**Proiectarea si implementarea unui sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi**

Cuprins:

* Ce este un polinom ?
* Obiectivul temei
* Analiza problemei
* Diagrama UML a claselor
* Implementare
* Rezultate
* Concluzii
* Bibliografie

**Ce este un polinom ?**

Polinoamele sunt construite din termeni numiți **monoame**, care sunt alcătuite dintr-o constantă (numită **coeficient**) înmulțită cu una sau mai multe variabile. Fiecare variabilă poate avea un exponent constant întreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu **gradul** acelei variabile în acel monom. Pentru că x=x^1{\displaystyle x=x^{1}}, gradul unei variabile fără exponent este unu. Un monom fără variabile se numește *monom constant*, sau doar *constantă*.

**Obiectivul temei**

Obiectivul propus in rezolvarea acestei teme este de a realiza un program care preia prin intermediul unei interfete user friendly doua polinoame si efectueaza anumite operatii asupra acestor polinoame. Operatiile realizate sunt de doua feluri: operatii care folosesc ambele polinoame sau operatii care se refera doar la un singur polinom. Spre exemplu, adunarea, scaderea, inmultirea si impartirea sunt operatii care se efectueaza intre cele doua polinoame, iar derivarea si integrarea sunt operatii care se refera doar la un polinom.

Acest program nu este foarte eficient, ci mai degraba user friendly. Este mai mult un program care m-a ajutat sa inteleg mai bine conceptele programarii orientate pe obiect si lucrul in Java.

Interfata grafica dispune de 6 butoane (Adunare, Scadere, Inmultire, Impartire, Derivare) si de 5 text field-uri. Primele doua text field-uri sunt pentru introducerea polinoamelor, cel de-al 3-lea numit Rezultat afiseaza rezultatul adunarii, scaderii, inmultirii si impartirii polinoamelor, al 4-lea camp afiseaza rezultatul derivarii primului polinom iar ultimul camp afiseaza rezultatul integrarii ultimului polinom.

**Analiza problemei**

Pentru realizarea aplicatiei am folosit 12 clase. Aceste clase se pot imparti in 2 categorii: clase folosite pentru interfata grafica si clase folosite pentru implementarea operatiilor dintre polinoame. Clasele folosite la interfata sunt : Interfata, Adunare, Scadere, Inmultire, Impartire, Derivare, cele pentru operatii sunt: Monom, Polinom, Operatie, Tokenizer iar clasa main se numeste App.

Pentru parsarea polinoamelor am folosit metoda de StringTokenizer. Acest lucru inseamna ca avem de respectat niste reguli in momentul in care dorim sa introducem un polinom si acesta sa fie interpretat correct de StringTokenizer.

Reguli de introducere a polinoamelor:

1. Nu putem introduce un polinom cu coeficientul 0
2. Intodeauna trebuie sa scriem gradul polinomului, indiferent daca acesta este 0 sau 1
3. Polinoamele pot fi introduse in orice ordine, nu este necesar sa introducem prima data monoamele cu gradul cel mai mare. M-am ocupat de acest aspect sortand monoamele din polinom in ordinea descrescatoare a gradelor acestora
4. Intre monoame trebuie pus un spatiu si abia apoi punem semnul urmatorului monom, coeficientul si gradul acestuia
5. Coeficientul poate fi si numar real pe cand gradul monomului nu poate fi decat intreg
6. Coeficientul si gradul monomului sunt separate de “x^”

Exemple de introducere a polinoamelor:

* 3x^3 +2x^4 -5x^2 +7x^1 -9x^0
* 2x^1 -3x^0

In momentul in care dorim sa afisam aceste doua polinoame, ele vor arata astfel:

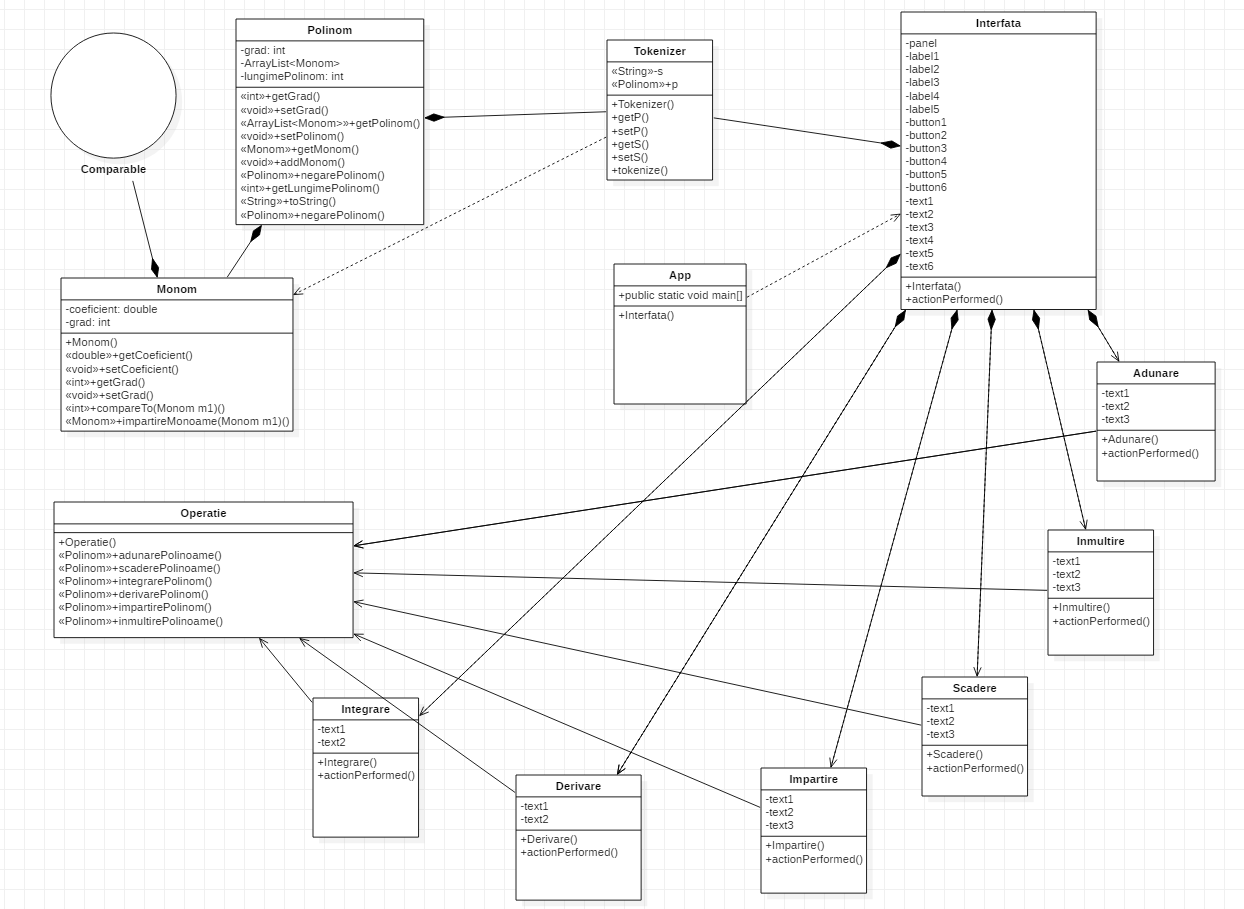
* 3.0x^3 +2.0x^4 -5.0x^2 +7.0x -9.0
* 2x -3

In momentul in care am suprascris functia toString() folosita pentru afisarea polinoamelor am tinut cont de toate aspectele necesare pentru ca afisarea polinomului sa fie facuta cat mai profesionist. Astfel, acolo unde gradul monomului este 1 acesta va afisa doar coeficientul sau urmat de X, iar acolo unde gradul este 0 si coeficientul este diferit de 0, va afisa doar coeficientul.

Am declarat coeficientul ca fiind real pentru a rezolva problema impartirii respectiv integrarii polinoamelor. Initial, in functia toString() de afisare m-am folosit de metoda Math.abs() care afiseaza cel mai apropiat intreg de numarul respectiv dar am considerat ca asa nu-si are rostul ca sa declar coeficientii de tip real, astfel afisand coeficientii de tipul real.

Toate testele efectuate pana acum au demonstrate faptul ca operatiile pe polinoame functioneaza asa cum ar trebui, mai putin impartirea acestora. Impartirea este o problema mai dificila, iar programul are mici erori aici, erori care vor putea fi remediate.

**Diagrama UML a claselor**



**Implementare**

Dupa cum se poate vedea si din diagrama UML a claselor, aplicatia noastra dispune de 12 clase. Aceste clase impreuna cu metodele si functionalitatile lor vor fi descries mai jos.

* Clasa ‘Monom’

Aceasta clasa implementeaza interfata Comparable cu ajutorul careia am reusit sa comparam 2 monoame.

Clasa are 2 atribute: coeficientul si gradul polinomului. Acestea sunt private deoarece utilizam conceptul de incapsulare a datelor.

Constructorul clasei este folosit pentru a crea un nou Monom cu coeficientul si gradul specificate

public Monom(double coeficient, int grad)

Dupa acest constructor se afla 4 metode care nu fac altceva decat sa seteze sau sa returneze coeficientul respective gradul monomului. De altfel, aceste 4 metode se numesc get-ere si set-ere.

Urmatoarea metoda numita compareTo este o metoda care ne ajuta sa comparam doua monoame intre ele, comparatie care se face in functie de gradul monoamelor. Daca gradul primului monom este mai mare decat gradul celui de-al doilea atunci returnam 1, altfel returnam 0.

public int compareTo(Object monom)

Metoda impartireMonoame este o metoda care ne ajuta sa impartim doua monoame si rezultatul sa fie transmis tot ca un Monom. Coeficientii celor 2 monoame de impartit se imparte si el iar din gradul primului monom scadem gradul celui de-al doilea monom si astfel obtinem monomul rezultat.

public Monom impartireMonoame(Monom m1)

* Clasa ‘Polinom’

Aceasta clasa are 3 constructori si anume grad de tipul integer, polinom care este un ArrayList de Monoame si lungimePolinom tot de tipul integer. Asa cum ni se precizeaza in cerinta temei, clasa Polinom dispune de o lista de Monoame.

Clasa Polinom are un constructor unde se creeaza un obiect de tipul Polinom si gradul acestuia se initializeaza cu -1. Puteam sa-l initializez si cu 0 dar tinand cont de faptul ca monoamele trebuie introduse in interfata grafica cu tot cu coefficient indifferent care ar fi acesta, am preferat sa initializez gradul unui nou polinom cu -1 ca sa evit orice eroare care ar fi putut sa apara pe parcurs.

public Polinom()

Urmatoarele 4 metode ale clasei Polinom sunt metode accesoare adica set-ere si get-ere.

public void addMonom(Monom m)

Metoda addMonom este o metoda care nu returneaza nimic. Aceasta metoda este folosita pentru a adauga un monom in lista de monoame a polinomului nostru. In momentul in care se face apelul functiei se verifica daca monomul care se doreste a fi adaugat este mai mare decat gradul polinomului. In caz afirmativ, polinomului primeste valoarea gradului monomului de inserat iar lungimea polinomului se incrementeaza cu 1.

public Polinom negarePolinom(Polinom p)

Metoda negarePolinom este o metoda care mi s-a parut foarte utila si a meritat spre a fi implementata. Aceasta metoda isi dovedeste utilitatea in momentul in care dorim sa facem scaderea a doua polinoame. Scopul acestei metode este de a nega toti coeficientii polinomului trimis ca parametru. Asta inseamna ca daca avem spre exemplu polinomul 3x^2 +5x^3 -4x^1, dupa apelarea metodei negarePolinom acestea devine -3x^2 -5x^3 +4x^1. Consider foarte utila aceasta metoda deoarece se stie ca operatie de scadere este negarea operatiei de adunare. Astfel ca a-b se traduce ca fiind a+(-b).

public String toString()

In afara de cele 2 metode de set si get a atributului lungimePolinom, mai avem o metoda destul de stufoasa numita toString. In aceasta metoda am incercat sa facem un pretty-print a polinomului. Pentru a face acest lucru m-am gandit la toate posibilitatile pentru ca un polinom sa arate cat mai frumos, astfel, daca gradul polinomului este 0, nu-l vom mai afisa pe X, vom afisa doar coeficientul monomului, daca gradul monomului este 1 vom afisa doar coefficient\*X si multe altele.

* Clasa ‘Operatie’

Clasa Operatie este cea mai mare clasa din programul nostru, dar si cea mai utila. In aceasta clasa se implementeaza toate operatiile pe polinoame. Aceasta clasa nu are nici un atribut si nici un alt constructor decat cel implicit. Cum spuneam si mai sus, este folosita doar pentru implementarea operatiilor pe polinoame.

public Polinom adunarePolinoame(Polinom p1, Polinom p2)

Numele este intuitiv asa ca este foarte usor de ghicit ce face aceasta clasa: aduna doua polinoame pe care le primeste ca si parametru si returneaza polinomul rezultat care este declarant in interiorul clasei.

Pentru a realiza adunarea polinoamelor se foloseste urmatoarea strategie: se iau, rand pe rand, elementele din primul polinom si se compara cu toate celelalte elemente din al doilea polinom. In momentul in care se gasesc doua polinoame care sa aiba acelasi grad, coeficientii lor se aduna, gradul ramane acelasi si adaugam noul monom in polinomul result declarat la inceputul metodei. Daca nu s-au gasit 2 monoame care sa aiba acelasi grad inseamna ca in al doilea polinom nu exista un monom care sa aiba acelasi grad cu monomul din primul polinom si astfel in polinomul result se adauga monomul de pe pozitia curenta din primul polinom. Dupa ce s-au parcurs toate monoamele din primul polinom, se parcurg din nou, rand pe rand, toate monoamele din al doilea polinom cu aceeasi strategie de mai sus, doar ca se cauta in polinomul result. Adica cautam cate un correspondent din al doilea polinom in polinom result. In cazul in care nu gasim doua monoame cu acelasi rezultat, este clar ca in primul polinom nu exista un monom cu acelasi coefficient al monomului current din al doilea polinom asa ca monomul current din al doilea polinom se adauga in polinomul result. Dupa cele 2 parcurgeri vom avea in polinomul result rezultatul adunarii celor 2 polinoame primite ca si parametru al functiei noastre.

public Polinom scaderePolinoame(Polinom p1, Polinom p2)

Dupa cum sugereaza si numele, in aceasta metoda se face scadere celor doua polinoame, respective din polinomul p1 se scade polinomul p2. De fapt, aceasta metoda apeleaza metoda adunarePolinoame, dar inainte de a face acest lucru, ne asiguram ca este apelata metoda negarePolinom care, cum am spus si mai sus, neaga toti coeficientii polinomului primit ca si parametru. Dupa apelarea acestei metode totul devine foarte simplu deoarece scaderea se transforma intr-un alt tip de adunare. Este demonstrate faptul ca a+b = a+(-b) asa ca metoda scaderePolinoame implementeaza aceasta demonstratie matematica, iar descrierea acestei clase, care la inceput credeam ca se va intinde pe multe linii, ocupa doar 5 linii de cod.

public Polinom derivarePolinom(Polinom p1)

Metoda derivarePolinom este metoda care implementeaza metoda de derivare a polinomului primit ca parametru. Polinomul primit este parcurs monom cu monom si pe acesta se aplica metoda de derivare. In cazul in care gradul monomului este mai mare ca 1, gradul current se inmulteste cu coeficientul monomului si mai apoi se seteaza gradul monomului ca fiind gradul precedent -1. In cazul in care gradul monomului este 1, dupa derivare ne va ramane doar coeficientul monomului.

public Polinom integrarePolinom(Polinom p1)

Aceasta metoda este metoda care integreaza polinomul primit ca si parametru. Se parcruge polinomul monom cu monom, iar gradul monomului devine gradul current + 1 iar coeficientul se imparte la gradul monomului care a fost inainte de a fi decrementat cu 1.

public Polinom inmultirePolinoame(Polinom p1, Polinom p2)

Aceasta metoda nu a fost foarte greu de implementat deoarece se iau rand pe rand monoamele din primul polinom si se inmultesc cu fiecare monom din cel de-al doilea polinom dupa care se adauga monomul obtinut in polinomul result declarant la inceputul metodei. Dupa aceasta etapa se sorteaza elementele polinomului obtinut in functie de grad, in ordine descrescatoare.

public Polinom impartirePolinoame(Polinom deimpartitul, Polinom impartitorul)

Metoda de impartire este o metoda care nu functioneaza asa cum ar trebui. Aceasta primeste ca parametru doua polinoame si ar trebui sa le imparta, doar ca nu intotdeauna ne returneaza rezultatul corespunzator. Aceasta metoda ar trebui revizuita.

* Clasa ‘Interfata’

In aceasta clasa sunt definite toate butoanele carora le-am atribuit diferite operatii, text-field-uri si label-uri, dimensiunea ferestrei implementate dar si metodele de actionLisetener.

* Clasa Adunare

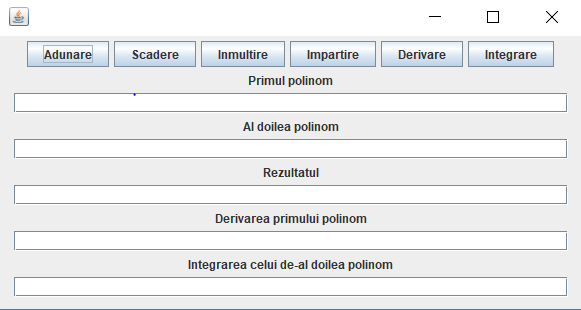
Aceasta clasa este asemanatoare cu restul claselor respective Scadere, Inmultire, Impartire, Derivare, Integrare, adica sunt clasele unde setam text-field-uri si le dam functionalitate butoanelor.

* Clasa App

Aceasta clasa este de fapt clasa care implementeaza functia main, adica locul unde setam Interfata ca fiind vizibila.

**Rezultate**

In momentul in care rulam aplicatia, ea arata astfel:



**Concluzii**

In concluzie, consider ca aceasta tema a fost foarte importanta si benefica pentru a ne da seama cum se lucreaza in Java in momentul in care avem proiecte mari de facut.

Initial, problema implementarii unei aplicatii Java care sa opereze pe polinoame a parut a fi destul de simpla, dar cand a trebuit sa facem implementarea propriu-zisa ne-am intalnit cu o multitudine de erori si cu probleme de logica deoarece gandirea noastra nu era formata pentru programarea orientata pe obiect.

Ca dezvoltari ulterioare, ar trebuie sa se lucreze la metoda de impartire care nu functioneaza correct si cel mai probabil la eficienta programului. Cu siguranta se poate optimiza cu alti algoritmi sau alta logica, dar pentru operatii pe polinoame simple este indeajuns de efficient.

**Bibliografie**

* http://stackoverflow.com/questions/
* http://docs.oracle.com/en/
* http://www.tutorialspoint.com/java/