Precup Horea

Grupa 7

**DOCUMENTATIE**

Tema 1

- Calculator polinomial-

1. **Obiectivul temei**

Obiectivul principal este realizarea unui calculator polinomial care sa execute operatii cum ar fi: adunare, scadere, inmultire, impartire, derivare si integrare a polinoamelor.

Obiectivele secundare (pasii de urmat):

- realizarea unei interfete grafice cu doua campuri pentru introducerea polinoamelor, un camp pentru afisarea rezultatelor si butoane care efectueaza operatiile la apasare

- crearea de clase pentru interfata, pentru datele cu care lucram (respective monom si polinom), pentru operatiile care trebuie executate si nu in ultimul rand pentru teste

- dupa introducerea polinomului, respectiv a polinoamelor, trebuie efectuata conversia corecta de la tipul de date “String” la clasele noastre monom si polinom pentru a putea utiliza datele furnizate in calcule

- implementarea corecta a operatiilor, care returneaza rezultatul si acoperirea atat a cazurilor generale, cat si a unor cazuri mai putin intalnite sau particulare

- conversia rezultatului de la tipul cu care lucram (polinom) la tipul String, pentru a-l putea afisa in campul de rezultat din interfata grafica

- implementarea si executia de teste cu ajutorul Junit

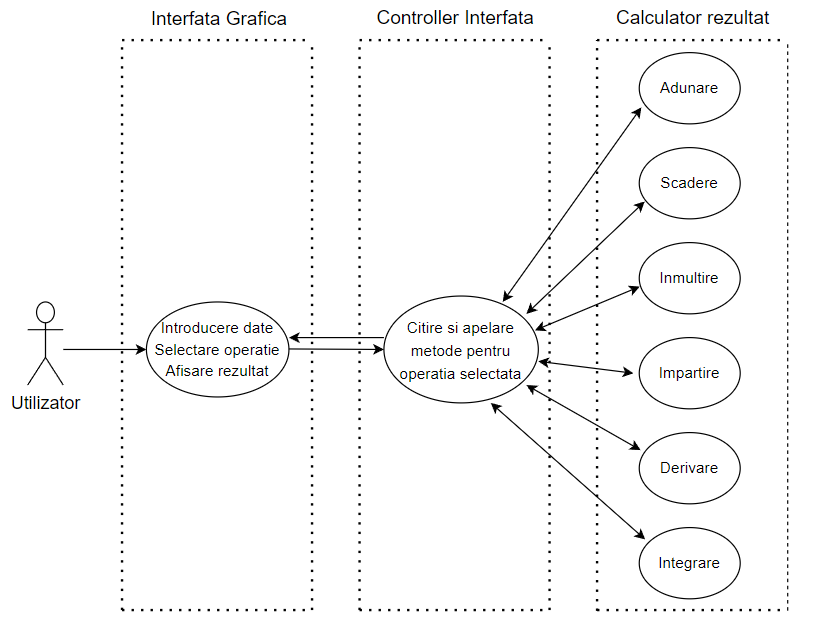
1. **Analiza problemei**

Cerintele functionale descriu interactiunile dintre sistem si mediul acestuia.

Aplicatia noastra este un calculator polinomial care afiseaza rezultatul operatiei intre doua polinoame cu ajutorul unei interfete grafice. Calculatorul afiseaza rezultatul la apasarea unui buton de catre utilizator dupa ce acesta a introdus datele de intrare (polinoamele cu care vrea sa execute operatii).

Utilizatorul este nevoit sa introduca cele doua polinoame intr-un format corect, in functie de variabila “x”, iar apoi sa apese un buton corespunzator operatiei pe care doreste sa o execute. Calculatorul va afisa rezultatul final in campul “rezultat” din interfata grafica a programului.

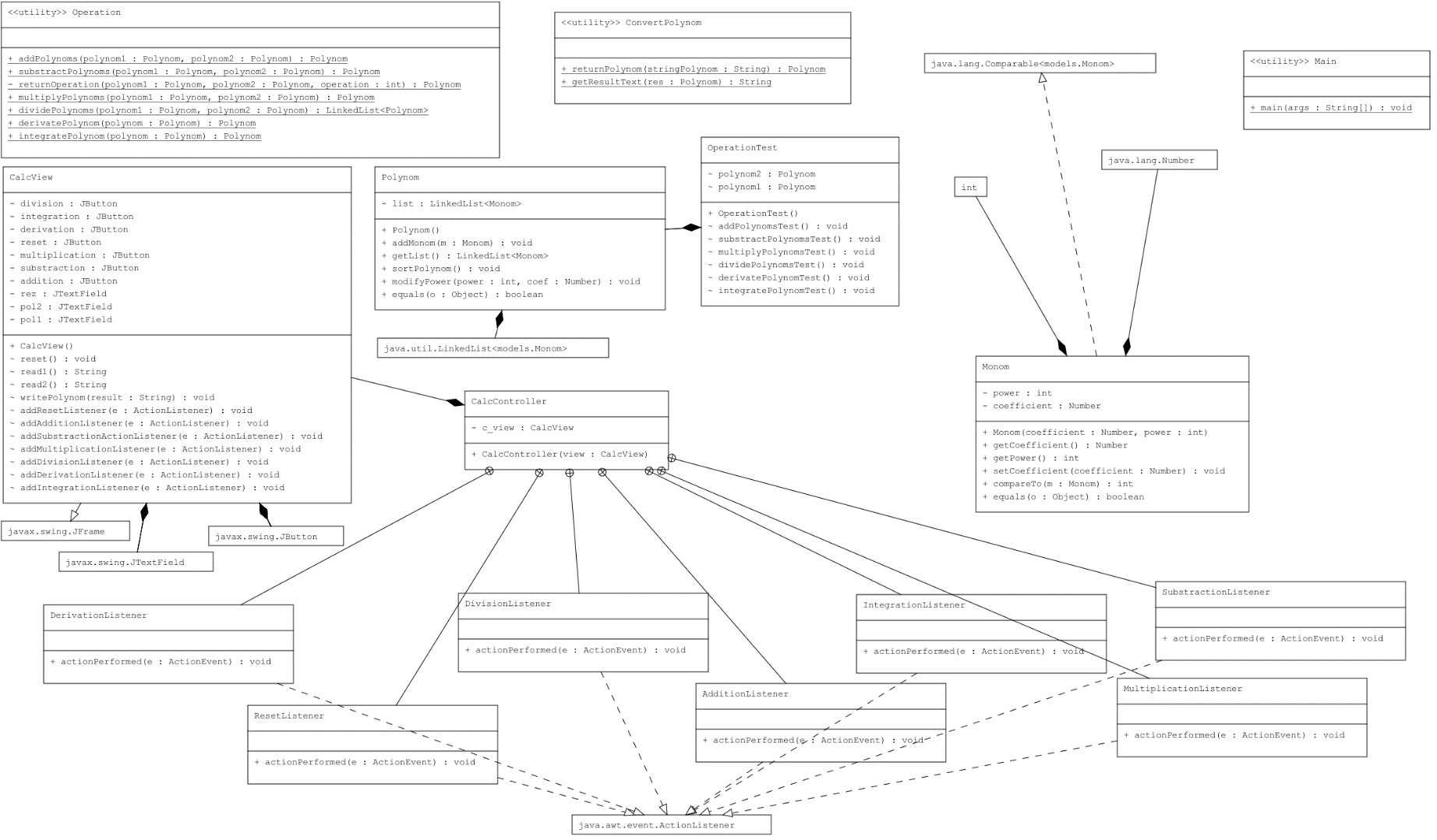
Flow-chart:



1. **Proiectare**

Aplicatia este impartita in pachete, in felul urmator:

* Pachetul “gui” -> reprezinta interfata grafica
  + CalcView : prezinta interfata vazuta de utilizator (fereastra cu butoane si campuri) si are rol de afisare
  + CalcController : face legatura intre interfata grafica si metodele care nu sunt vazute de utilizator si “leaga” componentele din interfata de functionalitatile care trebuie indeplinite
* Pachetul “models” -> reprezinta tipurile de date folosite, monom si polynom
  + Monom : data de tipul “coeficient\*x^putere”
  + Polynom : presupune un LinkedList de monoame
* Pachetul “operations”
  + ConvertPolynom : transforma datele de intrare, de tipul String, in date de tip Polynom si datele rezultat de tipul Polynom in date String pentru afisare
  + Operation : clasa care contine toate metodele necesare efectuarii operatiilor dintre polinoame
* Pachetul “testing\_main\_program”
  + OperationTest : clasa de teste JUnit
  + Main : clasa main de unde pornim programul

Diagrama UML:

1. **Implementare**

Clasele si metodele corespunzatoare :

* CalcView

- JTextField (x3) : pentru cele 2 polinoame si rezultat

- JButton (x7) : pentru operatii si reset(pune “0” la cele 2 polinoame)

Constructor : se creeaza JPanels pe care adaugam butoanele si zonele de text

Metode pentru citirea polinoamelor, scrierea rezultatului si pentru adaugarea de ActionListeners la butoane.

* CalcController

- contine obiectul de tip CalcView

Contine clasele specifice pentru fiecare ActionListener a butoanelor. Fiecare clasa incepe cu citirea polinoamelor din interfata, transformarea acestora din tipul String in tipul Polynom, apoi se apeleaza functia specifica operatiei. Rezultatul obtinut se scrie in interfata.

* Monom

- Number coefficient; int power

S-au implementat doar metode de tip getter, setter si s-au suprascris metodele *compareTo* si *equals* pentru sortarea si compararea polinoamelor

* Polynom

- LinkedList<Monom>

Metode de tip getter, setter, suprascrierea metodei *equals*.

*-void modifyPower(int power, Number coef)*: metoda folosita la inmultirea polinoamelor, cu ajutorul careia actualizam informatia din polinomul nostrum (modificam coeficientul lui x^putere sau daca nu exista aceasta putere a lui x, adaugam monomul in lista noastra)

* ConvertPolynom

- nu are atribute

Este alcatuita doar din metode statice.

*-static Polynom returnPolynom(String stringPolynom)*: transformam polinomul citit sub forma de String in obiect de tipul Polynom

*-static String getResultText(Polynom res)*: transformam obiectul rezultat Polynom sub forma de String pentru afisare

* Operation

- nu are atribute

Este alcatuita doar din metode statice.

*-static Polynom addPolynoms(Polynom polynom1, Polynom polynom2)*: returneaza suma dintre cele doua polinoame

*-static Polynom substractPolynoms(Polynom polynom1, Polynom polynom2)*: returneaza diferenta dintre cele doua polinoame

*-static Polynom returnOperation(Polynom polynom1, Polynom polynom2, int operation)*: metoda apelata de cele de mai sus, pentru ca sa nu scriem aproape acelasi cod de 2 ori, deoarece algoritmii de adunare si scadere sunt foarte asemanatori; operatia care se executa se decide in functie de valoarea parametrului “operation”

*-static Polynom multiplyPolynoms(Polynom polynom1, Polynom polynom2)*: returneaza produsul polinoamelor, iar pentru fiecare produs de monoame, se apeleaza metoda implementata din clasa Polynom, pentru a verifica daca este nevoie sa adaugam un nou monom sau doar sa schimbam un coefficient

*-static LinkedList<Polynom> dividePolynoms(Polynom polynom1, Polynom polynom2)*: returneaza o lista cu 2 elemente (cat si rest) daca avem catul diferit de 0, o lista cu un element (rest) daca avem catul 0 si o lista fara elemente daca numitorul este 0.

*-static Polynom derivatePolynom(Polynom polynom)*: returneaza derivata polinomului

-*static Polynom integratePolynom(Polynom polynom)*: returneaza integrata polinomului cu coeficienti reali

* OperationTest

- Polynom polynom1, polynom2

Contine toate metodele de test in care se verifica corectitudinea metodelor din clasa Operation.

* Main

- clasa de unde ia nastere programul

1. **Rezultate**

Pentru cazul de testare cu JUnit am luat polinoamele astfel:

- polinom 1: x^3-12x^2+38x-17

- polinom 2: x-7

Toate operatiile au fost executate correct.

1. **Concluzii**

Pentru realizarea proiectului au necesitat multe modificari, mai ales in cazul impartirii problemei pe clase. Initial, proiectarea era diferita, insa cu adaugarea treptata a mai multor metode/functionalitati, apareau unele erori de proiectare si a necesitat schimbarea modului de rezolvare.

In acest proiect am avut ocazia sa aprofundez conceptele OOP si sa imi fac o imagine mai buna asupra unui proiect mai complex, in care avem nevoie de o interfata care comunica cu utilizatorul si un cod bine structurat in spatele acesteia.