DOCUMENTATIE

TEMA *NUMARUL\_1*

NUME STUDENT: IRIMIE DARIA-ALEXIA

GRUPA: 30228

# CUPRINS

[1. Obiectivul temei 3](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297886)

[3. Proiectare 4](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297887)

[4. Implementare 6](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297888)

[5. Rezultate 9](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297889)

[6. Concluzii 11](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297890)

[7. Bibliografie 11](file:///D:\Utilizator-nu%20sterge\Downloads\PT2023_Template_Documentatie_RO%20(2).doc#_Toc95297891)

# Obiectivul temei

**Obiectivul principal**

Obictivul acestei teme este dezvoltarea unei intefete caracteristice unui calculator, care opereaza cu polinoame. Calculatorul are abilitatea de a efectua operatii pe aceste polinoame, cum ar fi adunarea, scaderea, care vor fi detaliate in analiza problemei.

**Obiectivele secundare**

* Analiza problemei, descrierea abordarii de implementare**-** Capitolul 2
* Proiectarea calculatorului. Organizarea codului, definirea claselor pe pachete in functie de rolul acestora, cum ar fi : Logic pentru logica din spate a programului, GUI pentru interfata grafica – Capitolul 3
* Implementarea calculatorului de polinoame – Capitolul 4
* Testarea functionalitatii calculatorului – Capitolul 5

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Use-case diagram

*Diagram

Description automatically generated*

Utilizarea calculatorului de catre utilizator este foarte simpla. Tot ce trebuie sa faca acesta este sa introduca doua polinoame, fapt care este sugerat si in mesajele indrumatoare din interfata. Acesta trebuie sa introduca polinomul dupa un format specific, alcatuit dintr-un coeficient real si un exponent intreg, dupa cum este prezentat in exemplul urmator „4x^3+2x^1+6x^0”. Daca formatul introdus nu este corespunzator, utilizatorul va primi un mesaj de eroare care sa ii indice faptul ca polinomul pe care l-a introdus nu are structura valida. Daca formatul este corespunzator, mai departe acesta trebuie doar sa aleaga operatia pe care doreste sa o efectueze. Operatiile de care dispune programul sunt :

* Adunare
* Scadere
* Inmultire
* Derivare
* Integrare

In functie de operatia dorita, avem doua posibilitati :

* Cazul in care dorim sa facem derivare sau integrare, necesita introducerea unui singur polinom, in primul TextField
* Cazul in care dorim sa facem adunare, scadere sau inmultire, necesita introducerea a doua polinoame, in ambele TextFiel-uri.

In final, rezultatul operatiei alese va fi afisat in TextField-ul corespunztor rezultatului, in partea de jos a calculatorului.

# Proiectare

**Organizarea programului pe clase si pachete**

Proiectul este impartit pe pachete, in functie de functionalitate:

* GUI (Grafical User Interface)

Reprezinta partea de interfata grafica a programului. Aici sunt definite toate componentele vizuale ale calculatorului, butoane, textField-uri si Label-uri.

De asemenea, tot in acest pachet se afla implementarea functionala a butoanelor.

* Logic

Reprezinta partea de implementare a programului si anume operatiile ce se pot aplica pe un polinom (Adunare, Scadere, etc).

Aici avem implementate metode corespunzatoare fiecarei operatii, pentru o mai buna organizare a clasei Polinom si un aspect mai usor de inteles la apelarea metodelor.

De asemenea, avem si clasa PolynomialReader care gestioneaza citirea unui polinom sub forma de String si parsarea sa intr-o structura care retine exponent si coeficient. Aceasta structura se va folosi la implementare.

* Model

Contine clasa care modeleaza polinomul. In clasa Polinom avem definita o structura de tipul Map care stocheaza polinomul si metodele de acces (getters si setters) prin care putem accesa elementele acestuia.

De asemenea, tot in acest pachet se afla si exceptia care se arunca daca inputul introdus nu este unul valid.

In afara acestor pachete am implementat clasa Main care comunica cu interfata si clasa pentru testarea unitara a operatiilor.

Graphical user interface, text, application

Description automatically generated

Diagrama de pachete

Diagram

Description automatically generated

Diagrama de clase

Diagram

Description automatically generated

# Implementare

* **Main :**

Aceasta clasa contine doar metoda main.

In metoda main se instantiaza un obiect al clasei CalculatorInterface. Astfel, se initializeaza fereastra aplicatiei.

* **Polynomial :**

Campuri importante :

Aceasta clasa are ca si camp o structura de tipul Map, cu numele de terms, care stocheaza exponent si coeficient de tipul Integer si Double.

Metode importante :

* addTerm : Aceasta metoda adauga termenul urmator(exponent + coeficient) la polinom.

* **InvalidInputException :**

Nu contine campuri

Contine doar constructorul care primeste ca si parametru un mesaj sub forma de String.

* **PolynomialReader :**

Campuri importante :

* REGEX, sub forma de String, camp final
* pattern care primeste REGEX

Acest pattern reprezinta tiparul sub care trebuie introdus input-ul

Metode importante :

* Read : Aceasta metoda ia un sir ca intrare si returneaza un obiect de tipul Polynomial. Metoda arunca o InvalidInputException daca intrarea nu este valida. Se creeaza o noua instanta a clasei Polynomial pentru a pastra termenii polinomului. Foloseste un model de expresie regulata pentru a potrivi si a extrage termenii polinomului din sirul de intrare. Pentru fiecare termen potrivit, extrage valorile coeficientului si puterii si le adauga la obiectul Polinom. Daca nu se potrivesc termenii, se afiseaza o exceptie InvalidInputException.
* **PolynomialOperations :**

Nu contine campuri

Metode importante :

* AddPolynomials : Aceasta metoda aduna doua obiecte Polynomial si returneaza obiectul Polynomial rezultat. Utilizeaza un HashMap pentru a stoca termenii rezultati ai polinomului. Metoda itereaza peste termenii polinomului p1 si adauga fiecare termen la HashMap. Dca un termen cu acelasi exponent exista deja in HashMap, acesta adauga coeficientul noului termen la coeficientul existent.
* SubPolynomials : Aceasta clasa are o implementare asemanatoare cu cea a clasei AddPolynomials. Diferenta dintre cele doua metode consta in scaderea celor doua polinoame si returnarea polinomului rezultat si implicit in scaderea coeficientilor termenilor cu acelasi exponent.
* DerivePolynomial : Aceasta metoda primeste ca parametru un obiect Polynomial si returneaza derivata polinomului ( un obiect Polynomial rezultat). Ca si la metodele AddPolynomials si SubPolynomials, metoda aceasta utilizeaza un HashMap pentru a stoca termenii rezultati ai polinomului si itereaza peste termenii polinomului si adauga fiecare termen la HashMap. Calculul derivatei polinomului se face prin modificarea exponentului fiecarui termen cu exponent – 1 si modificarea coeficientului fiecarui termen prin inmultirea sa cu exponentul initial. In plus, metoda contine o verificare, astfel, in cazul in care exponentul este 0, acesta nu va mai fi scris.
* IntegratePolynomial : Aceasta metoda este asemanatoare cu metoda DerivePolynomials. Diferenta consta in calcularea integralei polinomului, in loc de a derivatei. Calculul se face prin modificarea exponentului fiecarui termen cu exponent + 1 si modificarea coeficientului prin impartirea acestuia la noul exponent (exponent + 1).
* MultiplyPolynomial : Aceasta metoda inmulteste doua obiecte Polynomial si returneaza obiectul Polynomial rezultat. Metoda ia ca intrari doua obiecte Polynomial p1 si p2. Se creeaza o noua instanta a clasei Polynomial pentru a pastra termenii rezultati ai polinomului. Se creeaza un nou HashMap pentru a pastra termenii polinomului rezultat. Metoda itereaza peste termenii lui p1, obtinand coeficientul si exponentul fiecarui termen. Apoi itereaza peste termenii lui p2, obtinand coeficientul si exponentul fiecarui termen. Inmulteste coeficientii celor doi termeni pentru a obtine noul coeficient pentru termenul rezultat. Se adauga exponentii celor doi termeni pentru a obtine noul exponent pentru termenul rezultat. Dacă un termen cu acelasi exponent exista deja in HashMap, acesta adauga noul coeficient la coeficientul existent. In cele din urma, stabileste termenii polinomului rezultat la termenii din HashMap. Returnează obiectul polinom rezultat.
* **CalculatorInterface :**

Aceasta clasa se ocupa de partea grafica a programului. Reprezinta interfata vizuala si logica din spatele programului.

Campuri importante :

* 3 TextField-uri :

fieldFirstPol

fieldSecondPol

fieldResult

* 5 label-uri :

textTitle

textFirstPol

textSecondPol

textOperation

textResult

* 6 butoane :

btnAdd

btnSub

btnMultiply

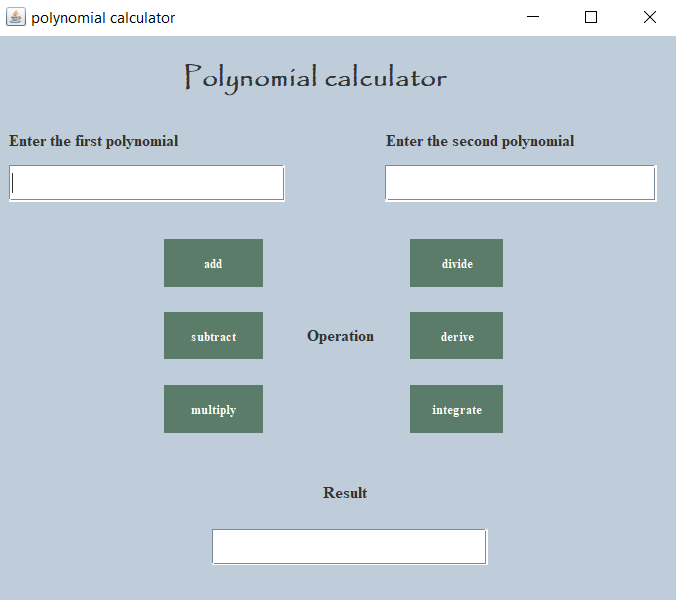
btnDivide

btnDerive

btnIntegrate

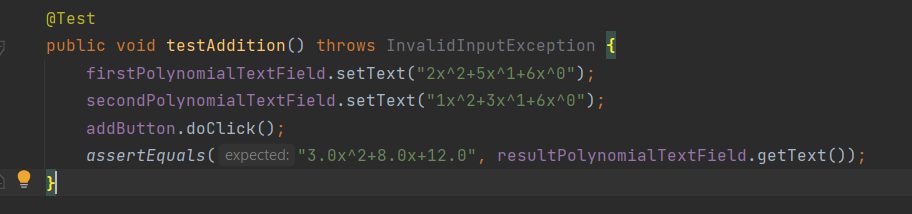
Aceasta clasa implementeaza ActionListener pentru fiecare buton, utilizand metodele implementate pentru operatii, in clasa PolynomialOperations.

Interfata grafica arata in felul urmator :



# Rezultate

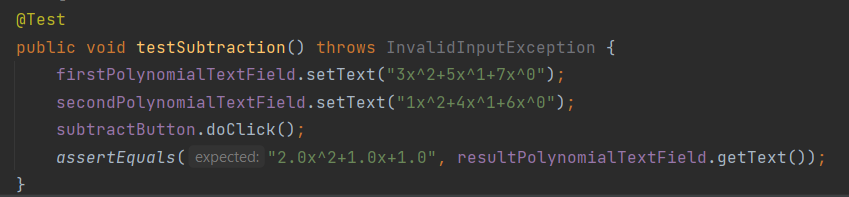
* Adunare



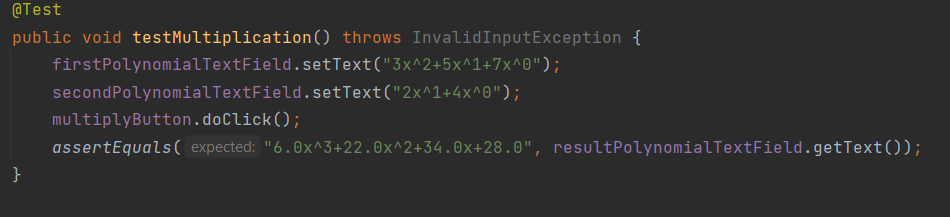
In acest caz de testare, sunt setate doua polinoame cu valorile care se pot observa in codul de mai sus. Apoi se face click pe AddButton pentru a adauga cele doua polinoame si , in final, se face o afirmatie folosind assertEquals pentru a verifica daca rezultatul este egal cu „3.0x^2+8.0x+12.0”.

Acest caz de testare verifica corectitudinea functionalitatii de adaugare a polinoamelor interfetei si ajuta la asigurarea faptului ca implementarea functioneaza conform asteptarilor.

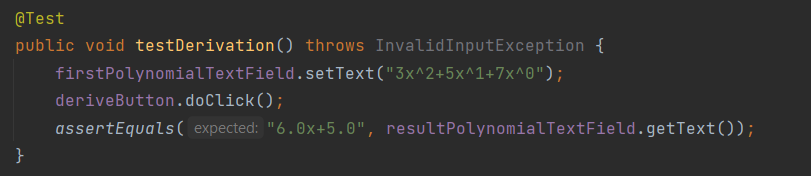
* Scadere



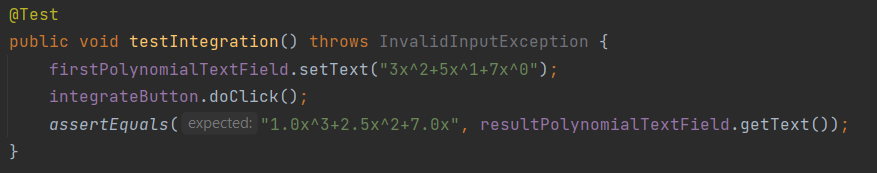
* Inmultire



* Derivare



* Integrare



In final, daca testarea functioneaza corect pentru aceste operatii, vom astepta acest rezultat :

Graphical user interface, text

Description automatically generated

# Concluzii

In concluzie, aplicatia dezvoltata respecta paradigmele programarii orientate pe obiect. Calculatorul functioneaza in conformitate cu cerinta daca utilizatorul respecta conventia de introducere a polinoamelor.

Personal, aceasta tema m-a ajutat sa imi aprofundez cunostintele de dezvoltare a unei aplicatii, sa ma familiarizez cu limbajele POO, in special cu expresiile de pattern-matching, cu care nu am mai lucrat pana la aceasta tema.

# Bibliografie

* <https://docs.oracle.com/javase/tutorial/essential/regex/>
* <https://docs.oracle.com/javase/tutorial/essential/regex/>
* <https://www.tutorialspoint.com/junit/junit_using_assertion.htm>
* <https://users.utcluj.ro/~igiosan/Resources/POO/Lab/12-Testarea_Unitara.pdf>