Universitatea Tehnică Cluj-Napoca

Documentaţie

Tema 1

Calculator Polinoame

Mureşan Ioana Diana

CTI-Romana

Gr.30227

Cuprins

1.Cerinta ........................................................................3

2.Obiective ......................................................................3

3.Consideratii Matematice ..............................................3

4.Analiza Cerintei ............................................................6

5.Proiectare ....................................................................7

5.1.Structura .............................................................7

5.2.Diagrama de Clasa ..............................................7

6.Implementare .............................................................7

7.Testare .......................................................................12

8.Concluzii .....................................................................13

9.Bibliografie .................................................................13

**1.Cerinţa Problemei**

Propuneţi, proiectaţi şi implementaţi un sistem de procesare a polinoamelor de o singură variabilă cu coeficienţi întregi.

**2.Obiective**

 Obtinerea unei aplicatii de tip calculator care preia 2 polinoame de forma

si le prelucreaza astfel incat sa se poata efectua operatii asupra lor.

Operatiile sunt:adunare, scadere, inmultire, impartire, derivare si integrare.

Obiectivele obligatorii pentru acceptarea temei sunt: respectarea paradigmelor OOP, aplicatia organizata pe pachete, clase de maxim 300 de linii(exceptie clasele de UI), metode de maxim 30 de linii(exceptie case cu switch-uri multe) si Java naming conventions.

Cerintelele pentru nota 5 sunt: implementarea interfetei grafice, operatiile de adunare si scadere, iar mai departe se vor implementa si alte operatii pentru o nota mai buna.

Pentru atingerea obiectivului final se parcurg urmatoarele etape:

-alegerea structurilor de date

-impartirea pe clase

-dezvoltarea algoritmilor

-implementarea solutiei

-testare

**3.Consideraţii Matematice**

În matematică, un polinom este o expresie construită dintr-una sau mai multe variabile si constante,folosind doar operatii de adunare, scadere, inmultire si ridicare la putere constanta pozitiva intreaga.

Polinoamele sunt construite din termini numiti monoame, care sunt alcatuite dintr-o constanta, numita coeficient inmultita cu una sau mai multe variabile.Fiecare variabila poate avea un exponent constant intreg pozitiv.Exponentul unei variabile dintr-nu monom este egal cu gradul acelei variabile in acel monom.Un monom fara variabile se numeste monom constant, sau doar constanta.Gradul unui termen constant este 0.Coeficientul unui monom poate fi orice numar , inclusiv fractii, numere irationale sau negative.Un polinom construit cu o singura variabila se numeste univariat.

Un polinom este o suma de unul sau mai multe monoame.De exemplu, urmatoarea expresie este un polinom:

4x^2 -3x+2

El este alcatuit din 3 monoame: primul de gradul 2, al doilea are gradu unu, iar al treilea are gradul 0.

Cand un polinom este dispus în ordinea naturală, el are termenii de grad mai mare înaintea celor de grad mai mic. În primul termen, coeficientul este 4, variabila este *x*, iar exponentul este doi. În al doilea termen, coeficientul este -3. Al treilea termen este o constantă. **Gradul** unui polinom este cel mai mare grad al unui termen al său. În acest exemplu, polinomul are gradul doi.

Un polinom de gradul unu este numit liniar, unul de gradul doi este pătratic, iar unul de gradul trei este un polinom cubic. Mai rar, polinoamele de gradul patru se numesc cuartice iar cele de grad cinci quintice.

Un polinom cu un singur termen este numit monom, unul cu doi termini binom, iar unul cu trei termini trinom.

Un polinom care are coeficientul 1 pentru termenul de grad maxim se numește monic.

Deoarece scăderea poate fi tratată ca o adunare cu termenul opus, și deoarece ridicarea la o putere întreagă pozitivă și constantă poate fi tratată ca înmulțire repetată, polinoamele se pot construi din constante și variabile folosind doar două operații: adunarea și înmulțirea.

Operatiile ce se pot efectua asupra polinoamelor sunt adunarea, scaderea,inmultirea, impartirea, derivarea si integrarea.Din proprietatile elementare ale polinoamelor stim ca:

1.Suma a doua polinoame este un polinom

2.Produsul a doua polinoame este un polinom

3.Derivarea unui polinom este un polinom

4.Primitiva unui polinom este unu polinom

**Adunarea**

Adunarea se face insumand coeficientii monoamelor cu acelasi grad si pastrandu-se partea literal:

De exemplu: f(x)=ax^2+bx+c

g(x)=dx^2+e

f(x)+g(x)=(a+d)x^2 +bx + c+e

**Inmultirea**

Inmultirea se face inmultind fiecare monom al primului polinom cu fiecare monom al celui de-al doilea polinom dupa regula: (ax)^n(bx)^m=(ab)x^n+m.

De exemplu: f(x)=2x+2

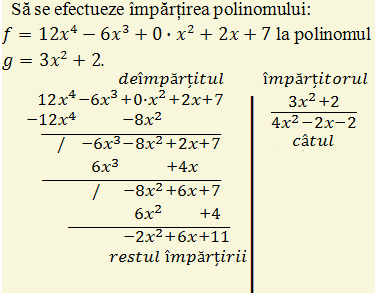
g(x)=x^2+1

f(x)g(x)=(2x+2)(x^2+1)

=2x\*x^2+2x\*1+2\*x^2+2\*1

=2x^3+2x+2x^2+2

=2x^3+2x^2+2x+2



**Impartirea**

Impartirea a 2 polinoame se face dupa urmatorul algoritm:

1.Se ordoneaza polinoamele f si g dupa

puterile descrescatoare ale nedeterminantei x.

2.Se face impartirea polinomului de grad mai

mare(f) la polinomul de grad mai mic.

3.Se imparte primul termen al lui f la primul termen al lui g si se obtine astfel primul termen al catului.

4.Se inmulteste rezultatul astel obtinut cu impartitorul g si se scade acest produs din deimpartitul f=> primul rest al impartirii

5.Se repete procedeul luand primul rest ca deimpartit

6.Algoritmul se termina cand gradul restului este mai mic decat al impartitorului.

Derivarea

Derivarea polinoamelor se face dupa urmatoarea regula de derivare: (x^n)’=n\*x^n-1

si se deriveaza fiecare monom al polinomului.De exemplu:

f(x)= -4x^4-3x^2+5x-3

f(x)’= -16x^3-6x+5

**Integrarea**

Integrarea polinoamelor se face dupa urmatoarea regula de integrare:

De exemplu:

 f(x)=x^3-2x^2+x-2

f(x)dx=x^4/4-2x^3/3+x^2/2-2x

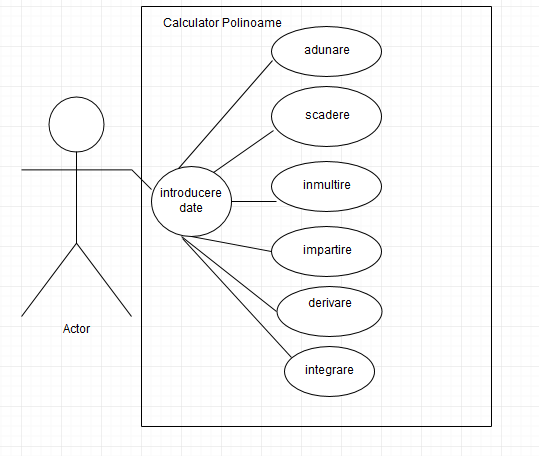
**4.Analiza Cerintei**

In prima faza, de analiza a cerintei ne vom raporta la conceptele programarii orientate pe obiecte care au la baza ideea unificarii datelor cu modalitatile de prelucrarea a acestora si manevreaza enititati reprezentate sub forma de obiecte.Programarea Obiectuala ofera posibilitati de modelarea a obiectelor, a proprietatilor si a realtiilor dintre ele, dar si posibilitatea de a descompune o problema in componentele sale.

Vom studia atent enuntul problemei pentru a putea stabili care sunt obiectele cu care vom lucra mai departe, cum vor interactiona unele cu altele si ce metode pot fi implementate .Astfel putem spune deja ca Polinom va fi o clasa, iar operatiile de adunare, scadere, inmultire, impartire, derivare si integrare vor fi metode.

Mai departe, stim ca un polinom este format din mai multe monoame, deci o alta clasa va fi clasa Monom cu ajutorul careia vor fi implementate operatiile, deoarece acestea se realizeaza asupra monoamelor care formeaza polinomul.

Diagrama UseCase



**5.Proiectare**

**5.1.Structura**

Am implementat 2 clase: Monom si Polinom si am ales ca polinomul sa fie reprezentat printr-un ArrayList pentru a putea avea acces la fiecare monom in parte ce alcatuieste polinomul.

Clasa Monom are 2 variabile de instanta in care se retine coeficientul si puterea fiecarui monom, iar clasa Polinom are o variabila de instanta, un ArrayList<Monom> care va retine o lista de monoame si tot aici sunt implementate metodele care realizeaza operatiile asupra polinoamelor.

**5.2.Diagrama de Clasa**

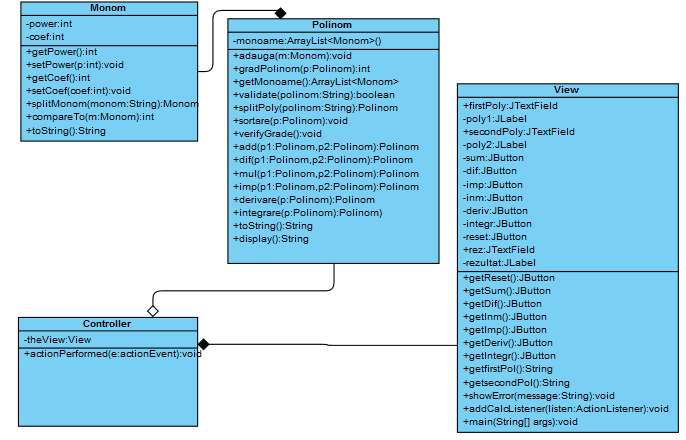
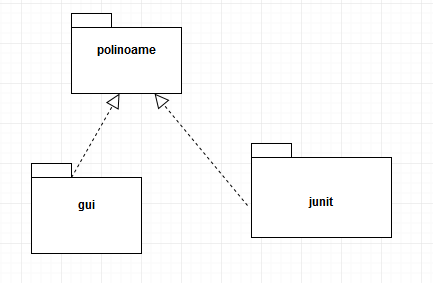
 Diagramele de clase sunt folosite pentru a specifica structura statica a sistemului, adica ce clase exista in sistem si care este legatura dintre ele.

Diagrama de pachete



**6.Implementare**

**Clasa Monom**

In aceasta clasa avem variabilele de instanta: **private int power**; ->exponentul monomului

**private int coef**; ->coeficientul monomului

pentru care am implementat **gettere** si **settere** care returneaza valoarea puterii si coeficientului unui monom, respectiv seteaza o noua valoare pentru putere si coeficient.De asemenea am scris un contructor care creeaza un obiect de tip Monom avand parametri (int power,int coef).

Clasa implementeaza interfata Comparable pentru a putea suprascrie metoda

**int compareTo(Monom m)** care ordoneaza monoamele descrescator dupa valoarea gradului.

In continuare am dezvoltat o metoda care primeste ca argument un String de forma „ax^b”

si il descompune pentru a obtine coeficientul si exponentul si returneaza un obiect de tipul Monom : **public Monom splitMonom(String monom):**

* Pentru extragerea puterii am luat mai multe cazuri:

1. Daca monomul contine caracterul „^” si „x” inseamna ca are o putere>1 si folesec metoda split din clasa String pentru a descompune string-ul dupa caracterul „^” in 2 elemente: „ax” si „b” unde b este puterea, si voi seta puterea monomului.
2. Daca monomul nu contine „^” , deci este de forma „ax” inseamna ca puterea este 1, iar altfel este doar o constanta care nu depinde de x, deci puterea este 0.

**if** (monom.contains("^") && monom.contains("x")) //putere >1

{

String[] el=monom.split("\\^");

**int** i=Integer.*parseInt*(el[1]);

**this**.power=i;

}

**else** **if**(monom.contains("x")) //puterea 1

**this**.power=1;

**else** //termen liber

**this**.power=0;

* Pentru extragerea coeficientului avem urmatoarele cazuri:

1. Daca monomul este “-x” coeficientul este -1.
2. Daca monomul este “x” coeficientul este 1.
3. Daca monomul este de forma “+-ax^b” vom descompune string-ul dupa caracterul “x” in 2 elemente de forma “+-a” si “^b”, iar daca primul caracter este “+” si al doilea “x” coeficientul este 1, daca primul caracter este ori “+” ori “-“ elementul va fi parsat folosind metoda parseInt si va fi setat ca si coefficient , daca elementul este simplu de forma “a” va fi parsat.Mai avem cazul in care monomul este de forma “a”, adica o constanta si atunci va fi parsat pur si simplu si setat ca si coeficient.

**if** (monom.charAt(0)==45 && monom.charAt(1)==120) //-x

{

**this**.coef=-1;

**return** **this**;

}

**if** (monom.charAt(0)==120) //x

{

**this**.coef=1;

}

**else** **if** (monom.charAt(0)!=120 &&monom.contains("x")) // de forma "coef\*x"

{

**int** i = 0;

String[] e=monom.split("x");

**if** (monom.charAt(0)==43 && monom.charAt(1)==120) //+x

**this**.coef=1;

**else** **if** (monom.charAt(0)==45 || monom.charAt(0)==43) //+-

{ i=Integer.*parseInt*(e[0]);

**this**.coef=i;

}

**else**

**this**.coef=Integer.*parseInt*(e[0]);

}

**else /**/monomul este o constanta

{

**int** i=Integer.*parseInt*(monom);

**this**.coef=i;

}

**Clasa Polinom**

In aceasta clasa avem o singura variabila de instanta:  
 **private ArrayList<Monom> monoame;** pentru care avem implementat getter.

Contructorul din acesta clasa este fara parametri si creeaza ArrayList-ul de monoame.

**public** Polinom(){

**super**();

monoame=**new** ArrayList<Monom>();

}

Mai departe am implementat o metoda **int gradPolinom(Polinom p)** care returneaza gradul polinomului, adica gradul maxim al monoamelor, o metoda **boolean validate(String polinom)** care verifica daca polinomul a fost introdus correct folosit Pattern.Am declarat un format care trebuie sa se regaseasca in structura polinomului, iar daca nu va returna false.Si aici, la fel ca si in clasa Monom avem o metoda **Polinom splitPoly(String polinom)** care primeste string-ul de forma ax^n+bx^n-1+… si il va descompune dupa caracterele “-“ si “+” luate inclusive :”+-ax^n”, adica il va descompune in monoame, iar apoi pentru fiecare va fi apelata metoda splitMonom si va fi adaugat in lista de monoame pe care o si returneaza.

String[] monoame1 = **null**;

ArrayList<Monom> mon=**new** ArrayList<Monom>();

monoame1=polinom.split("(?=-)|(?=\\+)");

**for** (String s:monoame1)

{

Monom m=**new** Monom();

mon.add(m.splitMonom(s));

}

**this**.monoame=mon;

**return** **this**;

Metoda **void sortare(Polinom p)** va sorta polinomul in ordinea descrescatoarea a gradelor monoamelor, iar metoda **void verifyGrade()** va aduce polinomul la o forma minima, in cazul in care utilizatorul introduce mai multe monoame cu acelasi grad.

Metodele care implementeaza operatiile:

* **public Polinom add(Polinom p1,Polinom p2)**-consta in faptul ca se parcurg cele 2 polinoame folosind 2 for each iar atunci cand se gasesc monoame cu acelasi exponent se va efectua suma coeficientilor, monomul rezultat fiind adaugat in polinomul care se returneaza.De asemenea, folosec o variabila booleana pentru a verifica daca toate monoamele din primul polinom au fost adaugate in polinomul rezultat in cazul in care nu s-a efectuat adunarea asupra lor.La fel procedez pentru a verifica daca toate monoamele neatinse din al doilea se afla in rezultat.La final polinomul rezultat este sortat si returnat.
* **public Polinom dif(Polinom p1,Polinom p2)-**functioneaza dupa acelasi principiu ca si metoda add doar ca se face diferenta coeficientilor.
* **public Polinom mul(Polinom p1,Polinom p2)**-la operatia de inmultire spre deosebire de adunare si scadere se va face inmultirea fiecarui monom din primul polinom cu fiecare monom din al doilea prin inmultirea coeficientilor si adunarea puterilor.Polinomul rezultat este apoi sortat si minimizat dupa care se returneaza.
* **public Polinom imp(Polinom p1,Polinom p2)**-la impartire se verifica mai intai daca impartitorul este 0 caz in care se va afisa eroare.Se impart monoamele cu grad maxim din primul si din al doilea polinom iar rezultatul se salveaza intr-un polinom rez care mai apoi se inmulteste cu impartitorul si se scade din deimpartit, dar se salveaza si in polinomul cat care va fi returnat.Aceasta diferenta va fi noul deimpartit iar procedeul se repeta pana cand gradul deimpartitului este mai mic decat gradul impartitorului
* **public Polinom derivare(Polinom p)**-la derivare avem 3 cazuri:

1. Daca puterea este 0 atunci monomul este o constanta si derivata este 0
2. Daca puterea este 1 ea va deveni 0
3. Daca puterea>1 coeficientul se va inmulti cu puterea, iar puterea va fi decrementata cu 1

* **public String integrare(Polinom p)-** am ales sa returnez un String pentru aceasta metoda pentru ca afisarea sa fie de forma “a/bx^n” .La integrare coeficientul monomului se va imparti la puterea incrementata cu 1 si totodata puterea va fi incrementata cu 1.

Exemplu pentru implementarea metodelor:

**Public** Polinom derivare(Polinom p)

{Polinom rez=**new** Polinom();

**for** (Monom m:p.getMonoame())

{**if** (m.getPower()==0)

**continue**;

**else** **if**(m.getPower()==1)

m.setPower(0);

**else**

{m.setCoef(m.getCoef()\*m.getPower());

m.setPower(m.getPower()-1);}

rez.monoame.add(m);}

rez.sortare(rez);

**this**.monoame=rez.monoame;

return this;}

**public** Polinom mul(Polinom p1,Polinom p2)

{Polinom rezult=**new** Polinom();

**for**(Monom m1: p1.getMonoame())

**for** (Monom m2:p2.getMonoame())

{

Monom rez=**new** Monom(m1.getPower()+m2.getPower(),m1.getCoef()\*m2.getCoef());

rezult.monoame.add(rez);}

rezult.sortare(rezult);

rezult.verifyGrade();

**this**.monoame=rezult.monoame;

**return** **this**;}

**Implementarea Interfetei Grafice**





Interfata Grafica este implementata in pachetul gui.Fereastra contine 2 TextField-uri in care utilizatorul introduce cele 2 polinoame, dupa care selecteaza operatia dorita, iar rezultatul va fi afisat in TextField-ul cu eticheta Rezultat.Pentru operatiile de derivare si integrare polinomul asupra caruia va fi executata operatia este Polinom1.Metodele care realizeaza operatiile sunt implementate in Controller.java.

7.Testare



Pentru testare am creat un pachet nou junit cu 2 clase.Clasa TestJunit care testeaza metodele din clasa Polinom.Pentru fiecare operatie am dezvoltat o metoda de test care foloseste assertEquals pentru a verifica egalitatea dintre rezultatul returnat de metoda operatiei si rezultatul care se asteapta a fi returnat.

Un exemplu de metoda de test este:

String pol1="3x^2-3x+12";

String pol2="-4x^3-x^2+4x";

Polinom p1=**new** Polinom();

Polinom p2=**new** Polinom();

Polinom rez=**new** Polinom();

@Test

**public** **void** testSum()

{

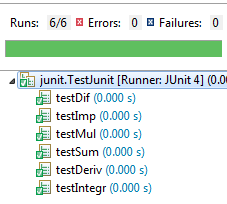
p1.splitPoly(pol1);

p2.splitPoly(pol2);

rez.add(p1, p2);

*assertEquals*("-4x^3+2x^2+x+12",rez.display());

}

Dupa rulare se obtine:

In clasa TestRunner.java am folosit metoda runClasses pentru a rula clasa TestJunit si pentru a obtine rezultatul intr-un obiect de tipul Result, apoi folosind metoda getFailures() am verificat daca a esuat vreun test din clasa TestJunit si in final se afiseaza rezultatul: true daca toate testele au avut succes si false in caz contrar.

**public** **class** TestRunner {

**public** **static** **void** main(String[] args)

{

Result result=JUnitCore.*runClasses*(TestJunit.**class**);

**for** (Failure failure:result.getFailures())

System.***out***.println(failure.toString());

System.***out***.println(result.wasSuccessful());

}

}

Rezultatul afisat este :true.

Pentru testarea unor date introduse prin intermediul interfetei grafice:

P1: 3x^3-2x^2-x+7

P2: x^3+2x-15

Adunare: 4x^3-2x^2+x-8

Scadere: 2x^3-2x^2-3x+22

Inmultire: 3x^6-2x^5+5x^4-42x^3+28x^2+29x-105

Derivare P1: 9x^2-4x-1

Integrare P1: 3/4x^4-2/3x^3-1/2x^2+7x

Impartirea pentru P1: x^3-x^2

P2: x

R: 1x^2-x

8.Concluzii si Dezvoltari Ulterioare

Consider ca aceasta este o aplicatie cu care utilizatorul poate interactiona foarte usor, datorita interfetei care-i ofera posibilitatea de a alege operatia dorita precum si resetarea campurilor.

In ceea ce priveste dezvoltarea ulterioara, aceasta aplicatie de tip calculator poate fi imbunatatita prin modificarea operatiei de impartire pentru a returna si restul, nu doar catul, adaugarea unor operatii noi asupra polinoamelor dar nu numai; se pot adauga si operatii asupra altor tipuri de operanzi.De asemenea poate fi imbunatatita si interfata grafica pentru a o face mai interactiva.Se mai pot aduce modificari si algoritmilor de calcul deoarece consider ca nu sunt cei mai eficienti.

9.Bibliografie

* <https://ro.wikipedia.org/wiki/Polinom>
* <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>
* <https://www.javatpoint.com/java-joptionpane>