DOCUMENTATIE

TEMA *1*

NUME STUDENT: Ardelean Mitică-Mario
GRUPA: 30224

CUPRINS

1. Obiectivul temei

Obiectivul principal al acestei teme este de a proiecta, respectiv implementa un calculator polinomial, acesta având o interfață grafică dedicată prin care utilizatorul poate insera 1 sau 2 polinoame, poate selecta operația matematică dorită și poate vizualiza rezultatul operației.

Obiectivele secundare al acestei teme sunt următoarele:

- Analiza problemei și identificarea cerințelor (Capitolul 2)
- Proiectarea calculatorului polinomial (Capitolul 3)
- Implementarea calculatorului polinomial, respectând structura și cerințele acestuia (Capitolul 4)
- Testarea calculatorului polinomial (Capitolul 5)

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

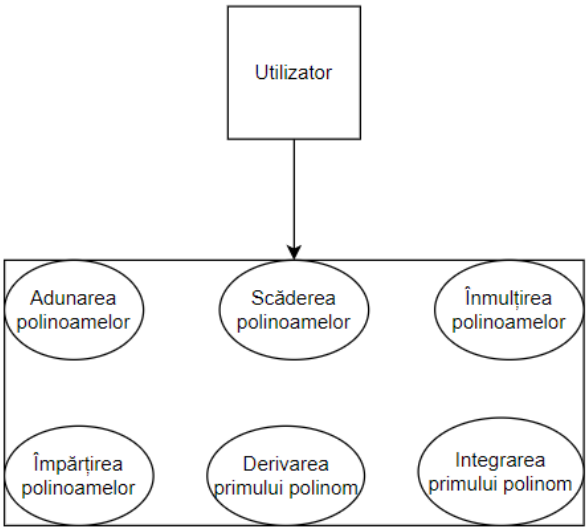
Cerințele funcționale sunt următoarele:

- Calculatorul polinomial ar trebui să permită utilizatorului să introducă polinoame
- Calculatorul polinomial ar trebui să permită utilizatorului să selecteze operația matematică dorită(adunare, scădere, înmulțire, împărțire, derivare, integrare)
- Calculatorul polinomial ar trebui să afișeze rezultatul operației alese

Cerințele non-funcționale sunt următoarele:

- Calculatorul polinomial trebuie intuitiv și ușor de utilizat de către utilizator
- Calculatorul polinomial trebuie să afișeze rezultatul operațiilor matematice într-un timp rezonabil

Diagrama de use-case:



Descrierile de use-case:

- Adunarea polinoamelor:
  - Actor primar: utilizatorul
  - Principalul scenariu de succes:
    1) Utilizatorul introduce 2 polinoame în interfața grafică
    2) Utilizatorul selectează operația “+”
    3) Calculatorul polinomial efectuează adunarea celor 2 polinoame și afișază rezultatul

- Secvență alternativă -> Polinoame incorecte:
    1) Utilizatorul introduce 1 sau 2 polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)
    2) Se afișază un mesaj de eroare în interfața grafică
    3) Scenariul revine la pasul 1

➢ Scăderea polinoamelor:
- Actor primar: utilizatorul
- Principalul scenariu de succes:
    1) Utilizatorul introduce 2 polinoame în interfața grafică
    2) Utilizatorul selectează operația "-"
    3) Calculatorul polinomial efectuează scăderea celor 2 polinoame și afișază rezultatul
- Secvență alternativă -> Polinoame incorecte:
    1) Utilizatorul introduce 1 sau 2 polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)
    2) Se afișază un mesaj de eroare în interfața grafică
    3) Scenariul revine la pasul 1

➢ Înmulțirea polinoamelor:
- Actor primar: utilizatorul
- Principalul scenariu de succes:
    1) Utilizatorul introduce 2 polinoame în interfața grafică
    2) Utilizatorul selectează operația "*"
    3) Calculatorul polinomial efectuează înmulțirea celor 2 polinoame și afișază rezultatul
- Secvență alternativă -> Polinoame incorecte:
    1) Utilizatorul introduce 1 sau 2 polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)
    2) Se afișază un mesaj de eroare în interfața grafică
    3) Scenariul revine la pasul 1

➢ Împărțirea polinoamelor:
- Actor primar: utilizatorul
- Principalul scenariu de succes:
    1) Utilizatorul introduce 2 polinoame în interfața grafică
    2) Utilizatorul selectează operația "/"
    3) Calculatorul polinomial efectuează împărțirea celor 2 polinoame și afișază rezultatul
- Secvență alternativă -> Polinoame incorecte:
    1) Utilizatorul introduce 1 sau 2 polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)
    2) Se afișază un mesaj de eroare în interfața grafică
    3) Scenariul revine la pasul

➢ Derivarea primului polinom:
- Actor primar: utilizatorul
- Principalul scenatiu de succes:
    1) Utilizatorul introduce un polinom (în primul TextField) în interfața grafică
    2) Utilizatorul selectează operația "d/dx"
    3) Calculatorul polinomial efectuează derivarea polinomului și afișază rezultatul
- Secvență alternativă -> Polinom incorect:
    1) Utilizatorul introduce polinomul (în primul TextField) greșit (de exemplu, cu 2 sau mai multe variabile)
    2) Se afișază un mesaj de eroare în interfața grafică
    3) Scenariul revine la pasul 1

➢ Integrarea primului polinom:
- Actor primar: utilizatorul
- Principalul scenariu de succes:
    1) Utilizatorul introduce un polinom (în primul TextField) în interfața grafică
    2) Utilizatorul selectează operația "∫"
    3) Calculatorul polinomial efectuează integrarea polinomului și afișază rezultatul

- Secvență alternativă -> Polinom incorect:
    1) Utilizatorul introduce polinomul (în primul TextField) greșit (de exemplu, cu 2 sau mai multe variabile)
    2) Se afișază un mesaj de eroare în interfața grafică
    3) Scenariul revine la pasul 1

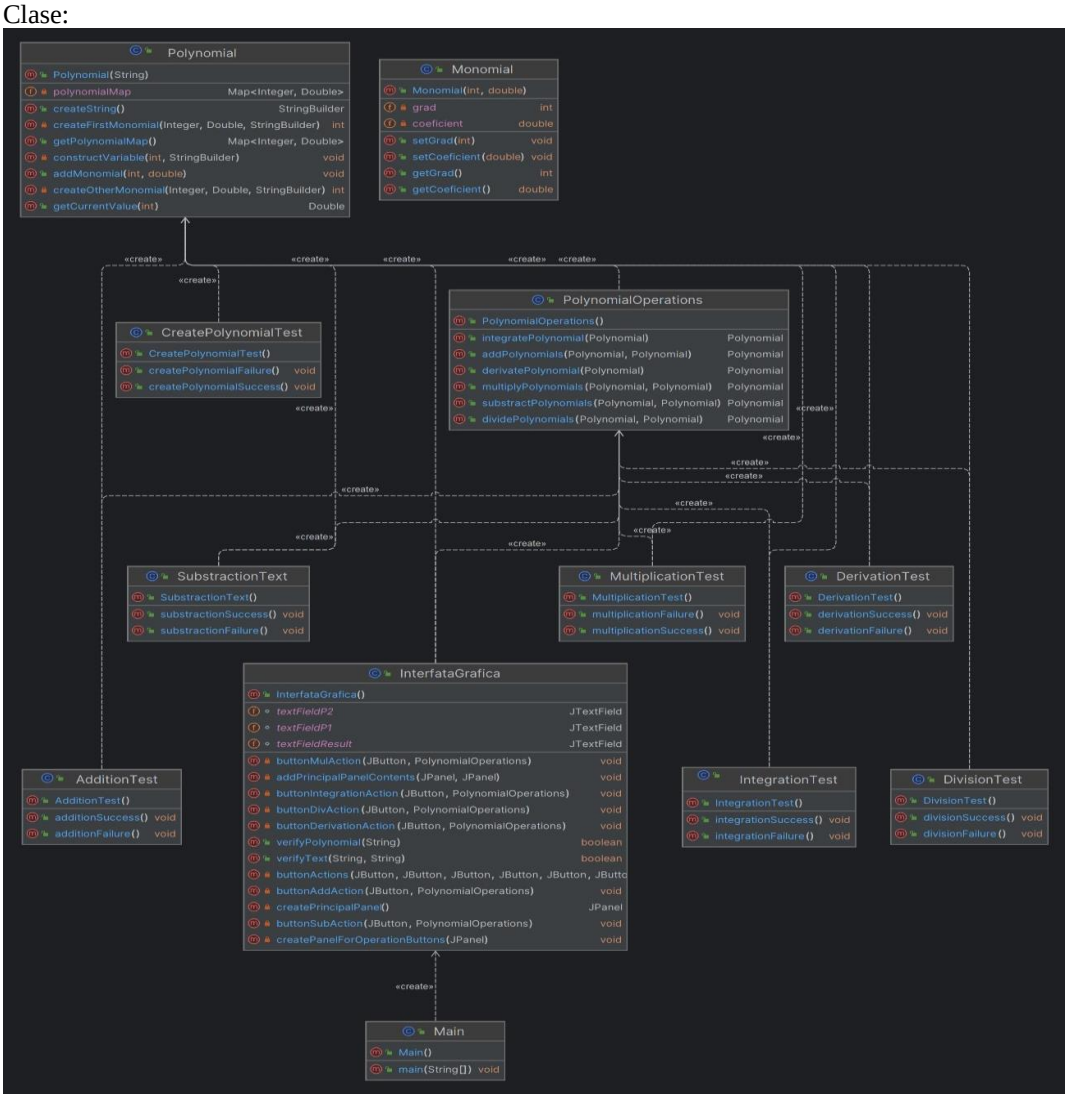
3. Proiectare

1) Proiectarea OOP

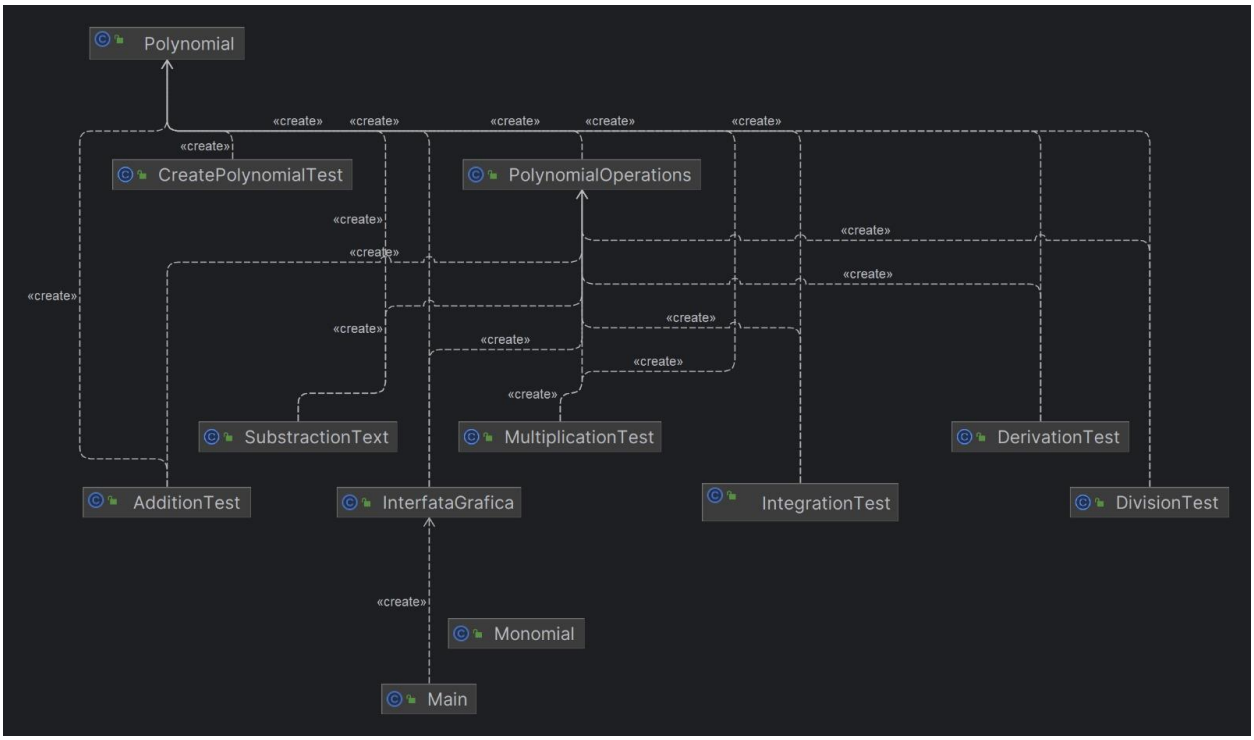
În proiectarea aplicației noastre de calculator polinomial folosind OOP, am identificat principalele entități și concepte din domeniul problemei noastre. Acestea includ polinoamele, operațiile matematice și interacțiunea cu utilizatorul. Am organizat aceste entități în clase corespunzătoare pentru a gestiona funcționalitățile și datele asociate.

2) Diagramele UML de clase și de pachete

Clase:



Pachete:



3) Structurile de date folosite
   Structura de date de tip TreeMap: o colecție care stochează perechi cheie-valoare și le organizează într-un arbore pe baza ordinii cheilor. În cazul nostru, cheie este gradul polinomului, iar valoarea este coeficientul polinomului.
4) Interfețe definite:
   Interfața `InterfataGrafica`: utilizează biblioteca Swing pentru a crea și gestiona componente precum ferestre, panouri și butoane. Această interfață conține următoarele:
   - 3 TextField-uri, primul pentru primul polinom, al doilea pentru al doilea polinom și al treilea pentru rezultat
   - 6 butoane, fiecare pentru fiecare operație în parte (adunare, scădere, înmulțire, împărțire, derivare și integrare)
   - metode pentru gestionarea acțiunilor butoanelor, inclusiv validarea polinoamelor introduse și efectuarea operațiilor matematice corespunzătoare
   - clase din alte pachete, cum ar fi `PolynomialOperations` și `Polynomial`, pentru a efectua operațiile matematice necesare.
5) Algoritmi folosiți:
   Algoritmi pentru operații matematice: am implementat algoritmi pentru adunare, scădere, înmulțire, împărțire, derivare și integrare pe baza regulilor algebrei polinoamelor.

4. Implementare

a) Clasa `Monomial`:

Câmpuri:

1. `grad`: un câmp privat de tip int care stochează gradul monomului
2. `coeficient`: un câmp privat de tip double care stochează coeficientul monomului.

Metode importante:

1. `getGrad()`: metodă care returnează gradul monomului

2. `setGrad(int grad)`: metodă care setează gradul monomului la valoarea specificată
3. `getCoeficient()`: metodă care returnează coeficientul monomului
4. `setCoeficient(double coeficient)`: metodă care setează coeficientul monomului la valoarea specificată.

b) Clasa `Polynomial`

Câmpuri:

1. `polynomialMap`: Un map care stochează termenii polinomului, cu gradul ca cheie și coeficientul ca valoare.

Metode importante:

1. `addMonomial(int degree, double coefficient)`: adaugă un monom la polinom, actualizând map-ul termenilor polinomului
2. `createString()`: construiește șirul de caractere reprezentând polinomul pe baza termenilor stocați în map-ul de termeni și returnează acest șir
3. `createFirstMonomial(Integer degree, Double coefficient, StringBuilder resultString)`: construiește primul monom din polinom și îl adaugă la șirul rezultat
4. `createOtherMonomial(Integer degree, Double coefficient, StringBuilder resultString)`: construiește ceilalți monomi din polinom și îi adaugă la șirul rezultat
5. `constructVariable(int exponent, StringBuilder resultString)`: construiește partea care reprezintă variabila (`x`) în cazul în care monomul are un exponent diferit de zer
6. `getPolynomialMap()`: returnează map-ul de termeni al polinomului
7. `getCurrentValue(int key)`: Returnează valoarea coeficientului pentru un anumit grad al polinomului.

c) Clasa `PolynomialOperations`

Câmpuri: -

Metode importante:

1. `addPolynomials(Polynomial x, Polynomial y)`: adaugă două polinoame și returnează rezultatul
2. `substractPolynomials(Polynomial x, Polynomial y)`: scade un polinom din altul și returnează rezultatul
3. `multiplyPolynomials(Polynomial x, Polynomial y)`: înmulțește două polinoame și returnează rezultatul
4. `dividePolynomials(Polynomial x, Polynomial y)`: împarte un polinom la altul și returnează rezultatul
5. `derivatePolynomial(Polynomial x)`: calculează derivata unui polinom și returnează rezultatul
6. `integratePolynomial(Polynomial x)`: calculează integrala unui polinom și returnează rezultatul.

d) Clasa `InterfataGrafica`

Câmpuri:

1. trei câmpuri de text (textFieldP1, textFieldP2 și textFieldResult): pentru introducerea polinoamelor și afișarea rezultatului
2. butoanele buttonAdd, buttonSub, buttonMul, buttonDiv, buttonDerivation și buttonIntegration: pentru execuția operațiilor matematice.

Metode importante:

1. `createPrincipalPanel()`: responsabilă de inițializarea și configurarea panoului principal al interfeței grafice

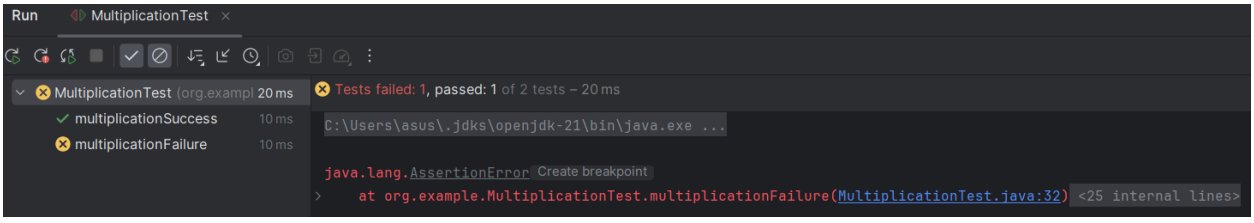
2. `addPrincipalPanelContents()`: adaugă etichetele și câmpurile de text pentru introducerea polinoamelor, butoanele pentru operații și câmpul pentru afișarea rezultatului pe panoul principal
3. `createPanelForOperationButtons()`: inițializează și configurează butoanele pentru diferitele operații matematice
4. `buttonActions()`: atribuie acțiuni butoanelor pentru operațiile matematice folosind obiectul `PolynomialOperations`
5. `verifyPolynomial()`: verifică dacă polinoamele introduse sunt formate corect
6. `verifyText()`: verifică corectitudinea polinoamelor introduse de către utilizator
7. Lista de metode: `buttonAddAction()`, `buttonSubAction()`, `buttonMulAction()`, `buttonDivAction()`, `buttonDerivationAction()` și `buttonIntegrationAction()` sunt responsabile pentru efectuarea operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare a polinoamelor și afișarea rezultatului în câmpul corespunzător.

5. Rezultate

Avem câte o clasă în parte pentru fiecare operație de testat + construirea polinomului, și anume:

1) CreatePolynomialTest:
   - Metoda ‘createPolynomialSuccess()’: testează dacă un polinom s-a creat corect
   - Metoda ‘createPolynomialFailure()’: testează dacă un polinom s-a creat greșit
2) AdditionTest:
   - Metoda ‘additionSuccess()’: testează dacă adunarea între 2 polinoame se face corect
   - Metoda ‘additionFailure()’: testează dacă adunarea între 2 polinoame se face greșit
3) SubstractionTest:
   - Metoda ‘substractionSuccess()’: testează dacă scăderea între 2 polinoame se face corect
   - Metoda ‘substractionFailure()’: testează dacă scăderea între 2 polinoame se face greșit
4) MultiplicationTest:
   - Metoda ‘multiplicationSuccess()’: testează dacă înmulțirea între 2 polinoame se face corect
   - Metoda ‘multiplicationFailure()’: testează dacă înmulțirea între 2 polinoame se face greșit
5) DivisionTest:
   - Metoda ‘divisionSuccess()’: testează dacă împărțirea între 2 polinoame se face corect
   - Metoda ‘divisionFailure()’: testează dacă împărțirea între 2 polinoame se face greșit
6) DerivationTest:
   - Metoda ‘derivationSuccess()’: testează dacă derivarea unui polinom se face corect
   - Metoda ‘derivationFailure()’: testează dacă derivarea unui polinom se face greșit
7) IntegrationTest:
   - Metoda ‘integrationSuccess()’: testează dacă integrarea unui polinom se face corect
   - Metoda ‘integrationFailure()’: testează dacă integrarea unui polinom se face greșit.

Un exemplu de rezultat al unei testări:



6. Concluzii

În concluzie, această temă a fost una destul de interesantă, prin prisma faptului că am creat, de la 0, un calculator care calculează operațiile de bază între polinoame. De asemenea, utilizarea unei interfețe grafice pentru inserarea polinoamelor, respectiv utilizarea unei operații, a fost un lucru pe care nu l-am mai făcut până acum, deci a fost interesant.

Din această temă am învățat cum se pot implementa operațiile de bază pe polinoame în limbajul de programare Java. De asemenea, am învățat cum să utilizez structura de date de tip TreeMap, ceea ce o să mă ajute foarte mult în viitorul apropiat.

Niște posibilități de dezvoltări ulterioare ar putea fi:

- Înfrumusețarea interfeței grafice(adăugare culori, font mai frumos, etc.)
- Verificarea total corectă a polinoamelor(adică să pot scrie și “x^2 + 2”, nu doar “x^2+2”)
- Alte operații matematice(derivata a 2-a a polinomului, integrarea pe un anumit interval [a,b], etc.)

7. Bibliografie

1. *https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf*
2. *https://regexr.com/*
3. *https://en.wikipedia.org/wiki/Polynomial_long_division*
4. *https://www.geeksforgeeks.org/treemap-in-java/*
5. *https://dsrl.eu/courses/pt/materials/PT_2024_A1_S3.pdf*