DOCUMENTATIE

TEMA 1

NUME STUDENT: Pop Tudor Ștefan

GRUPA: 30226

# CUPRINS

[DOCUMENTATIE 1](#_Toc1758258530)

[TEMA 1 1](#_Toc726120905)

[CUPRINS 1](#_Toc1327418216)

[1. Obiectivul temei 2](#_Toc1710619296)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 2](#_Toc1033766731)

[4. Implementare 5](#_Toc1347409158)

[5. Rezultate 7](#_Toc158899123)

[6. Concluzii 7](#_Toc1110925315)

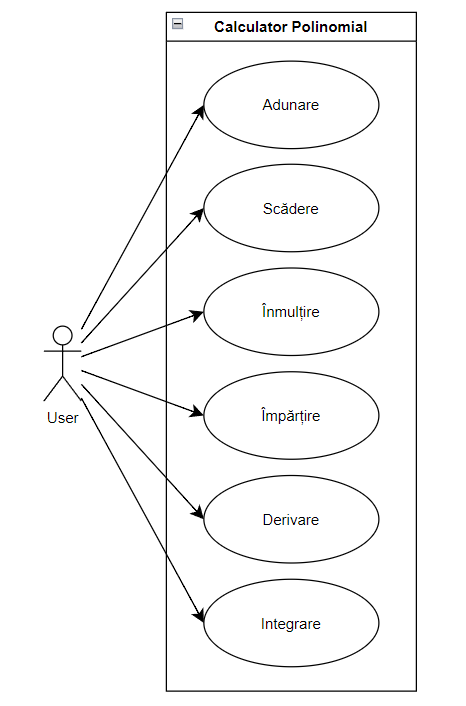
# Obiectivul temei

Obiectivul principal al acestei teme este implementarea unui calculator polinomial. În alte cuvinte, dorim să proiectăm și să implementăm un sistem capabil de adunare de polinoame, scădere de polinoame, înmulțire de polinoame, împărțire de polinoame, de a determina derivata unui polinom și integrala unui polinom.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Din punct de vedere funcțional, cerințele pe care trebuie să le satisfacă sistemul nostru sunt cele de a putea aduna, scădea, înmulți, și împărți două polinoame conform definițiilor acestor operații în sens matematic, precum și a putea deriva și integra un polinom, conform definiției.

Din punct de vedere nonfuncțional, cerințele pe care trebuie să le satisfacă sistemul nostru sunt strict legate de eficiența programului, ușurința în utilizare, și crearea unei interfețe grafice primitoare pentru utilizator, cu scopul de a încapsula detaliile tehnice din spatele calculatorului nostru.

Sistemul nostru prezintă un total de șase cazuri de utilizare, conform următoarei diagrame:

Vom prezenta acum, ca și considerente teoretice, fiecare dintre cazurile de utilizare ale sistemului nostru:

1. **Adunarea Polinoamelor:** În acest caz de utilizare, actorul principal este utilizatorul sistemului. Secvența principală în caz de scenariu de succes este următoarea:
2. Utilizatorul introduce polinoamele în primele două căsuțe text din interfața grafică utilizator.
3. Utilizatorul selectează operația de adunare prin apăsarea butonului “Add”.
4. Calculatorul polinomial execută în fundal operația de adunare de polinoame.
5. După calcul, sistemul afișează rezultatul adunării în a treia căsuță text a interfeței.
6. **Scăderea Polinoamelor:** În acest caz de utilizare, actorul principal este utilizatorul sistemului. Secvența principală în caz de scenariu de succes este următoarea:
7. Utilizatorul introduce polinoamele în primele două căsuțe text din interfața grafică utilizator.
8. Utilizatorul selectează operația de scădere prin apăsarea butonului “Subtract”.
9. Calculatorul polinomial execută în fundal operația de scădere de polinoame.
10. După calcul, sistemul afișează rezultatul scăderii în a treia căsuță text a interfeței.
11. **Înmulțirea Polinoamelor:** În acest caz de utilizare, actorul principal este utilizatorul sistemului. Secvența principală în caz de scenariu de succes este următoarea:
12. Utilizatorul introduce polinoamele în primele două căsuțe text din interfața grafică utilizator.
13. Utilizatorul selectează operația de înmulțire prin apăsarea butonului “Multiply”.
14. Calculatorul polinomial execută în fundal operația de înmulțire de polinoame.
15. După calcul, sistemul afișează rezultatul înmulțirii în a treia căsuță text a interfeței.
16. **Împărțirea Polinoamelor (NEFUNCȚIONALĂ):** În acest caz de utilizare, actorul principal este utilizatorul sistemului. Secvența principală în caz de scenariu de succes este următoarea:
17. Utilizatorul introduce polinoamele în primele două căsuțe text din interfața grafică utilizator.
18. Utilizatorul selectează operația de împărțire prin apăsarea butonului “Divide”.
19. Calculatorul polinomial execută în fundal operația de împărțire de polinoame.
20. După calcul, sistemul afișează rezultatul împărțirii ca și pereche de polinoame (cât, rest) în penultima, respectiv, ultima căsuță text a interfeței.
21. **Derivarea unui Polinom:** În acest caz de utilizare, actorul principal este utilizatorul sistemului. Secvența principală în caz de scenariu de succes este următoarea:
22. Utilizatorul introduce polinomul ce va fi derivat în prima căsuță text din interfața grafică utilizator. (Al doilea polinom va fi ignorat în contextul derivării.)
23. Utilizatorul selectează operația de derivare prin apăsarea butonului “Differentiate”.
24. Calculatorul polinomial execută în fundal operația de derivare a unui polinom.
25. După calcul, sistemul afișează rezultatul derivării în cea de-a treia căsuță text a interfeței.
26. **Integrarea unui Polinom:** În acest caz de utilizare, actorul principal este utilizatorul sistemului. Secvența principală în caz de scenariu de succes este următoarea:
27. Utilizatorul introduce polinomul ce va fi integrat în prima căsuță text din interfața grafică utilizator. (Al doilea polinom va fi ignorat în contextul integrării.)
28. Utilizatorul selectează operația de integrare prin apăsarea butonului “Integrate”.
29. Calculatorul polinomial execută în fundal operația de integrare a unui polinom.
30. După calcul, sistemul afișează rezultatul integrării în cea de-a treia căsuță text a interfeței.

În toate cele șase cazuri de utilizare, există două cazuri de eșec. Primul apare dacă unul dintre cele două șiruri de caractere introduse în căsuțele text nu sunt valide. Spre exemplu, "dfskhf",

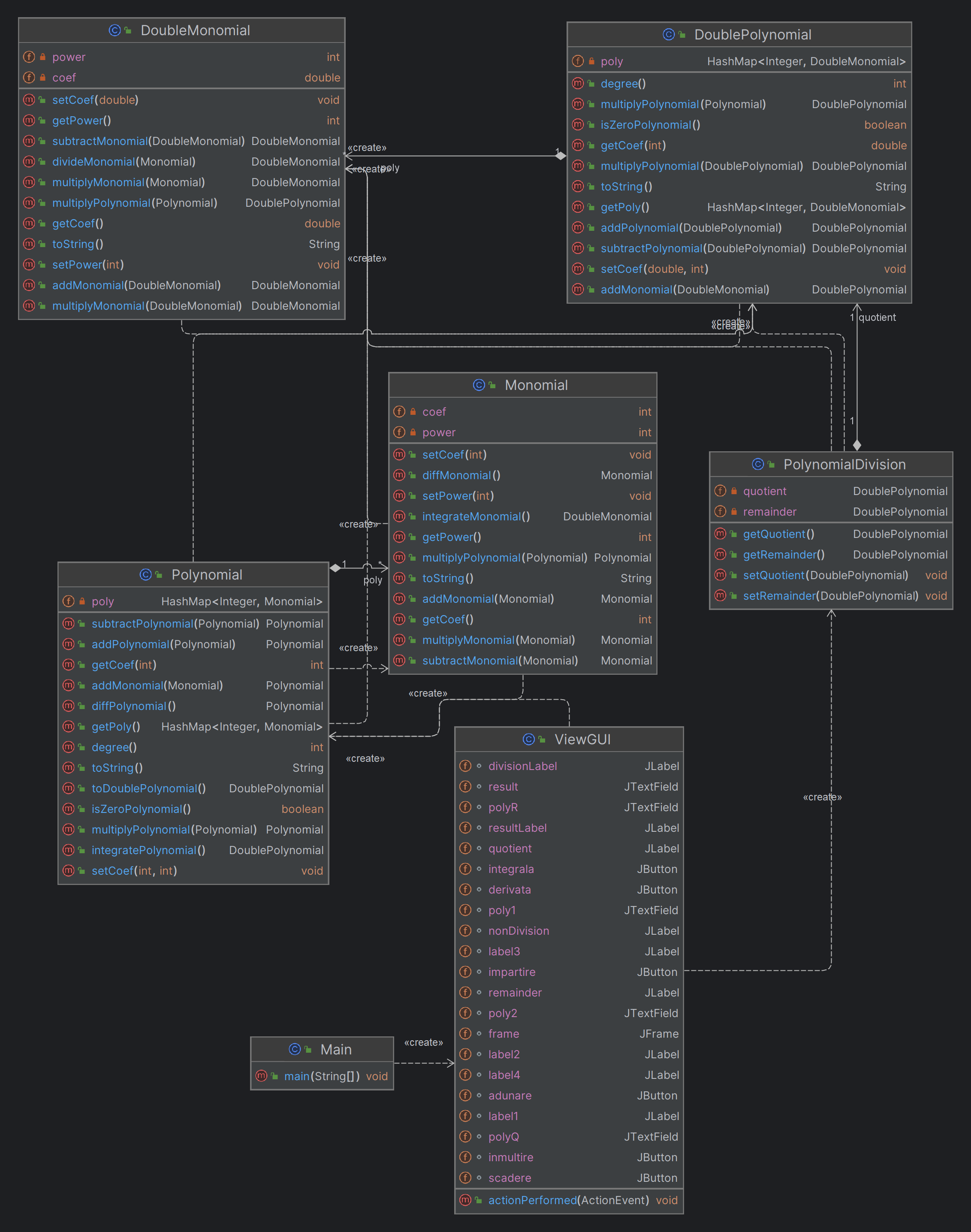
"-", și "5y^2" NU sunt considerate polinoame valide. Singurele șiruri care sunt considerate polinoame valide sunt cele unde toți termenii au forma "ax^n", "x^n", "-x^n", "ax", "x", "-x", sau "a", unde a este număr întreg, și n este număr natural, și toți termenii sunt separați între ei cu "+" sau "-".

Al doilea caz de eșec este dacă punem spații între monoamele componente și separatorii "+" și "-"-". Mai concret, eșecul apare în cazul din pricina separatorului "-". Spre exemplu, dacă scriem "4x^3 - 5x^2" în una dintre căsuțe, "-5x^2" va fi tratat ca și "5x^2", și calculele vor fi eronate.

1. **Proiectare**

Toate clasele necesare pentru implementarea acestui sistem vor fi implementate în cadrul aceluiași pachet, deoarece, numărul total de clase este unul redus, așadar, nu suntem nevoiți să separăm componentele de date în mai multe pachete.

Clasele pe care urmează să le implementăm sunt ilustrate în următoarea diagramă UML de clase:



# Implementare

1. **Clasa "ViewGUI.java":** În această clasă este definită interfața grafică utilizator, pentru care vom utiliza pachetul javax.swing. În această interfață grafică au fost utilizate căsuțe text de tip JTextField, etichete de tip JLabel, și butoane de tip JButton. Se va apela explicit comanda "frame.setLayout(null);" pentru a exprima dorința de a nu avea niciun layout implicit al paginii, ci de a ne defini noi unul singur de la 0. Fiecărui buton îi asociem câte un ascultător, care, în caz de scenariu de succes, va traduce intrările introduse de la tastatură din șiruri de caractere în polinoame reprezentate în formatul specificat de către clasa "Polynomial.java".
2. **Clasa "Monomial.java":** O clasă auxiliară, unde definim un monom, pe care îl vom utiliza ulterior pentru a descrie una dintre părțile componente ale polinomului din "Polymomial.java". Metodele importante din această clasă sunt următoarele:
3. "public Monomial(String s);": Unul dintre cei 3 constructori diferiți ai acestei clase. Acesta este cel mai important, deoarece, acesta va fi responsabil pentru trecerea din șir de caractere în reprezentare internă. Procedează conform formei discutate la punctul 2.
4. "public Monomial addMonomial(Monomial M);": Această metodă adună coeficienții a două monoame dacă și numai dacă acestea au aceeași putere. În caz contrar, adunarea ar rezulta într-un binom.
5. "public Monomial multiplyMonomial(Monoimal M);": Această metodă înmulțește coeficienții și adună puterile a două monoame diferite. În acest caz, orice produs de două monoame este tot monom (având în vedere că variabila independentă este aceeași).
6. "public Monomial subtractMonomial(Monomial M);": Analog cu addMonomial, această metodă scade coeficienții în caz de exponent egal în cele două monoame.
7. "public DoubleMonomial divideMonomial(Monomial M);": Împarte două monoame, dacă și numai dacă exponentul numărătorului este minim egal cu cel al numitorului. "DoubleMonomial.java" este o clasă ce va stoca monoame cu coeficienți reali.
8. "public Monomial diffMonomial();": Calculează derivata unui monom, conform formulei matematice.
9. "public DoubleMonomial integrateMonomial();": Calculează integrala unui monom, și, în caz că împărțirea nu este exactă, necesităm un coeficient real.
10. "public Polynomial multiplyPolynomial(Polynomial P);": Înmulțește un monom cu un polinom.
11. "public String toString();": Transformă un monom din reprezentare internă în reprezentare sub formă de șir de caractere, conform formei dorite discutate anterior la punctul 2.
12. **Clasa "DoubleMonomial.java":** O altă clasă auxiliară pentru operațiile de integrare și împărțire, unde definim un monom cu coeficienți reali pentru a descria părțile componente ale polinomului "DoublePolynomial.java". Metodele importante sunt, întâi, toate cele discutate anterior, doar că de tip DoubleMonomial, și cu parametru intern tot de tip DoubleMonomial, însă, de data aceasta, se mai definește încă o metodă în conjuncție cu celelalte, respectiv, "public DoubleMonomial multiplyMonomial(Monomial M);". Această metodă înmulțește un monom cu coeficient REAL împreună cu un monom cu coeficient ÎNTREG. Acest lucru va fi esențial mai târziu.
13. **Clasa "Polynomial.java":** Cea mai importantă clasă. Metodele acestei clase vor fi responsabile pentru toate parsările din șir de caractere în polinom intern, și pentru toate operațiile ce vrem să le definim. Un polinom este definit ca fiind o concatenare de monoame cu puteri distincte. Acest lucru este codificat prin utilizarea unei structuri de tip "HashMap<Integer, Monomial>()" care este o listă de asocieri "Integer => Monomial".
14. "public Polynomial(String input);": Unul dintre cei trei constructori ai acestei clase, și cel mai important dintre toți trei. Acest constructor utilizează expresii regulate pentru a separa pe rând fiecare coeficient de fiecare putere.
15. "public boolean isZeroPolynomial();": O funcție care determină dacă polinomul actual este funcția identic egală cu 0. Acest lucru este necesar pentru implementarea operației de împărțire.
16. "public Polynomial <add | subtract | multiply>Polynomial(Polynomial P);": Aceste trei funcții codifică trei dintre cele șase operații dorite, respectiv, adunarea, scăderea, și înmulțirea polinoamelor.
17. "public Polynomial diffPolynomial();": Determină derivata polinomului actual.
18. "public DoublePolynomial integratePolynomial();": Determină integrala polinomului actual.
19. "public DoublePolynomial toDoublePolynomial();": Produce traducerea directă a unui polinom cu coeficienți întregi într-un polinom cu coeficienți reali.
20. "public int degree();": Returnează gradul polinomului actual.
21. "public String toString();": Parsează polinomul actual în reprezentare de șir de caractere pentru a fi afișată în urma rezultatelor.
22. **Clasa "DoublePolynomial.java":** Echivalentul clasei "Polynomial.java" pentru clasa "DoubleMonomial.java". Clasă definită pentru a lucra cu calculul integralelor, respectiv, al împărțirii. Nu există nicio metodă cu caracter deosebit față de metodele din clasa precedentă.
23. **Clasa "PolynomialDivision.java":** În această clasă vom stoca perechea de rezultate cât și rest (definite ca DoublePolynomials) a două polinoame cu coeficienți întregi. Singura metodă a acestei clase (cu excepția metodelor mutatoare și accesoare) este constructorul său "public PolynomialDivision(Polynomial P, Polynomial Q);" care calculează câtul și restul împărțirii polinomului P la Q, conform algoritmului prezentat în pseudocodul din PT\_2024\_A1\_S1.pdf.
24. **Clasa "Main.java":** Apelăm deschiderea interfeței grafice, din care putem executa operațiile dorite.

# Rezultate

În contextul testării, a fost utilizată biblioteca JUnit 4.12 în clasa "OperationsTest.java" pentru testare unitară.

Pentru adunare, a fost definită funcția "public void additionTest();" cu polinoamele de test

"3x^4 + 4x^3 – x + 2", "3x - 1", și "3x^4 + 4x^3 + 2x + 1".

Pentru scădere, a fost definită funcția "public void subtractionTest();" cu polinoamele de test

"3x^4 + 4x^3 – x + 2", "3x - 1", și "3x^4 + 4x^3 – 4x + 3".

Pentru înmulțire, a fost definită funcția "public void multiplicationTest();" cu polinoamele de test "3x - 1", "-4", și "-12x + 4".

Pentru derivare, a fost definită funcția "public void diffTest();" cu polinomul de test

"x^3 + x^2 + x + 1", și "3x^2 + 2x + 1".

Pentru derivare, a fost definită funcția "public void diffTest();" cu polinomul de test

"x^3 + 3x^2 + x + 1", și "0.25x^4 + x^3 + 0.5x^2 + x".

# Concluzii

În concluzie, din această temă s-a învățat cum să lucrăm cu interfețele grafice utilizator pe cale de utilizare a funcțiilor din pachetul "javax.swing", cum să lucrăm cu expresii regulate și pattern-matching, respectiv, cum să implementăm structuri abstracte compuse precum polinoamele, utilizând implementări ale interfeței "Map<K, V>". Acest program NU este unul perfect, deoarece operația de împărțire nu este funcțională, așadar, aceasta constituie o posibilitate de dezvoltare ulterioară, respectiv, eficiența programului probabil nu a fost maximizată.