Documentatie Tema 1

Calculator Polinoame

Nume: Pop Alin

Grupa: 30227

1

Cuprins:

1) Cerinte functionale 1

2) Obiective 1

3) Analiza problemei 1

4) Proiectare 1

5) Implementare 1

6) Testare 1

7) Concluzii si dezvoltari ulterioare 2

8) Bibliografie 1

2

1. Cerinte functionale:

Dezvoltati in limbajul de porgramare java un calculator pentru polinoame care sa fie capabil sa efectueze urmatoarele operatii:

1. Citirea unui polinom de la tastatura sub forma 3x^2+x^2-x^1+1
2. Operatia de adunare intre doua astfel de polinoame
3. Operatia de scadere intre doua polinoame
4. Derivarea unui polinom
5. Integrarea unui polinom
6. Inmultirea a doua polinoame
7. Impartirea a doua polinoame
8. Obiective

Obiectivul principal este propunerea,proiectarea si implementarea unui sistem de procesare a polinoamelor de o singura variabila cu coeficienti numere intregi.

1. Analiza problemei

In matematica , un polinom este o expresie construita dintr-una sau mai multe variabile si constant , folosind operatii de adunare , scadere , inmultire , impartire , derivare sau integrare..

Se observa in particular faptul ca impartirea printr-o expresie ce contine o variabila nu este posibila la polinoame.

Polinoamele sunt constituite din termini numiti monoame , care sunt alcatuite dintr-o constanta numita coefficient, inmultita cu una sau mai multe variabile. Fiecare variabila poate avea un exponent constant intreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile in acel monom. Coeficientul unui monom poate fi orice numar, inclusiv fractii, numere irationale sau negative. Un exemplu de monom este: -3x^2.

1. Proiectare

Structura de date pe care am folosit-o in acest proiect este ArrayList<>.Am folosit aceasta structura de date pentru a reprezenta polinoamele ca liste de monoame ArrayList<Monom>.

Acest tip de structura de date are implementate un set de operatii pe care le putem folosi in codul nostru. Niste exemple de astfel de operatii sunt :

-arrayList.size() = care returneaza numarul de ele mente din lista noastra;

-arrayList.get(i) = care returneaza elemental de pe pozitia I din lista

-arrayList.clear() = sterge toate elementele din lista

- arrayList.remove(n) = sterge elemental de pe pozitia n

-arrayList.isEmpty() = returneaza true dca lista nu contine nici un element

1. Implementare

Proiectul este format din 5 clase , toate fiin incluse in acelasi pachet. Clasele care compun proiectul sunt : Monom , Polinom , OperatiiPolinoame , UI si clasa Test1 .

In aceasta clasa avem implementarea monomului care are ca si parametri o variabila intreaga de tip int , putere , care reprezinta puterea monomului si o variabila reala de tip float , numar , care reprezinta coeficintul monomului.

Constructorul cu parametric are rolul de a initializa monomul cu variabilele pe care I le trimitem ca si parametri.

Metoda public int getPutere din clasa Monom returneaza o variabila de tip int , aceasta fiind totodata si puterea monomului la care ne referim.

Metoda public void setPutere(int putere) din clasa Monom are rolul de a seta puterea monomului pentru care este apelata cu valoarea variabilei de tip int putere pe care 0 primeste ca si parametru

Metoda public float getNumar() din clasa Monom are scopul de a valoarea unei variabile de tip float , aceasta valoare fiind corespondentul coeficientului polinomului pentru care este apelata.

Metoda public void setNumar(float nr) din clasa Monom are rolul de a seta coeficientul monomului pentru care este apelata cu valoarea variabilei de tip float , nr pe care o primeste ca parametru.

Metoda public void suma(float nr) din clasa Monom are scopul de a adauga la valoarea curenta a coeficientului din monomul pentru care este apelata o valoare de tip float , nr pe care o primeste ca si parametru.

La final am suprascris metoda toString pentru a face o afisare lizibila pentru monomul nostru.

Am interpretat anumite cazuri pentru ca afisarea sa fie cat mai potrivita , de exemplu , pentru cazul in care puterea este 0 vom afisa doar coeficientul , pentru cazul in care puterea este 1 nu o vom mai afisa , fiin cea implicita, iar pentru caul cand coeficientul este 0 iar puterea mai mare decat 0 nici pe acesta nu il vom afisa.

Clasa Polinom:

Clasa Polinom are doua variabile de tip private , pol si rank.In pol se va retine un arrayList de monoame bazat pe inputul pe care il dam.constructorul Polinom(string s) pune intr-un arraylist de stringuri un input format din coeficientii si puterile stringului pe care il introducem de la tastatura.Acest lucru se realizeaza prin apelarea metodei input care returneaza un arraylist de stringuri. Aceasta functia ia stringul pe care il primeste ca parametru si incearca s ail parseze ,astfel ca in comp sa avem pe pozitii consecutive coeficientul si puterea fiecarui monom.Aceaqsta functie trateaza si cazurile particulare in care daca lungimea lui comp e 0, adica stringul introdus ca parametru e null , va pune un 1 , iar daca in comp avem un semn – sau + care nu este urmat de nici un numar, vom pune 1.

Dupa aceea , daca arraylistul nostrum pe care vrem s ail returnam nu are un numar par de elemente , adica perechi de coefficient si putere , la final vom adauga un 0 care ar fi puterea ultimului element.

Tot in aceasta functie se verifica daca in stringul introdus gasim o putere ca apare de mai multe ori , adunam toti coeficientii pentru puterile care sunt egale si eliminam duplicatele. La final functia returneaza variabila in care este un array de stringuri format din perechi de forma (coefficient,putere).

Acuma in variabila in avem stringul de la input parsat. Acuma parcurgem acest sir , luam fiecare pereche de tip coefficient putere , convertim valoarea lor la int si le retinem , ca ulterior in variabila pol declarata in clasa Polinom sa retinem monoamele prin setarea coeficientului si puterii pentru fiecare monom.La final retinem in variabila rank marimea polinomului , aceasta fiin echivalenta cu puterea cea mai mare din intreg polinomul.

Constructorul Polinom(ArrayList<Monom>) initializeaza polinomul actual cu cel transmis ca parametru , si variabila rank de asemena cu marimea polinomului transmis ca parametru.

Metoda public ArrayList<monom> getPol() din clasa Polinom are rolul de a returna polinomul la care se face referinta.

Metoda public void setPol(ArrayList<Monom> pol) are rolul de a seta valoarea polinomului la care se face referinta cu valoarea polinomului care il primeste ca si parametru.

Metoda public int getRank() din clasa Polinom are rolul de a return o variabila de tip int a carei valoare este corespondentul rankului pentru polinomul la care sa face referinta.

Metoda public void setRank(int rank) din clasa Polinom are rolul de a seta valoarea variabilei rank la care se face referinta cu valoarea variabilei pe care o primeste ca si parametru.

La final am rescris metoda toString() pentru o afisare cat mai lizibil posibila a polinoamelor cu care va opera programul.

Clasa OperatiiPolinoame :

In clasa operatii polinoame gasim implementarea metodelor pentru operatiile care trebuie sa fie executate pe polinoame.

Prima operatie implementat este adunarea. Pentru a implementa adunarea am facut o metoda static public Polinom adunare(Polinom pol1,Polinom pol2) care retine in doua variabile cei doi polinomi transmisi ca si parametric.Retinem in alte doua variabile lungimile polinoamelor .Parcurgem polinomul cu lungime mai mare , iar daca pe pozitiile din polinomul 2 gasim valori le adaugam la aceeasi putere coeficientul celui de al doilea polinom la coeficientul primului polinom.Dupa ce am terminat de parcurs intregul polinom returnam rezultatul.

A doua operatie este cea de scadere. Pentru a implementa operatia de scadere am creeat o metoda static public Polinom scadere(Polinom pol1, Polinom pol2) . In aceasta metoda am retinum valorile celor doi polinomi transmisi ca parametric functiei, le-am retinut si marimea. Am parcurs cei doi polinomi simultan cat timp am avut valori in ambii polinomi , iar in variabila in care am retinut primul polinom , am adunat coeficientul fiecarui monom din cel de al doilea polinom la monomii cu aceeasi putere din primul polinom , dar cu semnul - . La final daca , al doilea polinom are gradul mai mare decat primul , am adaugat la variabila rezultat monomii ramasi cu semn schimbat pentru coeficientii .

Derivarea: pentru implementarea operatiei de derivare a unui polinom am creeat , in clasa OperatiiPolinoame metoda static public Polinom derivare (Polinom pol). Aceasta metoda retine intr-o variabila polinomul , parcurge toate elementele sale si seteaza pentru fiecare monom din polinom coeficientul cu coeficientul inmultit cu puterea monomului sis cade puterea cu 1 pentru fiecare monom. La final returneaza polinomul rezultat.

Integrarea: pentru implementarea operatiei de integrare a unui polinom am creeat in clasa OperatiiPolinoame o noua metoda static public Polinom integrare (Polinom pol) , in care retinem polinomul transpmis ca parametru intr-o variabila de tip arrayList<Monom>. Parcurgem aceasta lista de monoame , iar pentru fiecare monom in parte ii modificam coeficientul cu coeficientul/puterea+1 si crestem puterea cu 1. La final returnam polinomul rezultat.

Inmultire: pentru a implementa operatia de inmultire dintre doua polinoame am creeat o noua metoda in clasa Operatii polinoame static public Polinom inmultire(Polinom a,Polinom b), in care am retinut in doua variabile pol1 si pol2 polinoamele transmise ca si parametru.Am creeat un nou arraylist<Monom> de lungimea puterilor maxime a celor doi polinomi adunate.Am parcurs cei doi polinomi , am facut suma puterilor pentru fiecare monom parcurs si am setat coeficientul pentru monomul cu suma puterilor respective cu produsul intre cei doi coeficienti ai monoamelor pentru care am afcut suma puterilor. La final am returnat polinomul rezultat.

Impartire: pentru a implementa impartirea dintre doua polinoame am creat o metoda in clasa OperatiiPolinoame static public Polinom[] impartire(Polinoma , Polinom b)

In care am retinut polinoamele transmise ca parametric in doua variabile pol1 si pol2.Am creat un arrayList<monom> pentru a retine rezultatul.Am memorat in putere puterea cea mai amre a impartitorului , si in coefficient coeficientul acesteia. Cat timp marimea lui pol1 e mai mare sau egala decat puterea +1 retinem in n marimea polinomului 1, in p facem diferenta dintre puterea lui pol1 si pol2, practice retinem puterea catului. Retinem in coef coeficientul pentru catul nostrum.La rezultat adaugam un monom de putere p si coefficient coef.Dupa aceasta trebuie sa actualizam eimpartitul,din el trebuie sa scadem impartitorul\*catul.Dupa aceea eliminam din pol1 monomul de pe pozitia n pentru ca in pol1 sa ne ramana doar restul impartirii.Dupa aceea inversam rezultatul pentru a-l avea in ordine crescatoare dupa puteri. La final intr-un Polinom[] rezultat retinem cei doi polinomi pentru cat si pentru rest si afisam rezultatul.

Clasa UI:

In aceasta clasa avem implementarea pentru interfata grafica a proiectului.Initial am creat tesxt fielduri pentru introducerea polinoamelor si afisarea rezultatului ex: **private** **static** JTextField *Pol1* = **new** JTextField(50). Am mai creeat butoane , care prin apasarea lor trebuie sa implementeze anumite operatii ex: **private** **static** JButton *deriv1* = **new** JButton("derivare") . Am mai creat si labeluri pentru a ne da seama care este polinomul1, polinomul 2 si rezultatul ex: **private** **static** JLabel *pol1* = **new** JLabel("Polinom 1:").

Metoda public void setRezultat(String rez) are scopul de a seta rezultatul care va fi afisat in text fieldul pentru rezultat cu valoarea rez care este transmisa ca si parametru pentru metoda.

Metoda public String getrezultat() are rolul de a returna textul care se afla in text fieldul pentru rezultat la momentul apelarii ei.

In metoda main avem Fame-ul care ne permite sa vedem interfata.Am creat 3 paneluri JPanel content1=**new** JPanel(); in care am introdus butoanele , text fieldurile si labelurile create anterior.

In continuare avem listenerurile pentru butoane. Aceste clase anonime ne ajuta ca atunci cand apasam un anume buton specificat acesta sa produca o schimbare pe care noi ne-o dorim. Pentru butonul deriv1 am creeat un listener ca atunci cand este apasat , in textfieldul pentru rezultat sa se afiseze derivarea polinomului din primul text field , acest lucru este posibil prin suprascrierea metodei actionPerformed. Acelasi lucru l-am facut si pentru butoanele integr1 , adunareRez , scadereRez , inmultireRez si impartireRez pentru a apela metodele si a efectua operatiile de integrare , adunare , scadere , inmultire si impartire intre polinoamele trimise ca input.

JUnit Test1

Ultima clasa din proiect este clasa Test1 in care am efectuat testarea unitara a operatiilor pe polinoame. In aceasta clasa am definit mai multe polinoame pentru a putea testa operatiile.

Testarea operatiilor se face prin creearea metodelor de test , in care comparam cu functia assertEquals rezultatul operatiilor cu variabile special introduce si calculate de noi .In cazul in care calculele facute de noi sunt bune si metodele sunt corecte testele din JUnit vor fii corecte.

De exemplu daca avem polinomul **private** Polinom pol3=**new** Polinom("4x^3") si **private** Polinom pol4=**new** Polinom("12x^2") si efectuam intr-o metoda de test instructiunea *assertEquals*(OperatiiPolinoame.*derivare*(pol3).toString(),pol4.toString()) aceasta va trebui sa ne spuna ca testul 1 este correct.

1. Concluzii si dezvoltari ulterioare

Aplicatia noastra este una foarte folositoare pentru calcularea de polinaome , cu o interfata placuta , usor de inteloes si care incearca sa corecteze cat mai multe din cazurile in care , daca este folosita de o personae care nu cunoaste prea bine polinoamele , datele introduce de acea persoana sunt gresite.

Ca si o dezvoltare ulterioara ar putea sa verifice mai multe cazuri de input gresit si ar putea sa implementeze mai multe calculi , sa fie transformat intr-un calculator complet , nu doar pentru polinoame.