**DOCUMENTATIE**

**Tema 1**

**Nume student: Faur Sergiu**

**Grupa: 30224**

**Cuprins**

1.Obiectivul lucrarii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.Analiza problemei, modelare, scenarii, cazuri de utilizare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

3.Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5

**4.Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7**

[5.Rezultate](#_Toc95297889) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9

6.Concluzii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

7.Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

# Obiectivul temei

**OBIECTIVUL PRINCIPAL**

Obiectivul principal este de a implementa I de face design la un calculator de polinoame pentru care se va face si o interfata grafica pentru utilizator prin care acesta sa introduca polinoamele si dupa aceea sa selecteze operatia pe care doreste sa o efectueze si sa apara polinomul rezultat.

**OBIECTIVE SECUNDARE**

**1.Analiza problemei si identificarea cerintelor – la aceasta se doreste intelegerea problemei si cee ace trebuie implementat – se va discuta mai detaliat in capitolul 2**

**2.Designul calculatorului de polinoame – pentru acest obiectiv se doreste a se gandi cum va arata acest calculator de polinoame pentru a fi cat mai usor de interpretat de utilizator si de asemenea ce clase vor trebui sa fie implementate si care e relatia dintre acestea– se va discuta mai detaliat la capitolul 3**

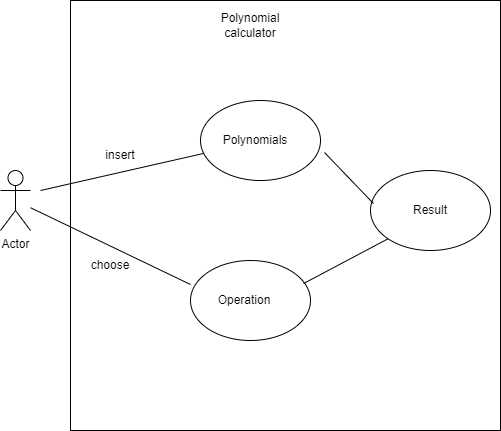
**3.Implementarea calculatorului de polinoame – pentru aceasta se va discuta structura operatiilor si implementarea interfetei grafice pentru utilizator – e va detalia la capitolul 4**

**4.Testarea calculatorului de polinoame – pentru aceasta se vor discuta cum s-au efectuat teste pentru a observa functionalitatea corecta a calculatorului – se va detalia la capitolul 5**

# 

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

**Diagrama use case:**



**Use case: a se face o operatie intre 2 polinoame**

**Actor principal : user**

**Scenariul de success :**

**1.Utilizatorul introduce cele 2 polinoame correct ( utilizatorului ii este cerut sa nu introduca alte caractere in afara de cele necesare pentru a define un polinom, aparitia acestora putand duce la un raspun eronat , de asemenea daca se introduce x singur la inceput va trebui sa se specifice + sau - x) sub forma de monoame de forma “coefx^exp” cu plus sau minus intre monoame .**

**2.Utilizatorul apasa pe butonul corespunzator operatiei dorite**

**3.Calculatorul va returna rezultatul operatiei .**

**Scenariu alternativ : Unul sau ambele polinoame ete introdus gresit , in aces caz raspunsul va fi eronat**

**Use case: a se face o operatie cu 1 polinom**

**Actor principal : user**

**Scenariul de success :**

**1.Utilizatorul introduce polinomul correct in spatial pentru primul polinom( utilizatorului ii este cerut sa nu introduca alte caractere in afara de cele necesare pentru a define un polinom, aparitia acestora putand duce la un raspun eronat ) sub forma de monoame de forma “coefx^exp” cu plus sau minus intre monoame .**

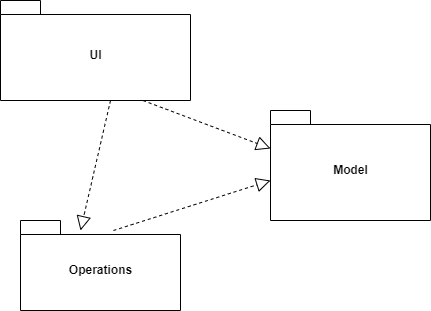
**2.Utilizatorul apasa pe butonul corespunzator operatiei dorite**

**3.Calculatorul va returna rezultatul operatiei .**

**Scenariu alternativ : Polinomul este introdus gresit , in aces caz raspunsul va fi eronat**

# 3.Proiectare

**Diagrama de pachete este :**



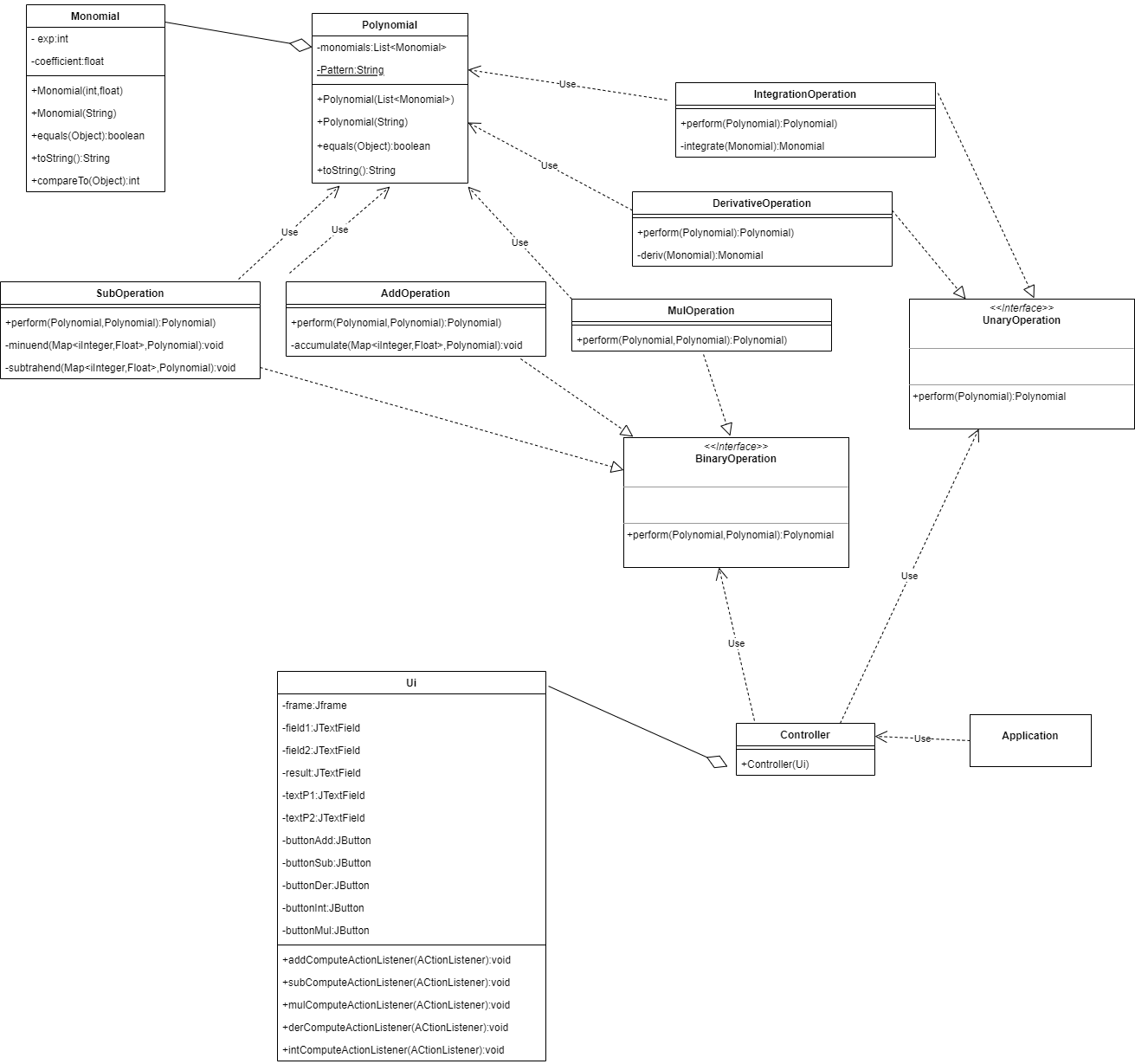
**Pachetul de Model contine clasele de Monomial si Polynomial**

**Pachetul de Operations contine clasele AddOperation , SubOperation,DerivativeOperation, IntegrationOperation, MulOperation si de asemenea si interfetele UnaryOperation si BinaryOperation . UnaryOperation este implementata de clasele DerivativeOperation si IntegrationOperation pentru ca metodele acestora necesita doar un polinom , iar interfata BinaryOperation este implementata de MulOperation, SubOperation si AddOperation pentru ca metodele necesita 2 polinoame.**

**Pachetul UI contine clasa Ui in care este construe view-ul pentru interfata grafica**

**De asemenea mai unt 2 clase numite Controller ce da functionalitate Ui-ului si Application in care se afla mainul ,unde se ruleaza aplicatia.**

**Diagrama UML este :**



**MODEL**

**Clasele din pachetul Model (Polynomial si Monomial) au rolul de a descrie obiectele Polinom si Monom .Acestea doua sunt intr-o relatie de agregare deoarece Polinomul are un atribut de tip Lista de monoame , dar un monom poate eista si fara polinomul in sine.**

**OPERATIONS**

**Clasele din pachetul Operations descriu operatiile implementate de catre Polinoame , de aceea , acestea sunt intr-o relatie de dependenta cu clasa Polynomial , acestea folosind obiecte de tipul polinom.**

**De asemenea toate operatiile implementeaza una dintre interfetele BinaryOperation au UnaryOperation , in functie de numarul de polinoame folosite pentru a efectua operatia.**

**UI**

**UI-ul reprezinta interfata utilizator prin care utilizatorul poate opera cu polinoame , aceata este intr-o relatie de agregare cu Controller deoarece Controllerul are ca parametru in contructor UI-ul.**

**De asemenea Controllerul este intr-o relatie de dependenta cu toate operatiile , deoarece acesta foloseste atat operatii de tip BinaryOperation cat si din UnaryOperation.**

**Clasa Application foloseste controllerul , aceasta ruland aplicatia.**

# 4.Implementare

Fiind un proiect OOP , se bazeaza pec lase sip e relatiile dintre acestea , in capitolul acesta vom descrie fiecare clasa cu functionalitatea sa si cee ace reprezinta .

Incepem cu clasele din pachetul MODEL :

**MONOMIAL**

Aceasta clasa este realizata deoarece fiecare polinom este format dintr-o lista de monoame , astfel aceasta sta la baza polinomului.

Atributele unui monom sunt in mod natural un coefficient si un exponent .

Au fost facuti 2 contructorii pentru aceasta clasa , primul fiind cel classic , iar al doilea avand rolul de a putea lua datele dintr-un string , deoarece utilizatorul introduce un string de la tastatura.Astfel , in contructorul cu string , se verifica mai multe cazuri de introducere a stringului , deoarece in matematica nu e scrie explicit coeficientul si exponentul in multe cazuri ( de exemplu: cand coeficientul e 1 se scrie x ,iar cand exponentul e 0 se scrie doar coeficientul).

Celelalte metode sunt doar niste suprascrieri la metode predefinite de java pentru obiecte. Equals verifica daca doua monoame sunt egale , returnand un Boolean , iar compareTo este folosita pentru a putea sorta lista de monoame din polinom.Metoda toString este suprascrisa pentru a afisa monoamele in mod natural ,mathematic .

**POLYNOMIAL**

Aceasta clasa are ca attribut o lista de monoame , ce defineste in mod natural un polinom.

De asemenea are un pattern reprezentand o expresie regex pentru a putea recunoaste un string ca un polinom.

Tot din aceasta cauza sunt facuti si 2 constructori , ca in cazul monomului.

Analog ca la monom , toate metodele au aceeasi functionalitate , metodele din monomial trebuind fiind implementate deoarece un polinom e compus din monoame.

Mai departe , pentru elementele din pachetul OPERATIONS , vom incepe cu interfetele , deoarece toate operatiile implementeaza una dintre interfete , in esenta reprezentand una dintre cele tipuri de operatii :BinaryOperation si UnaryOperation.

**UnaryOperation**

Aceasta interfata contine o singura metoda numita perform ce are ca paramtru un polinom .In esenta , reprezinta toate operatiile ce se efectueaza pe un singur polinom.

**BinaryOperation**

Ca si mai sus , aceasta interfata contine doar o metoda numita perform , dar de aceasta data cu 2 parametrii de tip polinom , reprezentand operatiile ce sunt effectuate intre 2 polinoame.

In continuare vom prezenta operatiile in sine:

**AddOperation**

In aceasta clasa se suprascrie metoda de perform din BinaryOperation ce in acest caz va face adunarea a doua polinoame .Pentru a efectua mai usor adunarea , clasa mai contine o metoda privata accumulate unde se pune intr-un hashmap pe rand ambele polinoame .Cand se pune al doilea , se va schimba coeficientul(valoarea din hashmap) cu suma dintre coeficentii monoamelor cu exponent la fel(exponentul reprezinta cheia din hashmap).La final functia perform va returna polinomul rezultat din adunarea celor 2.

**SubOperation**

Ca si adunarea , scaderea este o operatie de tip BinaryOperation , ce suprascrie metoda de perform ce va returna polinomul rezultat din scaderea celor doua . In loc de metoda accumulate , aici am folosit 2 metode private pentru scazator si descazut , pentru a pune in polinomul rezultat semnul – unde trebuie.

**MulOperation**

In aceasta operatie tot de tip BinaryOperation se va afla doar suprascrierea metodei perform, deoarece am facut parcurgerea hashmapului in ea si schimbam coeficientul doar in locurile unde suma exponentilor era deja in map.

**DerivativeOperation**

Aceasta descrie o operatie unara , efectundu-se pentru un singur polinom . Aceasta va suprascrie metoda din interfata UnaryOperation numita perform in care am apelat o metoda de ajutor numita deriv ce face derivarea pentru un singur monom, urmand ca in perform sa apelez metoda deriv pentru fiecare monom din lista.

**IntegrationOperation**

Aceasta este similara cu derivarea , singura diferenta fiind metoda auxiliara ce face integrarea pentru un ingur monom.

Am descries modelul proiectului, urmand sa descriu Ui-ul si controllerul.

**UI**

In aceasta s-a creat fereastra de interfata utilizator prin care se pot efectua si vizualiza operatiile pe polinoame.

Ui-ul contine 2 textField-uri pentru introducerea polinoamelor de catre utilizator , unul pentru rezultat si inca 2 in care se specifica care este primul polinom si al doilea , acest lucru fiind important deoarece operatiile unare e vor efectua pentru polinomul 1.

Fereastra mai contine si 5 butoane reprezentativ pentru fiecare operatie , functionalitatea acestora fiind descrisa de adaugarea unor ActionListeneri pentru fiecare , acestea reprezentand metodele .



+ =adunare

* = Scadere

X =inmultire

Deriv = derivare

I =integrare

**Controller**

**In aceasta clasa se da functionalitate interfetei grafice , prin implementarea metodelor de actionPerformed pentru ActionListenerii fiecarui buton .**

**Application**

**Aceasta clasa este pentru rularea aplicatiei prin definirea unui controller cu un ui.**

# 5.Rezultate

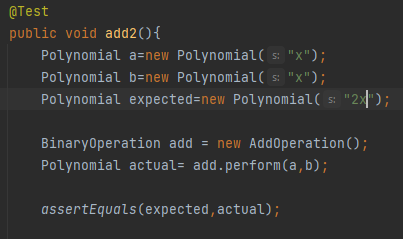
**Pentru testare am folosit JUnit si am facut cate o clasa de test pentru fiecare operatie.**

**Astfel s-au generat clasele AddOperationTest, SubOperationTest,MulOperationTest,DerivativeOperationTest si IntegrationOperationTest.**

**In fiecare am verificat aceleasi scenario si anume toate cazurile cand coeficientul sau exponentul nu este specificat direct prin numar de exemplu cand se va scrie x sau o constanta , exponentul se va pune pe 1 respectiv 0 ,iar coeficientul pe 1 respectiv constanta.**

**In fiecare metoda de testare am creat 2 polinoame a si b ,un polinom expected cu valoarea teoretica ce trebuia sa iasa si un polinom actual cu rezultatul operatiei aferente , urmand sa compar cele 2 .**

**Un exemplu de cod pt un test este:**



# 6.Concluzii

**Acest proiect a fost foarte bun pentru invatarea lucrului in OOP , prin incercarea de a respecta cat mai mult principiile OOP.**

**Niste dezvoltari ulterioare ar fi:**

* **Modernizarea interfetei grafice**
* **Adaugarea unor noi operatii pentru polinoame precum aflarea radacinilor**
* **Implementarea functionalitatii pentru coeficienti reali**

# 7.Bibliografie

<https://app.diagrams.net>

<https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf>

<https://www.geeksforgeeks.org/>