DOCUMENTATIE

TEMA *1*

NUME STUDENT: Oprisiu Ionut Daniel
GRUPA: 30223

CUPRINS

1. Obiectivul temei

Obiectiv principal: Dezvoltarea unei aplicații Java complexe, dedicată manipulării polinoamelor, este scopul central al acestei teme. Aplicația va permite utilizatorului să execute o varietate de operații matematice fundamentale - adunare, scădere, înmulțire, împărțire, derivare, și integrare - pe polinoame de diverse grade. Această funcționalitate va fi încapsulată într-o interfață grafică intuitivă, oferind astfel o utilizare accesibilă și eficientă.

Obiective Secundare

- **Analiza și modelarea cerințelor pentru manipularea polinoamelor:**
  Acest obiectiv implică identificarea și definirea clară a nevoilor utilizatorilor finali ai aplicației, precum și stabilirea funcționalităților specifice necesare pentru manipularea eficientă a polinoamelor. Se vor determina cazurile de utilizare principale și se va efectua o analiză detaliată a modului în care utilizatorii interacționează cu sistemul pentru a executa operații matematice asupra polinoamelor.

- **Proiectarea structurilor de date pentru stocarea polinoamelor și a monoamelor:**
  Acest pas presupune conceperea unor structuri de date optimizate pentru reprezentarea internă a polinoamelor și monoamelor, facilitând astfel operațiunile matematice. Se va acorda atenție selecției tipurilor de date și mecanismelor de colecție din Java care oferă echilibrul optim între performanță, flexibilitate și ușurință de utilizare.

- **Implementarea operațiilor matematice specifice asupra polinoamelor:**
  Implementarea corectă și eficientă a operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare reprezintă un obiectiv cheie. Se vor dezvolta algoritmi preciși și se vor testa riguros pentru a asigura acuratețea rezultatelor matematice furnizate de aplicație.

- **Crearea unei interfețe grafice pentru interacțiunea cu utilizatorul:**
  Scopul acestei faze este de a proiecta și implementa o interfață grafică prietenoasă și intuitivă, care să permită utilizatorilor să introducă polinoame, să selecteze operațiile dorite și să vizualizeze rezultatele într-un mod clar și accesibil. Interfața va trebui să fie adaptabilă la diferite platforme și dispozitive, oferind astfel accesibilitate largă.

- **Realizarea de teste pentru validarea corectitudinii operațiilor:**
  Finalmente, acest obiectiv se concentrează pe asigurarea calității aplicației prin testarea riguroasă a tuturor operațiilor și funcționalităților. Se vor implementa teste unitare pentru fiecare componentă a sistemului, urmărindu-se detectarea și corectarea oricăror erori sau comportamente neașteptate. Testarea va contribui la stabilirea fiabilității, preciziei și performanței aplicației, asigurându-se că aceasta îndeplinește toate cerințele și așteptările utilizatorilor.

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

Cerințe Funcționale

Introducerea Polinoamelor: Utilizatorii trebuie să fie capabili să introducă două polinoame în aplicație pentru a efectua operații matematice. Această introducere poate fi realizată prin tastarea directă a polinoamelor într-o interfață grafică utilizator (GUI) prietenoasă sau prin încărcarea acestora dintr-un fișier text. Polinoamele trebuie să poată fi exprimate într-un format standard, de exemplu, 3x^2 + 2x - 5, și aplicația trebuie să ofere validare pentru a asigura corectitudinea sintaxei.

Suport pentru Operații Matematice: Aplicația va suporta următoarele operații matematice fundamentale pe polinoame:

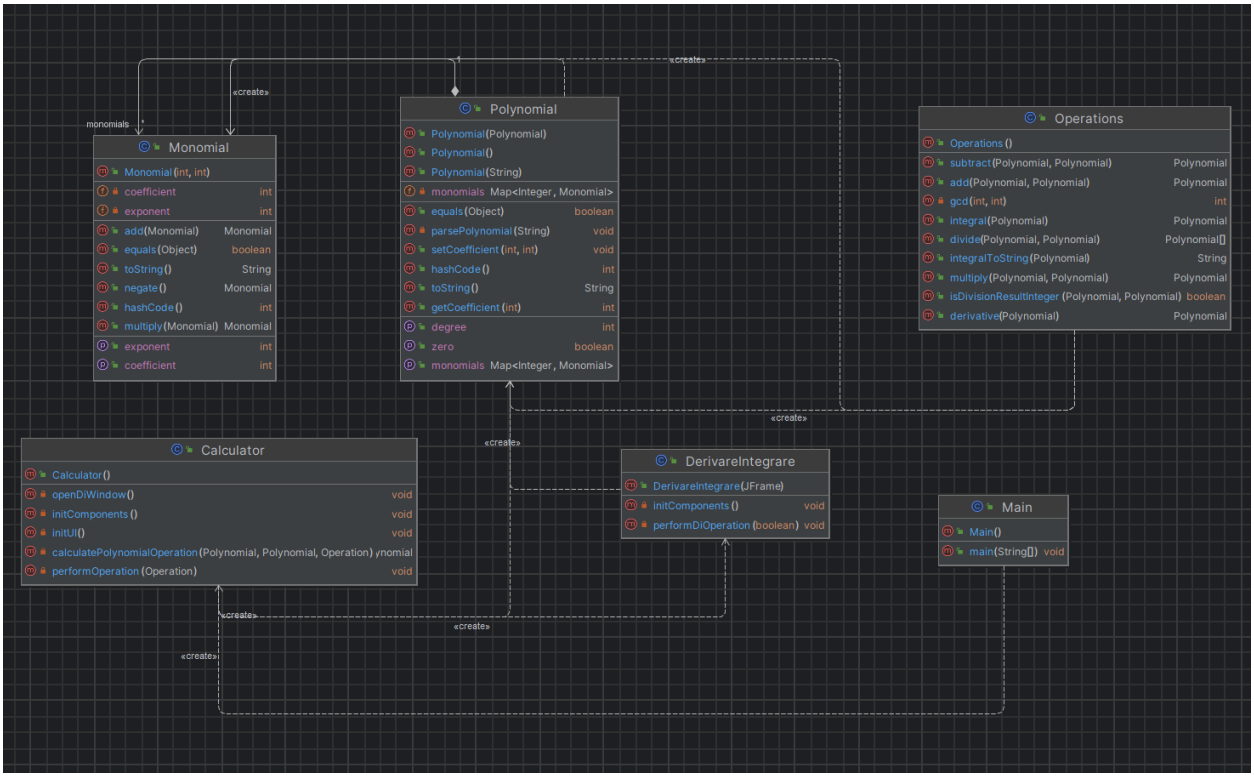
- Adunarea: Calculul sumei a două polinoame, realizând adunarea coeficienților termenilor cu grade similare.
- Scăderea: Determinarea diferenței dintre două polinoame, scăzând coeficienții termenilor cu grade similare.
- Înmulțirea: Producerea unui nou polinom prin înmulțirea fiecărui termen din primul polinom cu fiecare termen din cel de-al doilea.
- Împărțirea: Împărțirea unui polinom la altul, oferind atât câtul cât și restul operației.
- Derivarea: Calculul derivatei unui polinom, reducând exponentul fiecărui termen și înmulțind coeficientul cu exponentul original.
- Integrarea: Integrarea unui polinom, crescând exponentul fiecărui termen cu o unitate și împărțind coeficientul la noul exponent, plus adăugarea constantei de integrare.

3. Proiectare

Arhitectura aplicatiei

Aplicația pentru manipularea polinoamelor este construită folosind principiile programării orientate pe obiecte (OOP), ceea ce facilitează modularitatea, reutilizabilitatea codului și îmbunătățirea claritătii structural.

Diagrama UML:

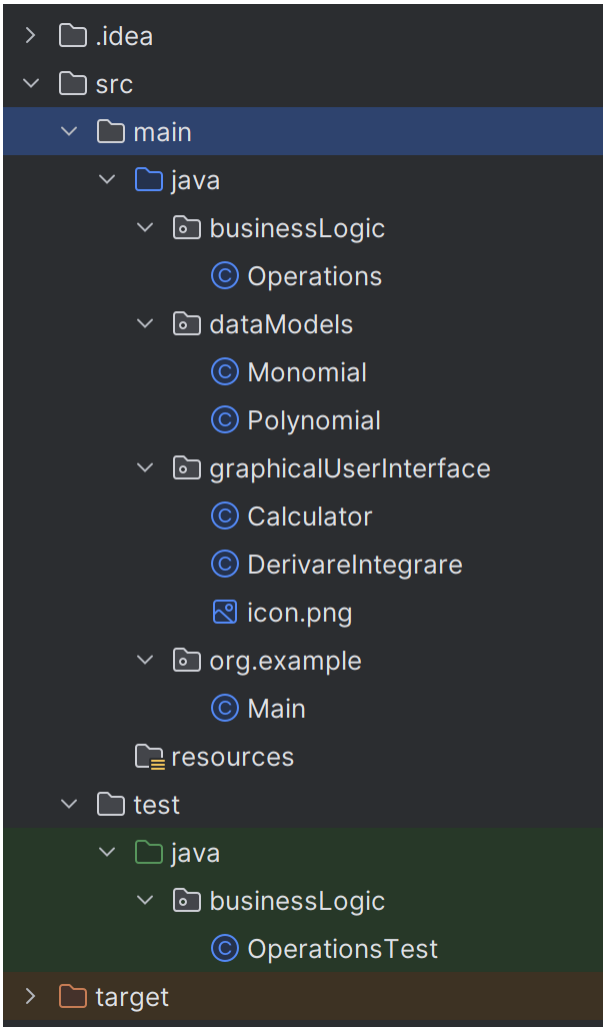


- **Clasa Monomial** reprezintă un monom individual cu coeficient și exponent. Atributurile includ coefficient (int sau float) și exponent (int). Metodele pot include constructori, getteri și setteri, și o metodă de afișare.
- **Clasa Polynomial** conține o listă sau un map de monoame (Monomial). Metodele includ adăugarea, scăderea, înmulțirea, și împărțirea de monoame, precum și calculul derivatei și a integralei.
- **Clasa Operations** este responsabilă pentru implementarea algoritmilor specifici operațiilor între polinoame, cum ar fi adunarea, scăderea, înmulțirea, împărțirea, derivarea, și integrarea. Aceasta utilizeaza metode statice care primesc ca parametri obiecte Polynomial și returnează noi obiecte Polynomial ca rezultat.

Structuri de Date Folosite - Pentru a gestiona colecția de monoame dintr-un polinom, folosim:

- **Listă de Monoame:** Permite stocarea și iterarea ordonată a monoamelor.
- **Mapă cu Exponentul ca Cheie**: Oferă acces rapid la monoame bazat pe exponentul lor, util pentru operații cum ar fi adunarea sau scăderea.

Pachete:



- **dataModels**: Conține clasele Monomial și Polynomial care formează structura de bază a datelor.

- **businessLogic:** Găzduiește clasa Operations și alte clase logice necesare pentru procesarea polinoamelor.
- **graphicalUserInterface**: Include clase pentru implementarea UI, facilitând interacțiunea cu utilizatorul.

4. Implementare

Implementarea aplicației pentru manipularea polinoamelor ilustrează aplicarea principiilor programării orientate-obiect pentru a facilita operații matematice complexe într-un mod structurat și eficient. În continuare, sunt detaliate implementările cheie ale claselor "Monomial", "Polynomial" și "Operations".

Clasa 'Monomial'

Câmpuri:

- **coefficient**: Tipul de date (de exemplu, double sau int) reprezintă coeficientul monomului.
- **exponent**: Tipul de date int, reprezintă exponentul monomului.

Metode importante:

- **Constructor**: Inițializează un nou monom cu un coeficient și un exponent specificat.
- **toString():** Returnează reprezentarea șir de caractere a monomului, pentru afișare.

Clasa 'Polynomial'

Câmpuri:

- **monomials**: O listă sau o mapă care stochează obiecte Monomial, reprezentând componentele polinomului.

Metode importante:

- **Constructor**: Inițializează un nou polinom gol sau cu monoame specifice.
- **addMonomial**(Monomial monomial): Adaugă un monom la polinom.
- **add(Polynomial other):** Implementează adunarea cu un alt polinom și returnează rezultatul.
- **toString():** Returnează reprezentarea șir de caractere a polinomului, pentru afișare.

Clasa ‘Operations’

Metode importante:

- static add(Polynomial p1, Polynomial p2): Această metodă statică primește două polinoame ca argumente și returnează un nou polinom rezultat prin adunarea monoamelor lor. Utilizează un algoritm de parcurgere a listelor sau mapelor de monoame, combinând coeficienții monoamelor cu același exponent.
- static subtract(Polynomial p1, Polynomial p2): Similar cu metoda de adunare, această metodă calculează diferența dintre două polinoame, scăzând coeficienții monoamelor cu exponenți identici din primul polinom cu cei din al doilea polinom.
- static multiply(Polynomial p1, Polynomial p2): Implementează înmulțirea a două polinoame, creând un nou polinom a cărui monoame sunt rezultate prin înmulțirea fiecărui monom din p1 cu fiecare monom din p2 și adunarea rezultatelor cu exponenți identici.
- static divide(Polynomial p1, Polynomial p2): Returnează un array sau o structură care conține atât câtul cât și restul împărțirii polinomului p1 la polinomul p2. Această metodă necesită un algoritm mai complex care să gestioneze scăderea succesivă a produsului dintre divizor și un monom ales astfel încât gradul restului să fie diminuat.
- static differentiate(Polynomial p): Calculează derivata polinomului p, aplicând regulile de derivare pentru fiecare monom. Acest proces implică scăderea exponentului fiecărui monom cu unitatea și înmulțirea coeficientului său cu exponentul original.
- static integrate(Polynomial p): Determină integrala polinomului p, creșterea exponentului fiecărui monom cu o unitate și împărțirea coeficientului său la noul exponent. Această metodă adaugă de asemenea o constantă de integrare, care poate fi specificată de utilizator sau tratată ca o constantă generică.

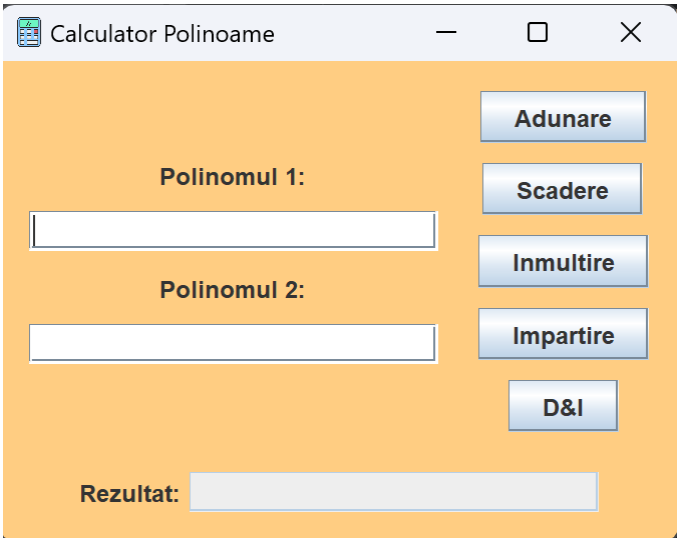
Clasa ‘Calculator’

Clasa ‘**Calculator**’ servește ca interfața principală pentru aplicația de manipulare a polinoamelor. Aceasta oferă utilizatorilor capacitatea de a introduce polinoame, a selecta operații matematice și de a vedea rezultatele operațiilor.

Câmpuri de Introducere: Două câmpuri de text permit introducerea polinoamelor cu care utilizatorul dorește să opereze. Acestea sunt etichetate corespunzător (ex. "Polinomul 1" și "Polinomul 2") pentru a indica unde să fie introduse polinoamele.

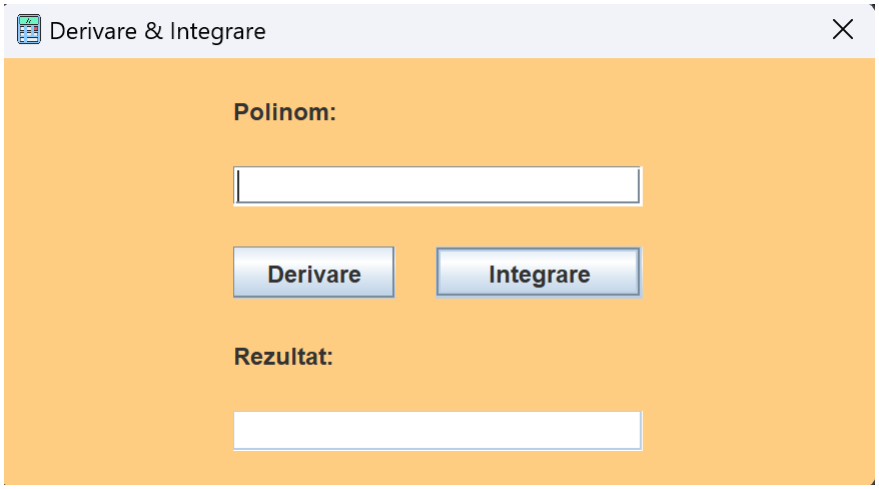
Selector de Operații Permite utilizatorului să selecteze operația dorită (adunare, scădere, înmulțire, împărțire, derivare, integrare).

Zonă de Afișare a Rezultatelor: Rezultatul operațiilor este afișat într-o zonă dedicată sub butoanele de calcul, permițând utilizatorului să vadă imediat outputul operației.



Clasa ‘DerivareIntegrare’

Clasa ‘**DerivareIntegrare’** oferă o interfață dedicată pentru realizarea operațiilor specifice de derivare și integrare a polinoamelor. Aceasta poate fi prezentată ca o fereastră secundară sau un panou în cadrul interfeței ‘**Calculator’**.



5. Rezultate

Scenariile de testare prezentate utilizează JUnit pentru a verifica corectitudinea implementării operațiilor matematice asupra polinoamelor în cadrul aplicației. Aceste teste acoperă funcționalitățile esențiale ale clasei Operations, evaluând operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare.

Rezultatele Testării cu Junit

Test Adunare (add):
Scenariu: Adăugarea a două polinoame, x și 2x, pentru a obține 3x.
Rezultat: Testul trece cu succes, confirmând corectitudinea operației de adunare.

Test Scădere (subtract):
Scenariu: Scăderea polinomului 2x din 3x pentru a obține x.
Rezultat: Testul trece, indicând corectitudinea implementării scăderii.

Test Înmulțire (multiply):
Scenariu: Înmulțirea polinomului x cu 2 pentru a obține 2x.
Rezultat: Testul validează cu succes operația de înmulțire.

Test Împărțire (divide):
Scenariu: Împărțirea 4x^2 la 2x pentru a obține 2x ca și cât și 0 ca și rest.
Rezultat: Testul confirmă corectitudinea împărțirii, inclusiv calculul corect al restului.

Test Derivare (derivative):
Scenariu: Derivarea 3x^2 pentru a obține 6x.
Rezultat: Testul trece, evidențiind corectitudinea calculului derivatei.

Test Integrare (integral):
Scenariu: Integrarea 6x pentru a obține 3x^2 + C ca reprezentare șir de caractere.
Rezultat: Testul validează corectitudinea integrării, inclusiv formatul corect al rezultatului.

Test Adăugare Termeni Multipli (addMultipleTerms):
Scenariu: Adăugarea 2x^3 + x^2 și x^2 + 3x + 4 pentru a obține 2x^3 + 2x^2 + 3x + 4.
Rezultat: Testul trece, demonstrând capacitatea aplicației de a gestiona adăugarea polinoamelor cu termeni multipli.

Test Scădere Termeni Multipli (subtractMultipleTerms):
Scenariu: Scăderea 3x^3 + 2x - 5 din 4x^3 - x^2 + x + 1 pentru a obține x^3 - x^2 - x + 6.
Rezultat: Testul confirmă corectitudinea scăderii termenilor multipli.

Test Înmulțire Termeni Multipli (multiplyMultipleTerms):
Scenariu: Înmulțirea x^2 + 2x + 1 cu x + 1 pentru a obține x^3 + 3x^2 + 3x + 1.
Rezultat: Testul trece, validând algoritmul de înmulțire pentru termeni multipli.

Test Derivare Termeni Multipli (derivativeMultipleTerms):
Scenariu: Derivarea 3x^4 - 5x^3 + 2x^2 + x - 7 pentru a obține 12x^3 - 15x^2 + 4x + 1.
Rezultat: Testul indică corectitudinea derivării polinoamelor cu termeni multipli.

Test Integrare Complexă (integralComplex):
Scenariu: Integrarea `3x^3 - 1/2x^2 + 4x - 5pentru a obține o expresie integrată corectă, inclusiv ajustarea coeficienților fracționari.
Rezultat: Testul confirmă că metoda de integrare gestionează corect termenii multipli și fracționari, oferind rezultatul așteptat"(3/4)x^4 - (1/6)x^3 + 2x^2 - 5x + C"`.

Aceste teste acoperă o gamă largă de scenarii, de la operații simple până la cele mai complexe cazuri care implică termeni multipli și coeficienți fracționari. Rezultatele testelor unitare cu JUnit asigură că implementarea operațiilor pe polinoame este corectă și că aplicația se comportă așa cum este așteptat în diferite condiții.

6. Concluzii

Dezvoltarea aplicației pentru manipularea polinoamelor a fost o experiență valoroasă și educativă, care a adus în prim-plan numeroase aspecte ale programării orientate pe obiect și a procesului de dezvoltare a software-ului. Acest proiect a oferit o oportunitate excelentă de a aplica concepte teoretice într-un context practic, consolidând înțelegerea algoritmilor matematici și a principiilor de design software. Mai jos sunt prezentate principalele concluzii și lecțiile învățate, precum și direcții pentru dezvoltări ulterioare.

Posibile Dezvoltări Ulterioare:

- **Extinderea Funcționalităților:** Introducerea unor noi operații matematice, cum ar fi găsirea rădăcinilor polinoamelor, ar putea oferi utilizatorilor unelte și mai puternice pentru explorarea matematicii polinoamelor.
- **Optimizarea Performanței**: Analiza și îmbunătățirea algoritmilor pentru operații complexe, precum împărțirea și integrarea, ar putea crește eficiența calculului, în special pentru polinoamele de grad înalt.
- **Interfață Web sau Mobilă**: Dezvoltarea unei versiuni web sau mobile a aplicației ar putea extinde accesibilitatea acesteia, permițând utilizatorilor să efectueze calcule de pe diverse dispozitive.
- **Integrarea cu Alte Aplicații Matematice**: Permiterea importului și exportului de polinoame din și către alte aplicații matematice ar putea facilita utilizarea aplicației într-un context educațional sau de cercetare mai larg.

7. Bibliografie

JUnit 5 User Guide, disponibil pe https://junit.org/junit5/docs/current/user-guide/. Acest ghid a fost esențial pentru implementarea și executarea testelor unitare, asigurând o metodologie structurată și eficientă pentru verificarea corectitudinii logicii aplicației.

Oracle Java Documentation, accesibil pe https://docs.oracle.com/javase/8/docs/. Documentația oficială Java a oferit referințe complete și detaliate pentru API-urile Java utilizate în aplicație, inclusiv colecții, fluxuri de intrare/ieșire și elemente de interfață grafică.

Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Addison-Wesley Professional, ISBN: 978-0201633610. Această carte a introdus modele de design relevante pentru structurarea și optimizarea codului aplicației, cum ar fi Factory, Singleton și Strategy, care au îmbunătățit modularitatea și reutilizabilitatea componentelor software.

Mathematics for Computer Science, Eric Lehman, F. Thomson Leighton, and Albert R. Meyer, disponibil online. Acest material a oferit fundamentul teoretic necesar pentru înțelegerea algoritmilor matematici utilizați în operațiile cu polinoame, contribuind la dezvoltarea unor soluții eficiente și corecte din punct de vedere matematic.