DOCUMENTAȚIE TEMA 1

CALCULATOR DE POLINOAME

Ghițun Patricia Roxana

Grupa 30227

Profesor laborator : Teodor Petrican

**Cuprins**

**1. Obiectivul temei**

**2. Analiza problemei**

**3. Proiectare**

**4. Implementare**

**5. Testare**

**6. Rezultate**

**7. Concluzii**

**8. Bibliografie**

1. **Obiectivul temei**

Obiectivul acestei teme este de a implementa un sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi .

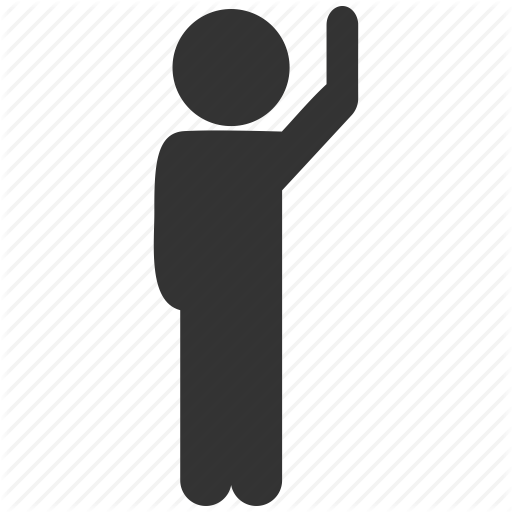
Un polinom este o expresie matematică construită dintr-una sau mai multe variabile și constante. Polinoamele sunt construite din monoame, care sunt alcătuite dintr-o constantă (= coeficient) înmulțită cu o variabilă. Fiecare variabilă poate avea un exponent constant întreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile în acel monom.

Scopul acestei aplicații este de a efectua operații de bază cu polinoame : adunare , scădere , înmulțire, derivare și integrare . Utilizatorul poate sa introduca doua polinoame in spatiile dedicate , iar apoi poate sa aleaga ce operatii doreste sa faca , dupa care poate observa rezultatul afisat pe ecran .

1. **Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori**

Aplicatia trebuie sa execute in mod corect cele cinci operatii asupra celor doua polinoame introduse printr-o metoda Oriented Objected. Operatiile de adunare , scadere si inmultire se realizeaza pe doua polinoame , iar derivarea si integrarea pe un singur polinom . Modul ales de reprezentare a polinoamelor este de a le introduce ca un sir de caractere , fara spatii intre ele .

**Use case :**

****

Scădere

Integrare

Derivare

Adunare

Înmulțire

Utilizatorul poate sa introduca polinoamele intr-un mod cat mai simplist , poate sa aleaga operatie pe care doreste sa o realizeze doar prin apasarea unui buton si poate sa analizeze rezultatul pe ecran .

Polinoamele se pot introduce intr-un mod cat mai simplu .

Exemplu de introducere polinom :

X+1 va fi interpretat ca : 1x^1+1x^0

X^2 va fi interpretat ca : 1x^2 (1-coeficient , 2-putere)

**3.Proiectare**

In aplicatia de fata am folosit urmatoarea structura de organizare :

Polinoamele sunt reprezentate de un obiect de tip Polinom , care are ca atribute un grad (de tip intreg) si o lista de monoame (ArrayList<Monom>) . Monoamele sunt si ele la randul lor obiecte denumite Monom , care au ca atribute un coeficient , o putere , si un marcaj (intregi) . Deci , un obiect de tipul Polinom va fi construit dintr-o lista de obiecte de tipul Monom . Puterea , coeficientul si marcajul unui singur monom sunt de tipul intreg .

Exemple de monoame : x^2 , 2x , 56x^485 , etc

**Ghid de utilizare**

Pentru a putea folosi aceasta aplicatie , este necesar ca utilizatorul sa urmeze acesti pasi :

1. Utilizatorul ruleaza aplicatia
2. Utilizatorul introduce datele in spatiile dedicate celor doua polinoame (textfield-urile P1 si P2)

* Introducerea datelor se va face fara spatii intre monoame .
* Datele se introduc dupa urmatorul model (ca string-uri):

SEMN + COEFICIENT + X + ^ + PUTERE

Semnul poate sa lipseasca , atunci se va considera implicit +

Coeficientul poate sa lipseasca , va fi considerat ulterior ca 1

Puterea poate sa lipseasca , va fi considerate ca 1 (monom de grad 1 = x) daca inainte de aceasta este si caracterul “x” , altfel 0 considerand ca este vorba despre un termen liber (1 = 1x^0)

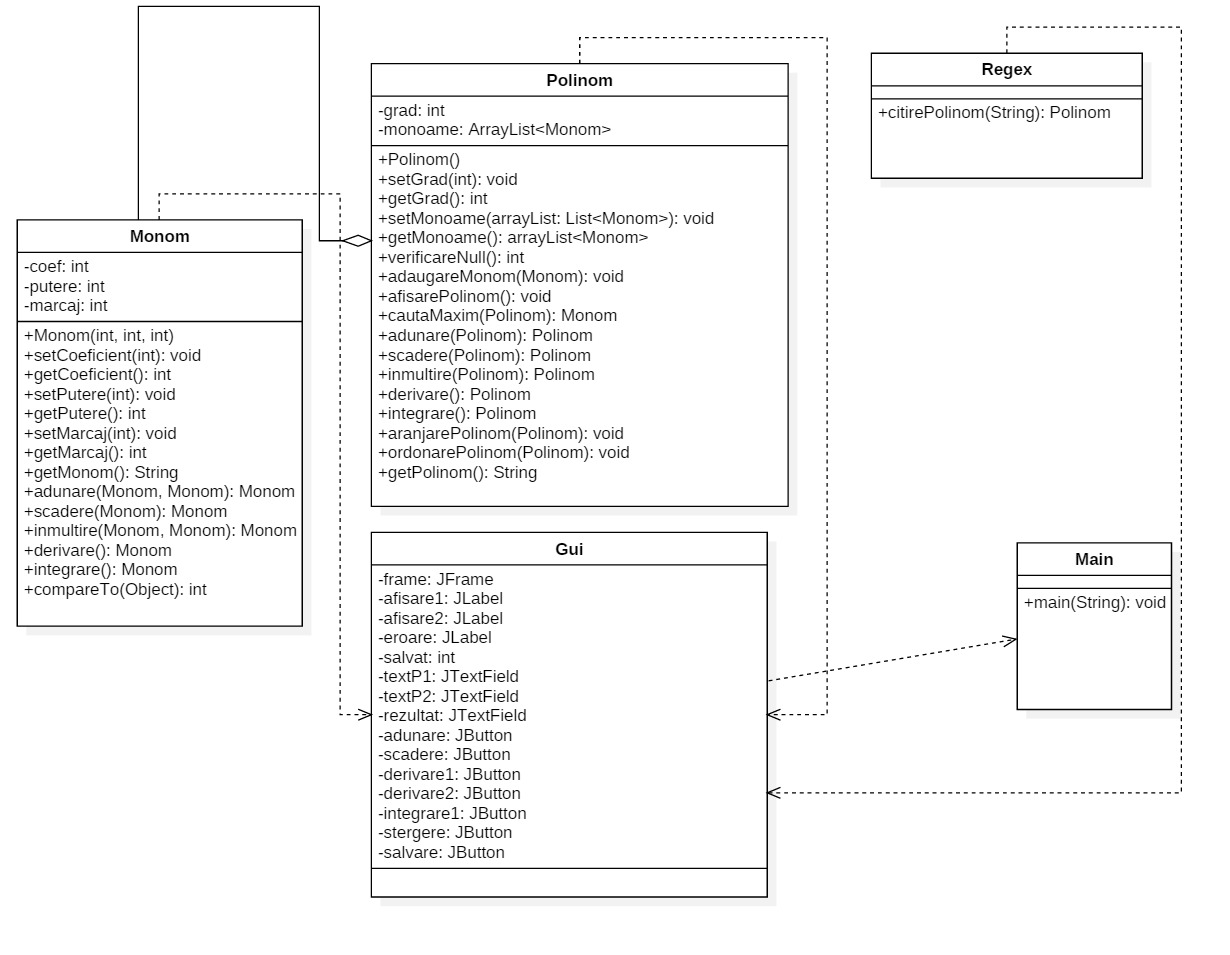
* Datele pot fi introduse si intr-un mod “neordonat” precum : 3x+2x+x^2 , iar programul il va considera ca x^2+5x^1

1. Dupa ce au fost introduce datele , se va apasa butonul “Salvare” pentru a le salva si pentru a putea efectua operatiile folosindu-le .

* Operatiile nu se pot implementa daca nu au fost introduce correct polinoamele !

1. La apasarea oricarui buton referitor unei operatii , in TextField-ul pentru rezultat se va afisa rezultatul operatiei respective .
2. Pentru a incepe o noua sesiune de calcul a aplicatiei , se poate apasa butonul “Stergere“ sau butonul Exit(X) .

Aceasta aplicatie contine 6 clase , dintre care 3 pentru partea de “back end” si 3 pentru partea de “front end ” (printre care si main).

**Diagrama de clasa :** 

**4. Implementare**

Proiectul este structurat in doua pachete . Pachetul de front end care include clasele Gui, Regex si Main , si pachetul back end ce contine clasele Monom si Polinom .

**Clasa MONOM**

Aceasta clasa are trei atribute de tip intreg declarate private : coef , putere si marcaj .

* Coef : coeficientul monomului (pozitiv sau negativ)
* Putere : puterea pozitiva a monomului
* Marcaj : Este 1 daca monomul a fost adaugat in noul polinom suma/diferenta si

0 in caz contrar . Acest atribut a fost introdus pentru a usura implementarea metodelor de adunare si scadere .

In afara constructorului cu parametri (int c , int p , int m) care initializeaza valorile unui monom la instantierea lui , si in afara metodelor de get si set care permit programatorului accesul la atributele declarate private , in aceasta clasa mai sunt si urmatoarele metode :

* getMonom : Aceasta metoda returneaza un monom sub forma de string pentru a putea afisa ulterior tot polinomul .
* Adunare : Aceasta metoda aduna coeficientii monoamelor cu aceeasi putere si rezulta suma lor .
* Scadere12 : Aceasta metoda returneaza un monom ce are ca rezultat diferenta : A – B . Il scade din primul monom pe al doilea .
* Inmultire : In aceasta metoda se inmultesc coeficientii si se aduna puterile a doua monoame .
* Derivare : se inmulteste coeficientul cu puterea , dupa care aceasta scade .
* Integrare : Se incrementeaza puterea , dupa care se imparte coeficientul cu aceasta . Rezultatul acestei metode este intreg (=> coeficienti intregi).

Aceasta clasa implementeaza interfata Comparable<Object> pentru a fi posibila suprascrierea metodei compareTo(Object o) . Acest lucru este folosit in metoda de sortare a polinomului dupa putere in clasa Polinom . In acest fel , polinoamele rezultate si afisare sunt ordonate in functie de exponent .

**Clasa POLINOM**

Aceasta clasa permite crearea de polinoame sub forma de lista de monoame . Variabilele instanta din aceasta clasa sunt : un intreg “grad” ce reprezinta gradul polinomului si un ArrayList<Monom> o lista de obiecte ce apartin clasei Monom .

Cu ajutorul constructorului se initializeaza de fiecare data gradul polinomul si lista de monoame .

In aceasta clasa se afla metodele principale ale aplicatiei , operatiile .

Metode :

* **verificareNull** : returneaza 1 daca toti coeficientii unui polinom sunt 0 (daca este nul ) si usureaza afisarea sa .
* **adaugareMonom** : adauga un monom in polinomul current
* **ordonarePolinom** : Aceasta metoda foloseste metoda implementata in clasa Monom compareTo pentru a sorta tot polinomul .
* **aranjarePolinom** : Aceasta metoda a fost implementata cu scopul de a elimina monoamele duplicate introduse de catre utilizator . De exemplu , daca se introduce polinom x^3 + 2x + 5x^3 + 2x^3 si se apeleaza aceasta metoda , polinomul rezultat va avea forma urmatoare : 8x^3 + 2x^1 deoarece restrange monoamele cu aceeasi putere si apeleaza la inceput metoda de ordonare .

Metoda aceasta este folosita la fiecare metoda principala pentru a avea un rezultat cat mai corect .

* cautaMaxim
* **afisarePolinom :** afiseaza polinomul monom cu monom . Se foloseste metoda din clasa Monom (getMonom) pentru a afisa pe rand fiecare monom din polinom sub forma de string .

**Operatiile fundamentale** : adunare , scadere , inmultire , derivare si integrare pe polinoame :

**ADUNARE**

**public** **Polinom** **adunare** (**Polinom** p2 )

Metoda va fi apelata astfel : A.adunare(B) . Nu se modifica cele doua polinoame A si B , rezultatul trebuie retinut intr-un nou polinom .

Aceasta metoda ia in considerare ambele cazuri : gradul primului polinom > sau < decat gradul celui de-al doilea polinom .

Pentru fiecare caz , se creeaza un nou polinom suma care va avea initial gradul polinomului mai mare si o lista noua de monoame (“vector”) care va fi ulterior lista din polinomul “suma” . Dupa aceea , se parcurg cele doua polinoame monom cu monom si daca se gasesc monoame cu aceeasi putere , se aduna folosind metoda din clasa Monom . Suma celor doua monoame va fi adaugata in lista de monoame a sumei („vector”) dupa care se marcheaza monoamele din p1 si p2 cu valoarea 1 (=marcate -> au fost adaugate deja in noua suma ).

Dupa ce se efectueaza aceasta parcurgere , se verifica daca cumva sunt coeficienti 0 in “vector” . Daca sunt , se vor sterge monoamele respective .

In ultimul rand , se va parcurge din nou fiecare polinom p1 si p2 pentru a adauga ce nu a fost adaugat deja (ce este marcat ca si 0) , dupa care se seteaza “vectorul” in polinomul suma ca lista de monoame si se returneaza suma ordonata si aranjata .

**SCĂDEREA**

**public** **Polinom** **scadere**( **Polinom** p2)

Si scaderea va fi apelata exact la fel ca si adunarea a doua polinoame . Polinomul rezultat este diferenta dintre primul polinom si al doilea , adica : dif=A.scadere(B) => dif=A-B

Pentru inceput , se creeaza trei liste de monoame . Prima lista reprezinta lista finala ce va fi returnata in polinomul dif(diferenta rezultata ca polinom) , iar urmatoarele doua reprezinta listele de monoame ale polinoamelor pe care dorim sa le scadem .

Similar cu adunarea , pasii prin care se trece in aceasta metoda sunt aceeasi , doar ca in loc sa se adune monoamele cu aceeasi putere , acestea se scad (se adauga diferenta lor in lista diferentei “coef” , dupa care se marcheaza fiecare monom cu 1).

In final , se sterg coeficientii cu 0 din lista diferentei , dupa care se parcurge din nou primul polinom si se adauga ce nu a fost adaugat deja cu semnul + , iar cele din al doilea polinom cu semn –

**ÎNMULȚIREA**

**public** **Polinom** **inmultire**(**Polinom** p2)

In aceasta metoda se parcurg cele doua polinoame si se apeleaza metoda de inmultire din clasa Monom pentru cele doua monoame la momentul curent . Rezultatul este adaugat ulterior in polinomul ce va fi rezultat la finalul metodei .

**DERIVAREA**

**public** **Polinom** **derivare**()

Metoda aceasta va returna un nou polinom si va modifica si polinomul curent .

Se parcurge polinomul monom cu monom si se apeleaza metoda de derivare din clasa Monom pentru monom current . Dupa terminarea acestei bucle , se parcurge polinomul rezultat si daca se gaseste vreun monom cu coefficient 0 sau o putere negativa , se sterge monomul respectiv.

Daca tot polinomul este nul ( se verifica polinomul cu metoda verificareNull ) atunci se va adauga in el un monom cu coeficientul 0 .

**INTEGRAREA**

**public** **Polinom** **integrare**()

Este identica cu derivarea doar ca in loc de metoda de derivare , se foloseste metoda de integrare din clasa Monom pentru fiecare monom al polinomului .

Similar cu derivarea , aceasta metoda va modifica polinomul initial .

**Clasa GUI**

Componentele de baza folosite in aceasta clasa sunt :

* JButon : pentru butoanele operatiilor (de ex : cate 2 butoane pentru fiecare operatie de derivare sua integrare – P1 si P2 ) si butoanele de salvare si stergere .
* JLabel : folosite pentru a afisare denumiri , afisarea polinoamelor sau eroare .
* JTextField : trei textfield-uri folosite pentru polinoame (2 pentru introducere si unul pentru afisarea rezultatului)

Pentru fiecare buton creat , am apelat metoda de addActionListener(new ActionListener()) , iar in momentul acesta se creeaza o instanta de clasa ActionListener si suprascriu metoda actionPerformed in care scriu ce doresc sa faca butonul respectiv . In acest mod , toate metodele sunt scrise in clasa Gui .

**Clasa Regex**

In aceasta clasa este implementata metoda statica citirePolinom(String s) prin care se introduce un string (formatul polinomului de la tastattura) si se converteste cu ajutorul unui Pattern .

Pattern-ul pentru un monom este urmatorul : "([+-]?)([0-9]\*?)[xX](\\^([0-9]+[1-9]\*))?|([+-]?)([0-9]+[1-9]\*)"

In acest pattern se afla 7 grupuri .

Grupul 0 : este tot monomul .

Grupul 1 : este format din semnul coeficientului unui monom . Monom poate avea un coefficient cu semn + , - sau sa nu existe . ([+-]?) Inseamna ca este un semn + sau – sau nu este deloc .

Grupul 2 : este format din coeficientul monomului fara semn ([0-9]\*?) . Coeficientul poate sa apara , sau nu . Daca apare , poate avea cifre intre 0 si 9 o data sau de mai multe ori . Deci un coeficient poate sa fie si 0 , 23 , 5 , 6777 sau etc .

Grupul 3 : (\\^([0-9]+[1-9]\*))? , reprezinta puterea cu semnul ^ inainte

Grupul 4 : ([0-9]+[1-9]\*) este efectiv exponentul . Acesta poate sa fie si un numar mai mare . Poate aiba o singura cifra sau poate avea mai multe .

Un monom poate sa aiba sau nu putere deoarece grupul 3 are la final “?” ceea ce inseamna ca poate sa apara sau nu . In cazul in care nu avem putere la monom, atunci in grupurile 5 si 6 se vor retine semnul termenului liber(cu puterea 0) si efectiv coeficientul acestuia . ([+-]?)([0-9]+[1-9]\*)

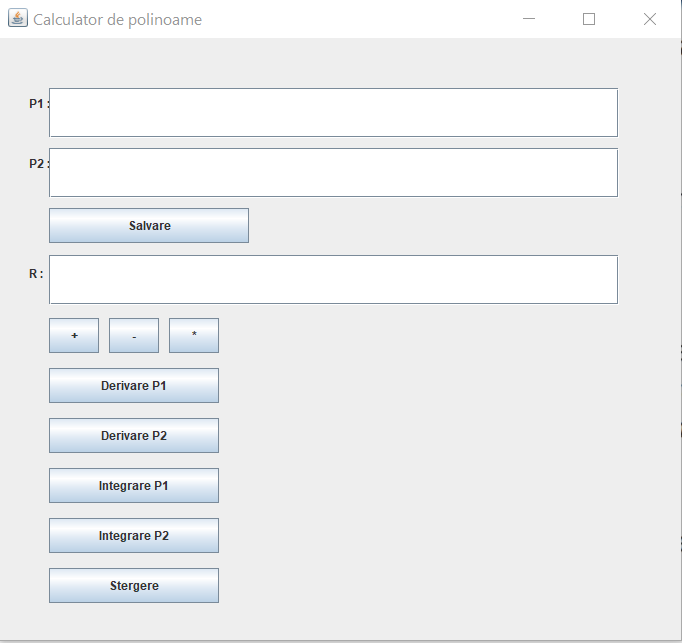
1. **Testare**

Testarea aplicatia a fost realizata cu biblioteca Junit .

Pentru testarea algorimitlor implementari , a fost creata o clasa Teste in pachetul back end . S-au testat cele cinci operatii iar in fiecare metoda se dau valori polinoamelor declarate global si se mai creeaza inca doua polinoame rezultatCorect si rezultatOp . Rezultatul correct este cel care ar trebuie sa rezulte in urma operatiei iar rezultatul Op este rezultatul generat de test .

La sfarsitul metodei , se foloseste assertEquals() care presupune ca cele doua stringuri date ca parametru sunt identice , iar in cazul in care acestea nu sunt identice , la rulare se vor genera erori .

1. **Rezultate**

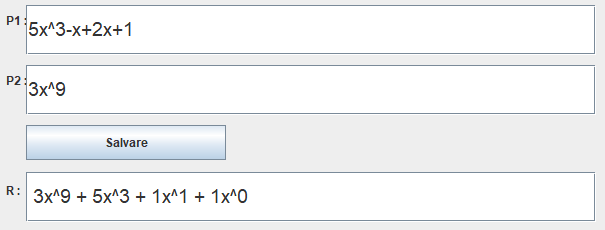
La rularea aplicatiei va aparea urmatoarea fereastra :

In dreptul label-urilor P1 si P2 se vor introduce polinoamele iar dupa apasarea butonului “Salvare” , utilizatorul va putea vedea raspunsul oricarei operatii . Rezultatul unei operatii nu va fi afisat daca cele doua polinoame nu au fost salvate sau daca nu au fost introduse corect .

In functie de operatia aleasa , rezultatul va fi afisat in al treilea textfield .

La apasarea butonului stergere , se vor sterge polinoamele introduse si va fi posibila introducerea unor noi polinoame .

Daca utilizatorul nu introduce nimic in campurile destinate polinoamelor , si doreste sa se salveze , va fi afisata eroarea , iar daca se apasa pe una dintre operatii , se va afisa “Introduceti datele”.



De exemplu , pentru operatia de adunare a doua polinoame ,

Programul va afisa urmatorul rezultat .

Polinoamele pot fi introduse cu sau fara semn , iar termenul liber poate fi doar o constanta .

1. **Concluzii**

Aplicatie este functionala pentru orice tip de polinom si efectueaza correct cele cinci operatii implementate .

Implementand aceasta tema mi-am imbogatit cunostintele de programare si am invatat sa folosesc

Ca o dezvoltare ulterioara , se poate modifica algoritmul in asa fel incat coeficientii monoamelor sa fie de tip real si astfel rezultatul integrarii sa fie un rezultat cat mai correct (real).

1. **Bibliografie**

**<https://www.regular-expressions.info/tutorial.html>**

**<https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html>**

**<http://users.utcluj.ro/~igiosan/teaching_poo.html>**

**<https://www.tutorialspoint.com/java/java_regular_expressions.htm>**

**<https://www.tutorialspoint.com/javaexamples/java_simple_gui.htm>**