**Tehnici de programare**

**Tema 1:**

**Procesarea polinoamelor de o singura variabila**

Student: Pop Tudor Andrei

Grupa:20229

Data: 21.03.2019

Cuprins:

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie
8. Obiectivul temei

***Cerinta:***Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

Un prim obiectiv pentru indeplinirea acestei cerinte este de a descompune “problema” polinoamelor si de a implementa operatiile pe o structura mai simpla, monoame. Un polinom nu este altceva decat o suma a mai multor monoame prin urmare operatiile asupra acestor componente individual stau la baza operatiilor pe o scala mai larga, a polinoamelor.

Interactiunea cu utilizatorul se va face prin intermediul unei interfete grafice (GUI) care va consta in 2 text field-uri unul pentru primul polinom, iar al doilea pentru un alt polinom. Utilizatorul va mai avea la dispozitie cateva butoane pentru a selecta operatiile dorite ( fie pe polinom individual, fie pe cele doua input-uri). Rezultatele vor fi afisate in 3 text-fielduri ce nu pot fi modificate de utilizator, avand rol strict de vizualizare alea acestora. Mai multe detalii se vor afla in sectiunea implementarii aplicatiei.

***Informatii generale despre polinoame:*** În [matematică](https://ro.wikipedia.org/wiki/Matematic%C4%83), un polinom este o [expresie](https://ro.wikipedia.org/w/index.php?title=Expresie_matematic%C4%83&action=edit&redlink=1) construită dintr-una sau mai multe [variabile](https://ro.wikipedia.org/wiki/Variabil%C4%83) și [constante](https://ro.wikipedia.org/wiki/Constant%C4%83_matematic%C4%83), folosind doar operați de [adunare](https://ro.wikipedia.org/wiki/Adunare), [scădere](https://ro.wikipedia.org/wiki/Sc%C4%83dere), [înmulțire](https://ro.wikipedia.org/wiki/%C3%8Enmul%C8%9Bire_(matematic%C4%83)) și ridicare la putere constantă pozitivă întreagă. ( 3x^5 – x +5 ) {\displaystyle x^{2}-4x+7\,}  este un polinom. Se observă în particular că împărțirea printr-o expresie ce conține o variabilă nu este permisă în polinoame.

Polinoamele sunt construite din [termeni](https://ro.wikipedia.org/w/index.php?title=Termen_(matematic%C4%83)&action=edit&redlink=1) numiți [monoame](https://ro.wikipedia.org/w/index.php?title=Monom&action=edit&redlink=1), care sunt alcătuite dintr-o constantă (numită [coeficient](https://ro.wikipedia.org/wiki/Coeficient)) înmulțită cu una sau mai multe [variabile](https://ro.wikipedia.org/wiki/Variabil%C4%83). Fiecare variabilă poate avea un [exponent](https://ro.wikipedia.org/w/index.php?title=Exponent&action=edit&redlink=1) constant întreg pozitiv exemplu monom: 3x^7 . O constanta se considera un monom deoarece se poate scrie sub urmatoarea forma: cX^0, unde c este o constanta/coefficient.

***Proproprietati folosite ale polinoamelor:*** Gradul unui polinom consta in valoarea cea mai mare a exponentului unui monom care are coeficientul diferit de 0 exemplu: x^2 + x + 1 are gradul 2, dar 0x^200 + x + 1 are gradul 1. Aceasta proprietate este foarte importanta pentru operatia de impartire.

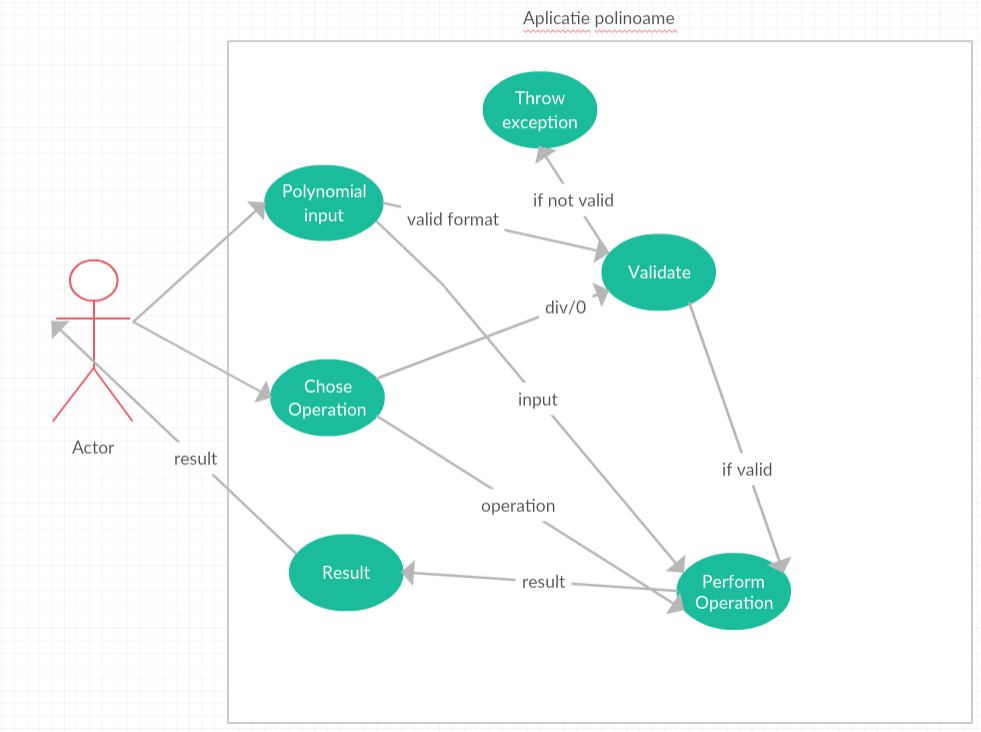
Doua polinoame sunt egale daca si numai daca coeficientii monoamelor cu acelasi exponent sunt egali

Forma algebrica a unui polinom este: P(x) = anX^en + (an-1)X^en-1 + . . . + a0X^e0 unde an, an-1 . . . a0 sunt coeficientii (!= 0), iar en, en-1, . . .e0 sunt exponentii polinomului.

***Operatiile implementate pe polinome:*** Adunarea: se vor aduna toti coeficientii terminilor cu acelasi exponent. Scaderea: se vor scadea toti coeficientii terminilor celui de-al doilea polinom din primul polinom pentru termenii cu acelasi exponent acelasi exponent. Inmultirea: se va inmultii fiecare termen (monom) din primul polinom cu fiecare terment din al doilea polinom, dupa care se vor elimina termenii asemenea rezultand astfel un nou polinom. Impartirea: in aplicatie impartirea se realizeaza conform metodei “long division”. Se inmulteste impartitorul cu un termen astfel incat la adunarea rezultatului impartirii cu deimpertitul sa se elimine termenul de grad maxim. Se va repeta acest proces pana cand in deimpartit ramane un polinom cu un grad mai mic decat impartitorul. Suma termenilor cu care s-a inmultit impartitorul reprezinta “Catul”, iar ce ramane la finalul algoritmului reprezinta “Restul”. Legatura dintre acesti termini este: C \* I + R = D, grad(R) <= grad (i). Derivarea: se va inmulti coeficentul fiecatui termen cu expenentul acestuia, iar exponentul se decrementeaza; derivarea unei constant este 0 exemplu: (2x^2 + 1)’ = 4x. Integrarea: se va imparti coeficientul fiecarui termen cu valoarea (coeficentului + 1), iar coefientul se va incrementa cu 1 exemplu: integr(2x^2 + 1) = 0.66x^3 + x.

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

UML Case Diagram:



Actorul – reprezinta utilizatorul, acesta introduce datele de intrare si alege ce operatii sa efectueze asupra acestora

Aplicatie polinoame – reprezinta intreaga aplicatie, ofera utilizatorului posibilitatea de a introduce date si de a efectua calcule asupra acestora, de asemenea comunica utlizatorului rezultatele, sau semnaleaza daca este vreo problema

Polynomial input – reprezinta campurile pe care utilizatorul poate introduce date de intrare

Chose Operation – reprezinta operatia pe care utilizatorul decide sa o faca asupra datelor introduce

Validate – valideaza datele de intrare. Pentru Polynomial input verifica ca datele de intrare sa corespunda formatului pe care aplicatia il poate parsa si ai apoi prelucra. Pentru Chose Operation va opri impartirea la 0 in cazul in care utilizatorul alege sa faca acest lucru

Throw Exception – va raspunde la o problema de validare a datelor si va oferi utilizatorului informatii referitoare la acestea, de exemplu: “format invalid”, sau “division by zero”

Perform operation – va prelua datele de la Polynomial input si Chose Operation si va executa operatia corespunzatoare cu datele introduce de utilizator

Result – preia rezultatele de la Perform operation si il va afisa utilizatorului.

Un posibil mod de utilizare:

1. Se introduce primul polinom
2. Se introduce al doilea polinom
3. Se cere derivata ambelor polinoame
4. Se cere integrala ambelor polinoame
5. Se cere suma celor doua polinoame
6. Se cere diferenta celor doua polinoame
7. Se cere produsul celor doua polinoame
8. Se cere impartirea celor doua polinoame

Daca oricare din cele polinoame este introdus gresit, sau cu un format gresit, aplicatia va semnala o problema utilizatorului si nu va executa nimic pana cand ambele inputuri sunt valide

Daca al doilea polinom este 0 si se doreste impatirea, programul va semnala problema utilizatorului printr-un mesaj;

Aceste semnalari de vor fi scrise in partea de sus a interfetei.

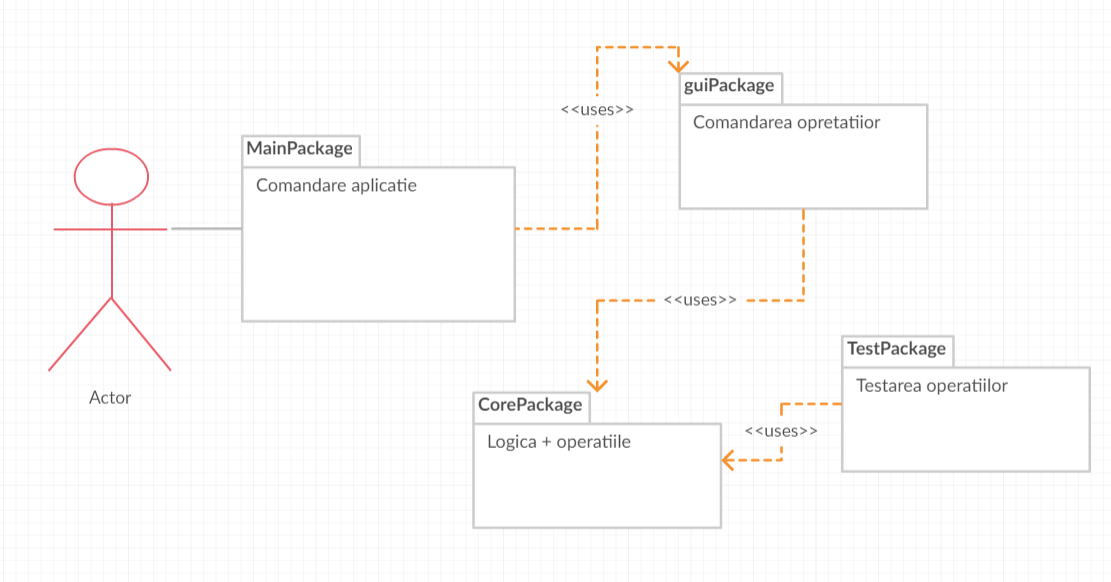
Cand nu se va mai dori utilizarea programului se va apasa butonu [X] , iar acesta se va inchide.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

Pentru proiectare am ales sa implementez o clasa Monom si o clasa Polinom pentru a crea o implementare OOP. Clasa Monom are implementate toate metodele care corespund operatiilor pe polinom la un nivel mai simplu, de asemenea aceasta clasa are un rol essential in validarea datelor de intrare. Aceasta clasa suprascrie si metoda toString() din clasa Object care ajuta enorm de la afisare corespunzatoare a datelor.

Clasa Polinom are in interiorul acestetia un obiect de tipul ArrayList<Monom> in care stocheaza monoamele din care este format. La ranul ei, clasa implementeaza metodele corespunzatoare operatiilor cerute prin apelui eficiente la clasa Monom, respectand astfel paradigma Object Oriented Programing. De asemenea clasa suprascrie metoda toString() din clasa Object care afisaza in mod corespunzator polinomul intr-un format cunoscut si usor de inteles.

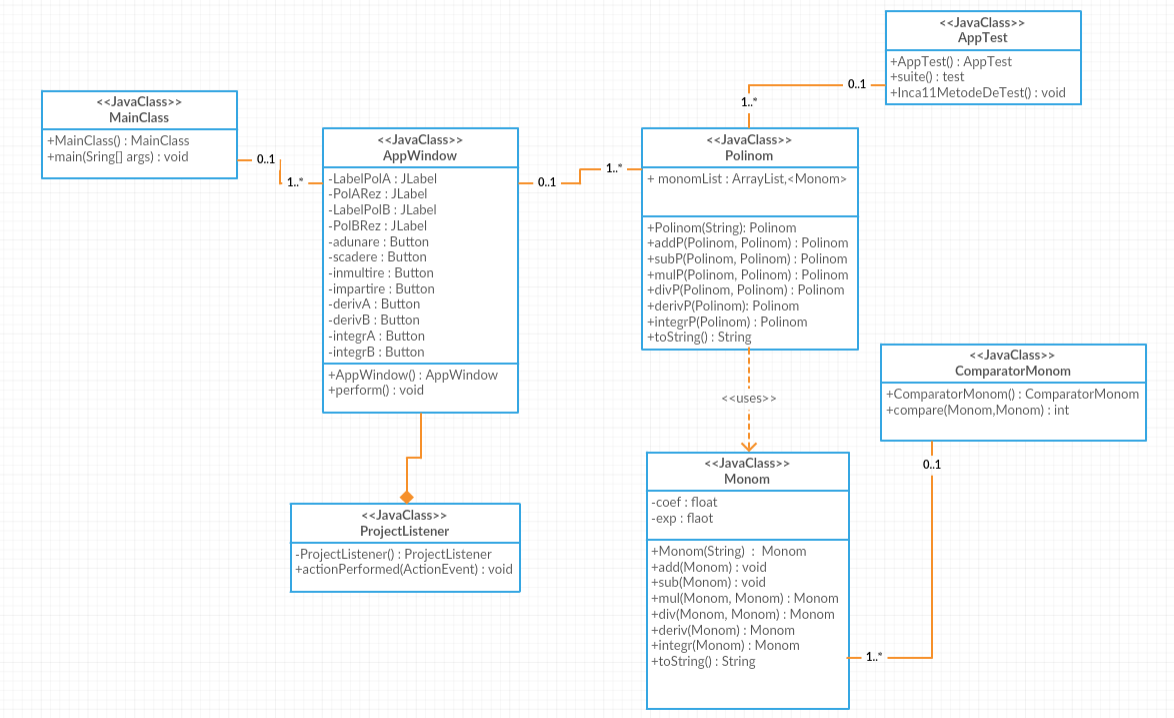
**Package Diagram:**



Dupa cum se poate observa din diagrama pachetelor utilizatorul interactioneaza direct doar cu casa pachetul MainPackage care la randul sau formeaza un lant care ajunge pana la “core” unde se prelucreaza datele.

Pachetul TestPackage interactioneaza strict doar cu CorePackage pentru a evalua corectitudinea metodelor si a logicii in general;

**Class Diagram:**

****

4. Implementare

Clasa MainClass : in interiorul aceste clasa se afla metoda “public static void main(String[] args)” in interiorul acesteia declar un obiect din clasa “AppWindow” si apelez cu acel obiect metoda “perform()”.

Clasa AppWindow : in interiorul acestei clasa construiesc efectiv interfata grafica si implementez clasa “ProjectListener”. Constructorul acestei clase da forma aplicatiei, un obiect ”JFram” care contine restul obiectelor intr-un “GridLayout (5,1)”: 1 label pentru afisarea exceptiilor, 2 panouri panouri pentru cele doua polinoame (2 rexfieduri si 2 butoane), un panou cu 4 butoane pentru operatiile pe cele 2 butoane, iar in final un panou cu un textfield pentru afisarea rezultatelor operatiilor de mai sus. Tot in constructor de seteaza ActionCommand pentru fiecare buton. De asemenea in aceasta clasa se afla metoda “perform()” care asociaza un listener fiecarul camp al acestei clase.

Clasa ProjectListener: se afla in interiorul clasei descrise anterior si contine metoda “actionPerformed()” care in functie de tipul ActionCommand – ului va comanda ce operatii sa se execute. De asemenea aceasta clasa vat rata exceptiile care s-ar putea sa apara si afisaza un mesaj in primul label al interfetei grafice.

Clasa **Monom** : este o clasa absolut esentiala acestui proiect.

Una din cele mai importante metode care apar in aceasta clasa este contructorul “public Monom(String str)” aceasta face descompunerea unui String folosind String.split( , ). Desparte in functie de x Stringul primit ca parametru si il functie de diferite posibilitati care metoda split le genereaza va adduce o valoare corespunzatoare pentru coeficientul si exponentul monomului, sau in va arunca NumberFormatException daca formatul este invalid. Acest constructor este supraincarcat de inca 2 metode “public Monom(Monom x)”, “public Monom(float coef, float exp)” care sunt folosite in metodele ce urmeaza sa fie explicate

Metodele add() si sub()

public void add(Monom b) {//Aduna, daca se poate, b-ul la a

if(this.getExp() == b.getExp()) {

this.coef = this.getCoef() + b.getCoef();

}

}

public void sub(Monom b) {//Scade, daca se poate, b-ul din a

if(this.getExp() == b.getExp()) {

this.coef = this.getCoef() - b.getCoef();

}

}

Aceste 2 metode sunt extrem de similar, va aduna coeficientul Monomului b la coeficientul lui a daca acestia au acelasi exponent.

Metoda mul:”public static Monom mul(Monom a, Monom b)”va efectua inmultirea celor doua polinoame, inmultind coeficientii si adunand exponentii, valori cere vor fi ulterior folosite pentru crearea unui nou polinom

Similar metoda div: “public static Monom div(Monom a, Monom b)” similar, va impartii coeficientii si va scadea exponentii. In plus daca s-ar incerca o impartire la 0 se va arunca “ArithmeticException”

Metoda derv: “public static Monom derivare(Monom a)” va returna un nou monom cu valorile urmatoare: coeficientul current inmultit cu exponentul current, iar exponentul va si decrementat cu 1

Metoda integr: “public static Monom integrare(Monom a)” va returna un nou monom cu valorile: coefcientul current impartit la exponentul current incrementat cu 1, iar exponentului i se va incrementa valoarea cu 1

Metoda toString: va returna un string in functie de coeficientii monomului care respecta conventiile: (1 nu apare la coefficient, 0 nu apare le exponent, etc.)

Clasa **Polinom** : reprezinta obiectivul principal al acestui proiect. Din definitie un polinom este o suma de monoame, prin urmare in aceasta clasa se afla un ArrayList<Monom> .

Una din metodele relevante care apar in aceasta clasa: Polinom(String str), acest constructor va elimina “white space” –urile din str, si va formata totul la lower case pentru a mari flexibilitatea formatului inputului. Constructorul foloseste metoda String.split(,) pentru a despartii monoamele intre ele, ale crea si ale introduce in lista.

Metodele public static Polinom addP(Polinom p1, Polinom p2) si public static Polinom addP(Polinom p1, Polinom p2) efectueaza in mod foarte similar operatiile de adunare/scadere a celor doua polinoame. Doua bucle “for” pentru a parcurge fiecare monom in parte care vor apela la randul lor metoda corespunzatoare operatiei pentru a obitinerea rezultatului.

Metoda public static Polinom mulP(Polinom p1, Polinom p2) va inmulti fiecare monom cu fiecare monom din ambele Polinoame intr-un ArrayList<Monom> auxiliar asupra caruia se va aplica un mod de a elimina termenii asemnea si a rezulta polinomul final. Inaintea returnarii obiectului se va sorat noul arraylist pentru a asigura o ordine prezentabila a componentelor acestuia.

Metoda public static ArrayList<Polinom> divP(Polinom p1, Polinom p2) va impartii polinomul p1 la polinomul p2 folosint metoda “LongDivision” prezentata la inceputul decumentatiei. Aceasta va returna un ArrayList care va cotine catul si restul impartirii. In plus aceasta metoda se foloseste ce ce existent in clasa Monom pentru a genera exceptia in cazul impartirii la 0.

Metodele public static Polinom derivP(Polinom p) si public static Polinom integrP(Polinom p) foarte simplu for apela metodele clasei Monom pentru fiecare component o singura data similar adunarii si scaderii.

Clasa ComparatorMonom: este folosita pentru a putea sorta Monoamele dintr-un polinom oarecare, mai apare folosita de-a lungul proiectului in operatiile esentiale,nu doar la inmultire.

Metoda implementata de aceasta clasa este “compare(Monom a, Monom b)” va asigura o sortare in oridine descrescatoare in functie de coeficientul monoamelor.

5. Rezultate

Pentru testarea si verificarea metodelor am folosit Junit3.8.1 si functiile assertEquals , assertTrue.

Metodele pentru testare : “public void testFormat1()”, avem 4 astsfel de metode care testeaza abilitatea aplicatiei de a detecta formatele incorecte si corectitudinea formatelor admise.

Metodele “public void testMul()” testeaza corectitudinea metotei “mulP()” o data pe un exemplu simplu, si intr-o alta metoda cu nume similar pe un exemplu putin mai complex

Metodele “public void testDiv()” si “public void testDiv2()”: prima metode de test va esua de fiecare data, deoarece metoda de impartire introduce multe componente considerate nulle care perturbeaza abilitatea metodei equals fie a clasei Polinom si chiar a clase String de a verifica corectitudinea metodei (totusi testele commentate in main + google sunt mai mult decat suficiente). A doua metoda verifica abilitatea acesteia de a trata impartirea la 0. Success .

Metodele “public void testDeriv()” si “public void testIntegr()” verifca aceste operatii simple pe polinoame.

Toate testele au rulat cu success, mai putin “public void testDiv()” din motivele mentionate anterior.

6. Concluzii

- documentatia s-a putut scrie in ultima zi

- proiectul se poate inbunatatii la fiecare pas: optimizare, design, implementarea operatiilor pe mai mult de 2 polinoame

- testul de la operatia de impartire trebuie revizuit

-am invatat multe despre cum functioneaza functia “String.split()” si clasa “DecimalFormat”

- descoperirea si corectarea bug-urilor care apar inevitabil

- folosire Maven pattern matching in loc de functia split();

7. Bibliografie

1) <https://en.wikipedia.org/wiki/Polynomial_long_divisio> -- pentru impartire

2) <https://creately.com/diagram-type/use-case> -- pentru diagrame

3) <https://stackoverflow.com/> -- pentru restul problemelor