description automatically generated

Verificare înmulțire:

Table

Description automatically generated

Verificare împărțire:

Table

Description automatically generated

Verificare împărțire cu zero:

Graphical user interface, application

Description automatically generated

Verificare derivare:

Table

Description automatically generated

Verificared integrare:

Table

Description automatically generated

La apăsarea butonului de reset se revine la conținutul inițial al interfeței și se pot efectua operațiile în continuare.

Pentru testarea cu JUnit, se introduc pentru fiecare test de metodă din clasa Operații polinoamele cu care se vor efectua operațiile sub formă de String-uri. Se efectuează operația și apoi se compară rezultatul obținut cu cel așteptat prin verificarea rezultatului metodei *assertEquals* aplicată asupra celor două polinoame.

Pentru a rula aceste teste, asigură-te că ai următoarele lucruri:

1. Ai biblioteca JUnit în proiectul tău. Dacă folosești Maven sau Gradle, adaugă dependența JUnit în fișierul de configurare (**pom.xml** sau **build.gradle**). Dacă folosești un IDE, cum ar fi IntelliJ IDEA sau Eclipse, poți adăuga biblioteca JUnit folosind opțiunea de a adăuga o bibliotecă externă.
2. Toate clasele de test și clasele testate trebuie să fie în aceeași ierarhie de pachete sau să aibă importuri corespunzătoare.
3. Asigură-te că ai anotările **@Before** și **@Test** importate din pachetul **org.junit**.

După ce ai configurat totul, poți rula testele în IDE-ul tău sau folosind un instrument de construire, cum ar fi Maven sau Gradle. Rezultatele testelor vor fi afișate în consolă sau într-un panou de testare dedicat în IDE. Dacă toate testele trec, înseamnă că metodele de operații polinomiale funcționează corect. Dacă un test eșuează, vei vedea o descriere a erorii, care te va ajuta să identifici și să corectezi problema.

Graphical user interface, text, application

Description automatically generated

# 6.Concluzii si Dezvoltări Ulterioare

Consider că abordarea utilizată în crearea acestei aplicații pentru efectuarea operațiilor asupra polinoamelor a fost simplă și logică. Interfața grafică dezvoltată facilitează accesul utilizatorilor, indiferent dacă sunt specialiști sau persoane cu cunoștințe limitate în domeniu, la specificațiile proiectului într-un mod simplu și ușor.

Dezvoltarea acestui proiect a contribuit la îmbunătățirea competențelor mele de programare în Java și la aprofundarea noțiunilor și conceptelor din Programarea Orientată pe Obiect. Am dobândit noi tehnici pentru crearea interfețelor și am acordat o mai mare importanță diagramelor UML și abordării de a descompune probleme complexe în subprobleme mai mici. În plus, am învățat despre expresiile regulate (regex) și cum să le folosesc, precum și despre Map-uri și modul lor de implementare.

Acest proiect prezintă oportunități de dezvoltare ulterioară, cum ar fi:

• îmbunătățirea eficienței algoritmilor de implementare a operațiilor și a timpului de execuție

• optimizarea organizării codului sursă

• adăugarea de noi operații asupra polinoamelor, precum evaluarea polinomului într-un punct specific sau calculul rădăcinilor

• explorarea altor metode pentru efectuarea operațiilor

• prezentarea rezultatelor în alte forme

• dezvoltarea unei interfețe mai atractive

• generarea de grafice pentru a reprezenta funcțiile într-un sistem de coordonate.

# Bibliografie

* https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression
* <http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>
* https://www.guru99.com/java-swing-gui.html
* laboratoarele si cursurile de Programare Orientata Obiect
* https://www.section.io/engineering-education/how-to-build-a-java-gui-calculator-from-scratch-using-box-layout/