TEHNICI DE PROGRAMARE

DOCUMENTATIA TEMEI 1

CALCULATOR DE POLINOAME

Hulea Andrei-Florin
Grupa 30225

Cuprins:

1. Obiectivul temei

Obiectivul principal al temei este proiectarea si implementarea unui calculator de polinoame cu o interfata grafica dedicata in care utilizatorul poate sa introduca polinoame, sa selecteze operatiile care vor fi efectuate asupra acestora (ex: adunare, scadere, inmultire, impartire, derivare, integrare) si sa vizualizeze rezultatul acestora. De asemenea, vom considera polinoamele ca fiind de o singura variabila intreaga si coeficienti intregi.

Operatii care trebuie realizate:

- Citirea unui polinom de la tastatura sub forma de String cu formatul a0X^n+a1X^(n-1)+a2X^(n-2)+...+a0, unde $n \in \mathbb{N}$si $a0, a1, a2 \ldots an \in Z$
- Adunarea a doua polinoame P1(x) + P2(x)
- Scaderea a doua polinoame P1(x) - P2(x)
- Inmultirea a doua plinoame P1(x) * P2(x)
- Impartirea a doua polinoame P1(x) / P2(x)
- Derivarea unui polinom (P1(x))'
- Integrarea unui polinom $\int$ P1(x) dx

Obiectivele secundare sunt:

- Dezvoltarea de use-case-uri
- Alegerea corecta a structurilor de date
- Impartirea pe clase
- Dezvoltarea algoritmilor
- Implementarea solutiilor
- Testarea programului

Se va aplica o abordare orientata pe obiect care sa foloseasca incapsulare si definirea claselor Polinom si Monom.

Alte consideratii ale implementarii:

- Programul va fi scris in limbajul de programare Java
- Pentru implementarea interfetei grafice pentru user se va folosi Java Swing
- Pentru verificarea validitatii polinoamelor se va folosi Regex
- Se vor folosi liste in loc de tablouri bidimensionale

- Se va folosi foreach in loc de for(int i=0...)
- Fiecare clasa trebuie sa aiba sub 300 de linii de cod
- Fiecare metoda trebuie sa aiba sub 30 de linii de cod
- Se vor folosii conventiile Java pentru numele variabelor,claselor,obiectelor,etc.

2. Analiza Problemei

Use case-uri

Cand programul este pornit, utilizatorul va fi prezentat cu o fereastra ce contine mai multe TextField-uri si butoane, numite sugestiv. Acesta poate introduce un polinom in primul din cele doua TextField-uri editabile daca doreste doar sa deriveze sau sa integreze polinomul respectiv sau in ambele daca doreste sa vada suma, diferenta, inmultirea sau impartirea acestora (operatiile vor fi efectuate doar la apasarea butonului respectiv de catre utilizator). Rezultatul operatiilor va fi afisat in TextField-ul de jos in cazul tuturor operatiilor, mai putin in cazul impartirii, in acest caz fiind afisat in TextField-urile Cat si Rest.

De asemenea, este posibil ca utilizatorul sa introduca gresit(astfel sa nu poata fi translatate corect de catre regex) sau sa nu introduca datele necesare intr-un TextField, caz in care acesta este atentionat printr-un mesaj de eroare.

3. Proiectare

În matematică un polinom este o expresie construită dintr-una sau mai multe variabile și constante, folosind doar operații de adunare, scădere, înmulțire și ridicare la putere constantă pozitivă întreagă. Se observă în particular că împărțirea printr-o expresie ce conține o variabilă nu este permisă în polinoame.

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. Un monom fără variabile se numește *monom constant*, sau doar *constantă*. Gradul unui termen constant este 0. Coeficientul unui monom poate fi orice număr, inclusiv fracții, numere iraționale sau negative

Proprietati:

1. Suma a două polinoame este un polinom
2. Produsul a două polinoame este un polinom
3. Derivata unui polinom este un polinom
4. Primitiva unui polinom este un polinom

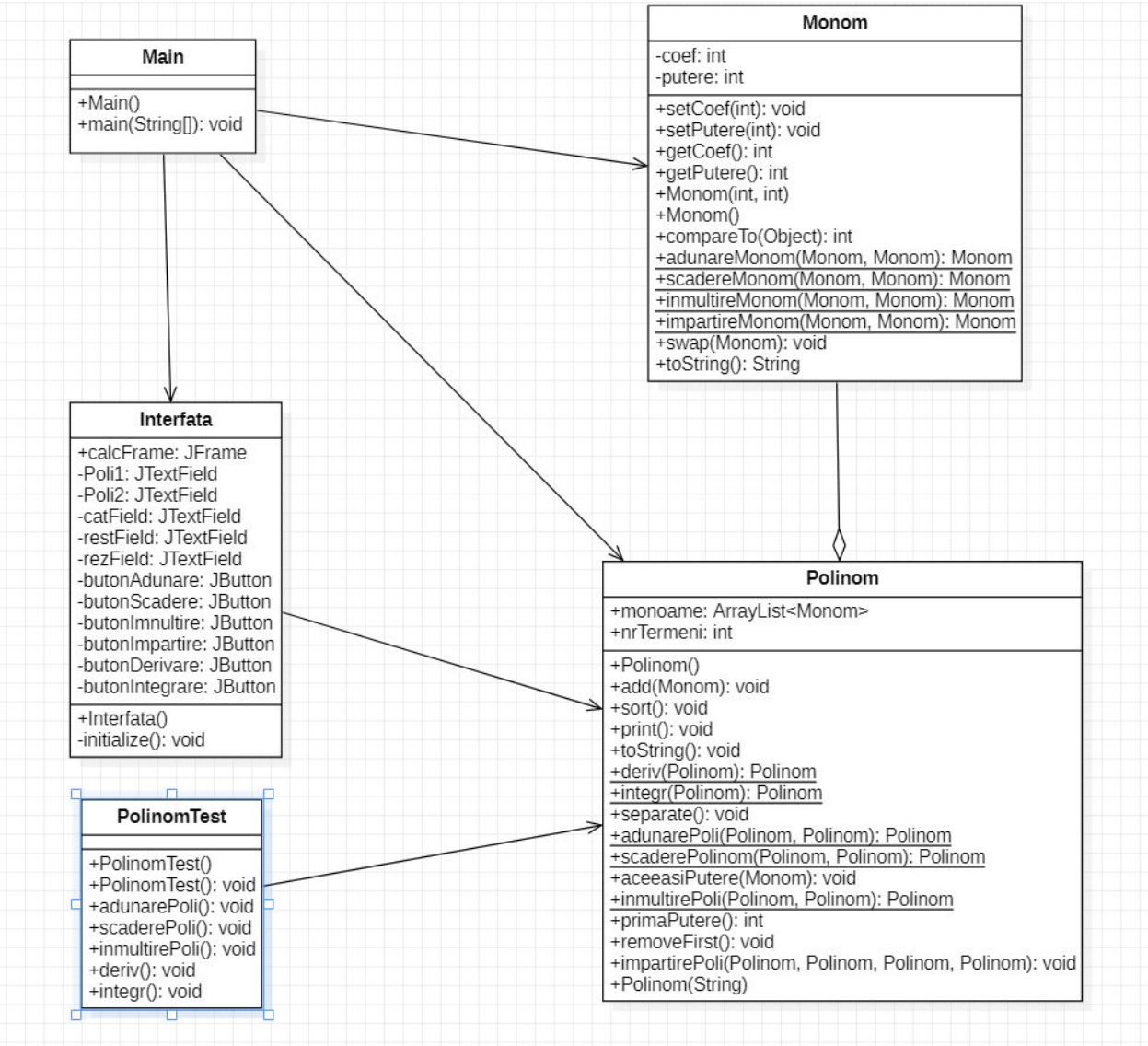
Structuri de date

Structura de date principala pe care am folosit-o ,cu ajutorul careia stocam polinoamele, este un ArrayList. In acesta stocam mai multe monoame(obiecte din clasa Monom ce contin coeficient si putere) care vor fi sortate dupa putere, de la cea mai mare la cea mai mica.

Polinoamele sunt formate din mai multe monoame. Un monom este format dintr-o variabila, un coeficient si un grad. Monomul va fi o clasa separata care va avea ca variabile de instanta acestea, coeficientul si puterea, fiind reprezentate printr-un intreg.

Ex: ArrayList<Monom> poli = new ArrayList<Monom>();

Diagrama de clase



Algoritmi

La adunare,scadere,derivare si integrare, datorita proprietatilor polinoamelor, aceste operatii se pot face termen cu termen, parcurgand polinomul sau polinoamele o singura data.

La impartire vom folosi algoritmul de long division pentru polinoame, psaeudocodul acestuia fiind:

```
degree(P):
  return the index of the last non-zero element of P;
      if all elements are 0, return -∞


polynomial_long_division(N, D) returns (q, r):
  // N, D, q, r are vectors
  if degree(D) < 0 then error
  q ← 0
  while degree(N) ≥ degree(D)
    d ← D shifted right by (degree(N) - degree(D))
    q(degree(N) - degree(D)) ← N(degree(N)) / d(degree(d))
    // by construction, degree(d) = degree(N) of course
    d ← d * q(degree(N) - degree(D))
    N ← N - d
  endwhile
  r ← N
  return (q, r)
```

4. Implementare

Clasa Interfata

In clasa Interfara vom defini componentele interfetei grafice, caracteristicile acestora si actiunile pe care acestea le executa.

Exemplu input corect: 1X^3 + 3X^2 + 2X^0

Daca nu e introdus nici un fel de input, va aparea un mesaj de eroare.

Componentele interfetei grafice sunt urmatoarele:

```
public JFrame calcFrame;
private JTextField Poli1;
private JTextField Poli2;
private JTextField catField;
private JTextField restField;
private JTextField rezField;
private JButton butonAdunare;
private JButton butonScadere;
private JButton butonInmultire;
private JButton butonImpartire;
private JButton butonDerivare;
private JButton butonIntegrare;
```

Fiecarei componente ii definim caracteristicile, dupa care o adaugam in frame, de exemplu:

```
calcFrame = new JFrame();
calcFrame.setTitle("Polynomial Calculator");
calcFrame.setBounds(100, 100, 500, 350);
calcFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
calcFrame.getContentPane().setLayout(null);


rezField = new JTextField();
rezField.setColumns(10);
rezField.setBounds(10,240,460,40);
rezField.setEditable(false);
calcFrame.getContentPane().add(rezField);
```

Actiunile pe care acestea le executa le definim cu ajutorul unui ActionListener

Ex:

```
butonAdunare.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {

      try{
         Polinom aux1 = new Polinom(Poli1.getText());
         Polinom aux2 = new Polinom(Poli2.getText());

         rezField.setText(Polinom.adunarePoli(aux1,aux2).toString());

      }
      catch(Exception e1){
```

```
            System.out.println(e1);
            JOptionPane.showMessageDialog(butonAdunare,"Incorrect input");
        }

    }
});
```

Clasa Monom – este folosita pentru a implementa operatii pe monoame.

Clasa monom va avea ca variabile de instanta puterea si coeficientul unui monom

private int coef;
private int putere;

Inauntrul acestei clase vom crea metode de calcul pentru monoame individuale, toString, si gettere si settere pentru coeficienti si puteri. Aceste clasa cu metodele ei ne vor ajuta sa punem bazele clasei polinom care se va folosi de acestea.

setCoef(int *coef*) – seteaza coeficientul monomului

setPutere(int *putere*) – seteaza puterea monomului

getCoef() – returneaza coeficientul monomului

getPutere() – returneaza puterea monomului

Monom(int *coef*,int *putere*) – creeaza un monom cu coeficientul si puterea dorita

Monom() – creeaza un monom care inca nu are valori

compareTo(Object *obj*) – implementam compareTo pentru a putea ordona in ArrayList-ul din polinom monoamele dupa putere.

Monom adunareMonom(Monom *m1*,Monom *m2*) – executa adunarea a doua monoame

Monom scadereMonom(Monom *m1*,Monom *m2*) – executa scaderea a doua monoame

Monom inmultireMonom(Monom *m1*,Monom *m2*) – executa inmultirea a doua monoame

Monom impartireMonom(Monom *m1*,Monom *m2*) – executa impartirea a doua monoame

public void swap(Monom *m1*) – schima valorea monomului curent cu monomul introdus

public String toString() – returneaza monomul sub forma de string de forma “(coef)X^(putere)”

Clasa Polinom

Inauntrul clasei polinom avem metode pentru cele 6 operatii pe care trebuie sa le putem executa, o metoda de toString pentru scrierea polinoamelor, un constructor pentru crearea polinoamelor fara nici o valoare, un constructor care se foloseste de regex.matcher si regex.pattern pentru a forma un polinom dintr-un string. Majoritatea operatiilor se pot face termen cu termen, deci ne vom folosi de metodele de calcul pentru monoame, parcurgand polinomul. De asemenea, mai avem si functia de add care adauga un monom in polinom, dupa care polinomul este ordonat pentru ca monomul sa fie plasat corect.

private ArrayList<Monom> monoame; -> structura in care stocam monoamele pentru a forma polinomul
private int nrTermeni; -> tine minte numarul de termeni din polinom

public Polinom() – constructor

public void add(Monom *mono*) – adauga un monom in polinom

public void sort() – sorteaza polinomul dupa puteri

public void print() – afiseaza polinomul

public String toString() – returneaza polinomul sub forma de string

public static Polinom deriv(Polinom *p*) – deriveaza polinomul termen cu termen

public static Polinom integr(Polinom *p) – integreaza polinomul termen cu termen*

public static Polinom adunarePoli(Polinom *p1*, Polinom *p2*) – returneaza rezultatul adunariia a doua polinoame (p1 + p2)

public static Polinom scaderePoli(Polinom *p1*,Polinom *p2*) – returneaza rezultatul scaderii a doua polinoame (p1 – p2)

public static Polinom inmultirePoli(Polinom *p1*,Polinom *p2*) – returneaza rezultatul inmultirii a doua polinoame (p1*p2)

public int primaPutere() – returneaza cea mai mare putere din polinom deoarece acestea sunt ordonate descrescator dupa putere

public void removeFirst() – sterge primul monom din polinom

public static void impartirePoli(Polinom *first*, Polinom *second*,Polinom *cat*, Polinom *rest) – returneaza rezultatul impartirii polinomului p1 la p2 in polinoamele cat si rest*

public Polinom (String *input*) – folosing libraria regex, aceasta functie construieste un polinom din string-ul introdus

5. Testare

Cu JUnit putem observa corectitudinea metodelor implementate prin teste. Pentru a verifica daca o operatie e corecta, aplicam metoda toString polinomului rezultat din operatie si inca unui polinom definit de noi care e rezultatul corect, Dupa acestea, vom compara cele 2 stringuri rezultate cu assertEquals, testul avand succes daca cele 2 sunt egale si esuand daca sunt diferite. Mai jos sunt scenariile de teste pentru verificarea corectitudinii adunarii,scaderii,inmultirii,impartirii,derivarii si integrarii:

```
@org.junit.Test
public void adunarePoli() {
   Polinom aux = new Polinom();
   Polinom a1 = new Polinom("+ 1X^2 + 1X^1");
   Polinom a2 = new Polinom ("+ 1X^2 + 2X^1");
   Polinom a3 = Polinom.adunarePoli(a1,a2);
   Polinom a4 = new Polinom("+ 2X^2 + 3X^1");
   assertEquals(a3.toString(),a4.toString());
}

@org.junit.Test
public void scaderePoli(){
   Polinom aux = new Polinom();
   Polinom a1 = new Polinom("+ X^1");
   Polinom a2 = new Polinom("+ X^1");
   Polinom a3 = Polinom.scaderePoli(a1,a2);
   Polinom a4 = new Polinom("+ 0");
   assertEquals(a3.toString(),a4.toString());
}


@org.junit.Test
public void inmultirePoli(){
   Polinom aux = new Polinom();
   Polinom a1 = new Polinom("+ 2X^2");
   Polinom a2 = new Polinom("+ 3X^1");
   Polinom a3 = Polinom.inmultirePoli(a1,a2);
   Polinom a4 = new Polinom("+ 6X^3");
   assertEquals(a3.toString(),a4.toString());
}

@org.junit.Test
public void impartirePoli(){
   Polinom aux = new Polinom();
   Polinom a1 = new Polinom("+ 3X^2 + 1X^1 + 1X^0");
   Polinom a2 = new Polinom("+ 1X^1 + 1X^0");
   Polinom cat1 = new Polinom();
   Polinom rest1 = new Polinom();
   Polinom.impartirePoli(a1,a2,cat1,rest1);
   Polinom cat = new Polinom("+ 3X^1 - 3X^0");
   Polinom rest = new Polinom("+ 3X^0");
}
```

```
@org.junit.Test
public void deriv(){
   Polinom a1 = new Polinom("+ 1X^3 + 2X^2 + 5X^1 + 2X^0");
   Polinom a2 = Polinom.deriv(a1);
   Polinom a3 = new Polinom("+ 3X^2 +4X^1 +5X^0 + 0X^0");
   assertEquals(a3.toString(),a2.toString());
}

@org.junit.Test
public void integr(){
   Polinom a1 = new Polinom("+ 1X^3 + 2X^2 + X^1 + 2X^0");
   Polinom a2 = Polinom.integr(a1);
   Polinom a3 = new Polinom("+ 0X^4 + 1X^3 + 2X^1");
   assertEquals(a3.toString(),a2.toString());
}
```

6. Concluzii si Dezvoltari Ulterioare

In concluzie, calculatorul de polinoame implementat este complet functional si poate fi folosit de oricine detine o cunoasere minimala a polinoamelor si a operatiilor cu acestea. In urma implementarii acestuia am invatat cum se foloseste regex, acesta avand intrebuintari nenumarate.

Algoritmii de adunare, scadere, inmultire si impartire pot fi imbunatatiti marginal, abordand maimulte cazuri speciale sau de nedeterminare decat cele prezente. O alta imbunatatire poate fi aplicata regexului, pentru ca interfata grafica sa accepte o gama mai larga de metode de introducere a polinoamelor.

7. Bibliografie

https://www.wikipedia.org/wiki/Monomial

https://en.wikipedia.org/wiki/Polynomial

https://en.wikipedia.org/wiki/Polynomial_long_division