DOCUMENTATIE

TEMA *1*

NUME STUDENT: Catruc Alexandru- Dan
GRUPA: 30228

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei este realizarea unui calculator polinomial care efectuează diferite operații între polinoame, lucru care este utilizat într-o multitudine de domenii precum cel didactic, financiar, științific etc.

Ca și obiective secundare se numără:

- Realizarea unei interfețe prietenoasă cu utilizatorul care să fie ușor utilizabilă.(4)
- Validarea datelor introduse de utilizator pentru a se asigura corectitudinea datelor de la input (în cazul nostru forma corectă a polinomului). (4)
- Optimizarea codului pentru a se asigura faptul ca aplicația rulează cât mai rapid și optim posibil.(4)
- Comentarea și explicarea codului pentru a fi înțeles și ușor de manipulat în cazul unei viitoare implementări.(4)
- Testarea codului pentru a se asigura faptul că funcționează adecvat.(5)
- Realizarea unei structuri de cod care permite adăugarea de noi funcționalități în viitor.(4)

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

Cerințe funcționale:

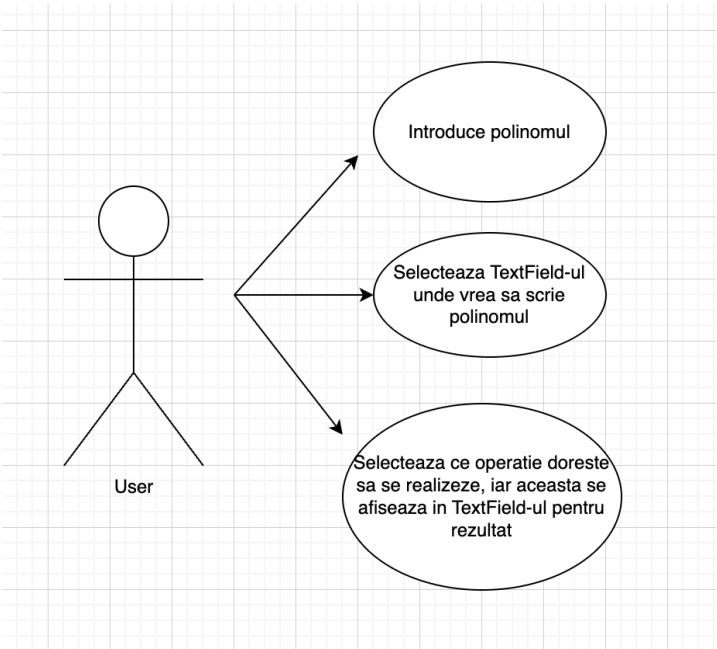
- Utilizatorul trebuie să poată introduce polinoame și operații cu acestea (adunare, scădere, înmulțire, derivare, integrare).
- Utilizatorul trebuie să poată vedea rezultatul operațiilor cu polinoame.
- Aplicația trebuie să poată gestiona polinoame cu coeficienți întregi sau raționali.
- Aplicația trebuie să poată afișa polinoamele în ordine descrescătoare a puterilor.

Cerințe non-funcționale:

- Aplicația trebuie să fie scrisă în limbajul Java.
- Interfața grafică a utilizatorului trebuie să fie intuitivă și ușor de utilizat.
- Aplicația trebuie să ofere o performanță bună chiar și pentru polinoame mari.
- Aplicația trebuie să fie robustă și să ofere mesaje de eroare clare în cazul în care utilizatorul introduce date invalide sau dacă apare o eroare în timpul executării.
- Codul sursă al aplicației trebuie să fie bine documentat și să respecte standardele de codare Java.

Use-cases:

1. Adăugarea unui polinom:
    - Utilizatorul introduce un polinom într-un câmp de introducere
    - Aplicația validează polinomul și afișează un mesaj de eroare dacă polinomul nu este valid
    - Aplicația adaugă polinomul în lista de polinoame
2. Adunarea a două polinoame:
    - Utilizatorul selectează două polinoame din lista de polinoame
    - Aplicația adună cele două polinoame și afișează rezultatul
3. Înmulțirea a două polinoame
    - Utilizatorul selectează două polinoame din lista de polinoame
    - Aplicația înmulțește cele două polinoame și afișează rezultatul
4. Derivarea unui polinom
    - Utilizatorul selectează un polinom din lista de polinoame
    - Aplicația derivă polinomul și afișează rezultatul
5. Integrarea unui polinom:
    - Utilizatorul selectează un polinom din lista de polinoame
    - Aplicația integrează polinomul și afișează rezultatul
6. Scăderea a două polinoame:
    - Utilizatorul selectează două polinoame din lista de polinoame
    - Aplicația scade cel de-al doilea polinom din primul polinom și afișează rezultatul
7. Împărțirea a două polinoame:
    - Utilizatorul selectează două polinoame din lista de polinoame
    - Aplicația împarte primul polinom la cel de-al doilea polinom și afișează rezultatul, împreună cu restul
    - Dacă împărțirea nu este posibilă, aplicația afișează un mesaj de eroare.



3. Proiectare

Diagrama de pachete:

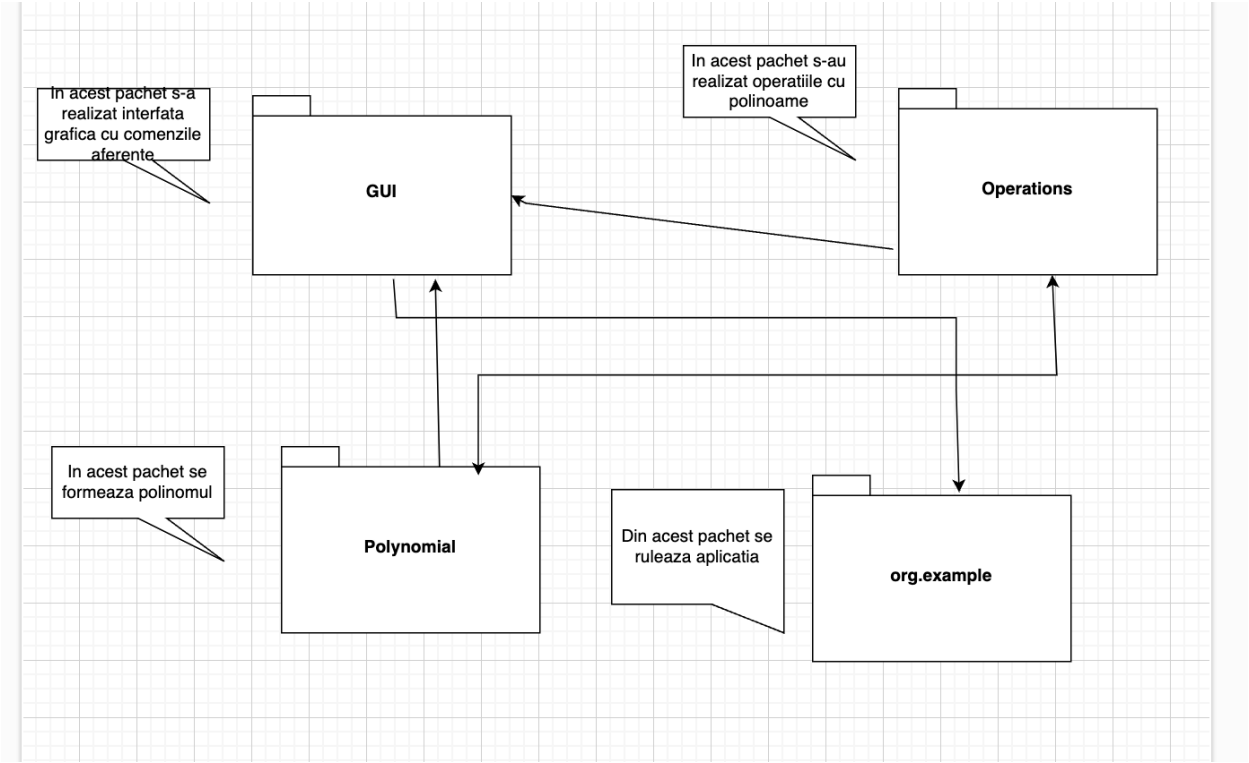
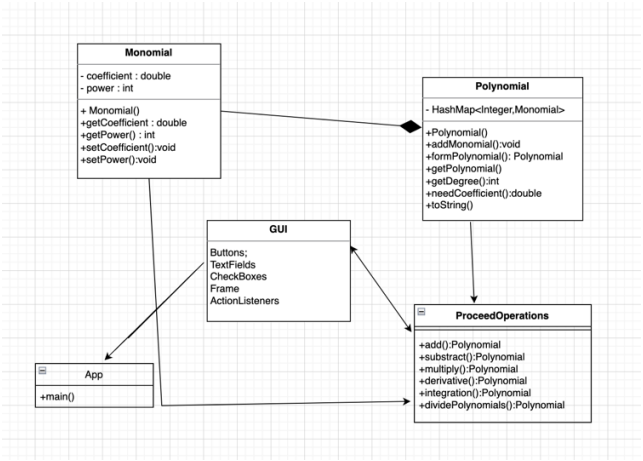


Diagrama de clase:



4. Implementare

Pentru implementarea acestui proiect s-au folosit 5 clase aparținând unui număr de 4 pachete.

Primul pachet este cel numit **Polynomial** care cuprinde clasele **Polynomial** si **Monomial**. Un polinom va fi format din mai multe monoame. În clasa **Monomial** vom avea câmpurile **coefficient** și **power** (puterea și coeficientul monomului), iar ca metode importante vom avea **contructorul**, **gettere** și **settere**.
Clasa **Polynomial** va fi formată dintr-un **HashMap<Integer, Monomial>** , colectie de date în care se va stoca fiecare monom în urma parsării. Ca și metode vom avea **constructorul**, metoda **addMonomial(Monomial monomial)** care va primi un monom și îl va adăuga în HashMap (în cazul în care se vor scrie în TextField două monoame cu aceeași putere, se vor aduna coeficienții, altfel se adaugă monomul). O a doua metodă și una dintre cele mai importante este metoda **formPolynomial(String polynomialInput)** care primește un String și va forma polinomul. Se va verifica dacă forma stringului este validă, în caz pozitiv se va realiza formarea, în caz negativ (când inputul nu este valid) se va arunca o excepție. Celelalte 3 metode sunt **getDegree()** – returnează gradul polinomului, **needCoefficient(int power)** – returnează coeficientul de la o anumită putere, **toString()** – oferă forma finală a polinomului.

Al doilea pachet este cel numit **Operations** și cuprinde toate metodele necesare efectuării operațiilor cu polinoame. Are o singură clasă, numită **ProceedOperations**. Această clasă are metodele **add(Polynomial p1, Polynomial p2)** – realizează adunarea dintre cele două polinoame, **substract(Polynomial p1, Polynomial p2)** – realizează scăderea dintre cele două polinoame, **multiply(Polynomial p1, Polynomial p2)** – realizează înmulțirea dintre cele două polinoame, **derivative(Polynomial polynomial)** – realizează derivarea unui polinom, **integration(Polynomial polynomial)** – realizează integrarea unui polinom, **dividePolynomials(Polynomial divident, Polynomial divisor)** - realizează împărțirea a două polinoame.

Cel de-al treilea pachet este **org.example** care conține clasa **App** de unde se va rula aplicația (cuprinde o metodă **main**).

Al patrulea pachet este cel numit **GUI** și reprezintă interfața cu utilizatorul. Este format dintr-o clasă numită **Calculator**. În clasă se crează o metodă numită **calculatorUserInterface** unde se realizează designul calculatorului. Avem un JFrame care conține 3 TextField – uri (unul pentru rezultat și două pentru introducerea polinoamelor). Calculatorul are și o tastatură numerică, simbol pentru semnul de ori (‘*’), semnul de plus (‘+’), semnul de minus (‘-’), semnul de ridicare la putere (‘^’), un buton cu simbolul pentru caracterul ‚X’ și un buton de delete (‘del’) pentru a șterge câte un caracter in caz de greșeală. Mai există 6 butoane pentru operațiile pe polinoame care au nume sugestive. În interiorul clasei există toate funcționalitățile necesare pentru a face aplicația să ruleze: ActionListeners, apelarea metodelor pentru operații etc.

5. Rezultate

Testarea operațiilor cu polinoame au fost realizată cu ajutorul JUnit. A fost creată clasa **TestOperations** în interiorul căreia au fost implementate în primul rând două metode de creare a două polinoame cu valori implicite. După aceea s-a trecut la implementarea metodelor de testare pentru fiecare operație, în cazul acestui proiect au fost testate în ordinea implementării.

În interiorul fiecărei metode de testare se crează câte 3 polinoame, cu excepția metodei pentru derivare și celei de integrare unde se crează doar două. Primele două vor fi cele create cu ajutorul metodelor menționate mai sus, iar cel de-al treilea va reprezenta rezultatul (în cazul derivării și integrării primul polinom va fi cel ce urmează a fi integrat, iar cel de-al doilea va reprezenta rezultatul integrării). Metoda va mai avea un bloc **try** unde se va realiza operația, iar dacă rezultatul acesteia coincide cu

valoarea de pe output, se va afișa în consolă un mesaj care va reprezenta trecerea cu succes a testului. În caz contrar, se va afișa un alt mesaj care va semnala picarea testului.

Spre exemplu, in cazul operației de adunare:

```
no usages  new *
@Test
public void testAddition() {
    Polynomial polynomialAddition = new Polynomial();
    Polynomial polynomial1 = createPolynomial1();
    Polynomial polynomial2 = createPolynomial2();
    ProceedOperations proceedOperations = new ProceedOperations();
    try {
        polynomialAddition = proceedOperations.add(polynomial1, polynomial2);
        assertEquals( expected: "3.0x^3+3.0x^2+4.0x+2.0", polynomialAddition.toString());
        System.out.println("Test for addition passed");
    } catch (Exception e) {
        System.out.println("Test for addition failed");
    }

}
```

```
Test for addition passed
```

În urma testării operațiilor, s-au obținut următoarele rezultate:

```
Test for derivation passed
Test for multiplication passed
Test for addition passed
Test for integration passed
Test for substraction passed
Test for division passed
```

6. Concluzii

În urma realizării acestui proiect au fost învățate o sumedenie de noi lucruri în mediul Java precum: validarea unui anumit format pentru input (în acest caz polinoamele introduse de la tastatură), utilizarea testării cu JUnit pentru operațiile cu polinoame, manipularea mult mai bună a colecțiilor de date (iterații, adăugare, actualizare). În plus, au fost îmbunătățite scrierea de cod în Java, modul de organizare al unei aplicații, precum și realizarea unei bune interfețe cu utilizatorul.

În concluzie, implementarea acestui calculator a avut un rol benefic în dezvoltarea de noi aptitudini în Java dar și în învățarea de a realiza un proiect într-un mod bine organizat (împărțirea pe pachete și legarea bună dintre clase).

7. Bibliografie

1. Java HashMap - https://www.w3schools.com/java/java_hashmap.asp
2. How to for each the HashMap? - https://stackoverflow.com/questions/4234985/how-to-for-each-the-hashmap
3. Using regex - https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression
4. How to focus on a JTextField? - http://www.java2s.com/example/java-utility-method/jtextfield-select-index-0.html#:~:text=Automatically%20selects%20the%20field's%20text%20when%20it%20gains%20focus.&text=Add%20a%20focusGained%2Dlistener%20to,(FocusEvent%20e)%20%7B%20SwingUtilities.
5. Polynomial toString() - https://stackoverflow.com/questions/13424302/java-tostring-method-for-polynomial-terms