Tema 1:Sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi

Student:Oprean Andrei

Grupa:30228

1.Obiectivul temei

Obiectivul acestei teme este implementare unui sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi in limbajul de programare Java. Operatiile implementate sunt adunare,scadere,inmultirea,impartirea,derivarea si integrarea polinoamelor. Sistemul trebuie sa ofere o solutie care sa poata fi folosita corect de orice fel dde utilizator indiferent de nivelul de pregatire tehnica.

Utilizarea diferitelor functionalitati ale aplicatiei se face cu ajutorul unei interfete grafice implementate cu ajutorul componentelor Swing.

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

In matematica polinomul este o expresie algebrică constituită din mai multe monoame, legate între ele prin semnul plus sau minus.Analiza problemei se va face folosind elementele programarii orientate pe obiecte,deci vom incerca sa extragem din enuntul problemei substantive si verbe importante care sa ne creeze o baza de pornire (ex: Polinom,aduna,imparte etc.).Urmatorul aspect important este interactiunea aplicatiei cu utilizatorul.Solutia acestei probleme este o interfata grafica usor de utilizat care va cuprinde toate functionalitatile aplicatiei impreuna cu o modalitate de introducere a datelor si una de afisare a acestora.Avand in vedere faptul ca majoritatea operatiilor implementate necesita mai mult de un polinom,va trebui sa avem in vedere o solutie care permite introducerea simultana a datelor de intrare.

Cazurile de utilizare se impart in 2 categorii:

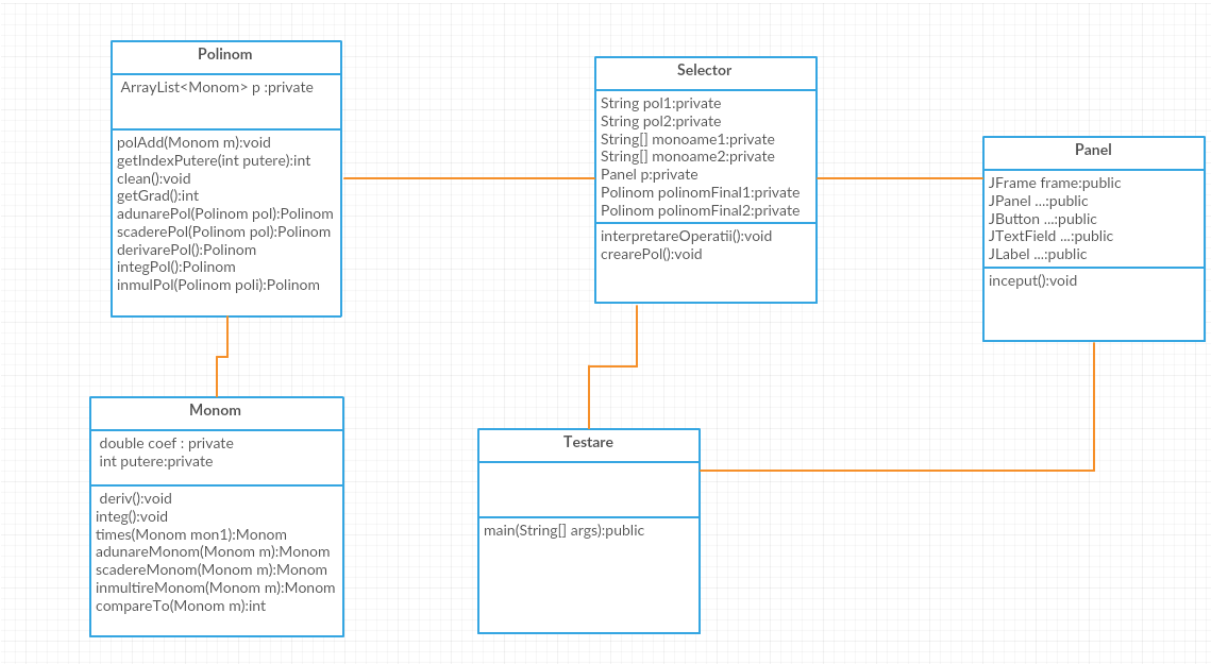
-operatii pe 2 polinoame:adunare,scadere,inmultire,impartire;

-operatii pe 1 polinom:derivare si integrare;

La ambele cazuri rezultatul va fi afisat in casuta de text din josul panoului.

Presupunem ca scenariu universal ca utilizatorul va introduce lista de monoame care compune polinomul intr-un format corect:coeficientx^putere.In cazul in care este introdus intr-un mod diferit,programul arunca o exceptie si utilizatorul poate sa introduca alt polinom sau sa il corecteze pe cel current.Odata introduce datele de intrare,utilizatorul poate selecta cu ajutorul mouse-ului diferite operatii etichetate sugestiv cu semnele matematice corespunzatoare sau in cazul derivarii si integrarii cu numele lor.O alta solutie ar fi fost folosirea prefixelor deriv si integ sau d si i scrise inainte de polinom pentru a indica faptul ca se doreste efectuarea operatiei de integrare sau derivare.

3.Proiectare si implementare



Aplicatia contine 4 clase principale si o clasa de test in care se afla functia main.In fiecare clasa se respecta principiul incapsularii,variabilele de instant fiind private cu exceptia clasei Panel care implementeaza interfata grafica,ea avand o functionare mai speciala.

Clasa Monom:

Are 2 variabile de instanta care descriu coeficientul si gradul/puterea fiecarui monom din polinomul introdus. Clasa aceasta implementeaza interfata Comparable<>,de care vom avea nevoie pentru a sorta lista de monoame in functie de grad.

```
public int compareTo(Monom m){
            if (this.getPutere() == m.getPutere()){
```

```
                return 0;
        }
        else if (this.getPutere() < m.getPutere()){
                return -1;
        }
                else {
                return 1;
                }
    }
```

Se respecta principiul de incapsulare,accesul la aceste variabile facandu-se cu ajutorul getterelor si setterelor.Metodele din aceasta clasa ajuta la manipularea monomului in diferite situatii.De exemplu,metoda deriv() implementeaza derivarea unui monom manipuland coeficientul si puterea.

```
    public void deriv() {
        if (putere != 0) {
                coef = coef * putere;
                putere-=1;
        }
        else if (putere == 0){
        coef = 0;
        }
    }
```

Metoda integ() face operatia de integrare pe un monom dupa regula matematica, modificand mai intai puterea si apoi coeficientul in functie de aceasta.

```
public void integ() {
            putere++;
            coef = coef / putere;
    }
```

Metoda times() implementeaza inmutirea dintre doua monoame,adunand puterile si coeficientii intre ei.Metoda returneaza un obiect de tip Monom care va contine rezultatul acestei inmultiri

```
public Monom times(Monom mon1) {
            Monom ret;

            double c = this.getCoef() * mon1.getCoef();
            int p = this.getPutere() + mon1.getPutere();
            ret = new Monom(c,p);

            return ret;
    }
```

Clasa Polinom:

Are o singura variabila de instant,un ArrayList cu elemente de tip Monom,reusind cu ajutorul acesteia modelarea stringului introdus la intrare intr-o lista de monoame usor manipulabile.Accesul la aceasta variabila se face cu ajutorul metodei getP() .

```
private ArrayList<Monom> p = new ArrayList<Monom>();
```

In clasa Polinom descriem cu ajutorul metodelor operatiile aritmetice.De exemplu adunare se face in metoda adunarePol().Ideea algoritmului este parcurgerea integrala a unui polinom,iar la fiecare element se va verifica daca gradul respective exista si in al 2 lea polinom.Daca da,atunci se vor aduna coeficientii celor 2 impreuna si elemental gasit in polinomul 2 este sters cu metoda remove.Daca nu,se va aduna doar coeficientul primului polinom.La final se parcurge polinomul 2 pentru a aduna elementele ramase.Rezultatul adunarilor se memoreaza in variabila locala aux de tip Polinom,care se va si returna.

```
public Polinom adunarePol(Polinom pol){
		Polinom aux = new Polinom();
		for(int i=0;i<this.p.size();i++) {
			int p1 = this.p.get(i).getPutere();
			double c1 = this.p.get(i).getCoef();

			int ipow = pol.getIndexPutere(p1);
			if(ipow == -1) {

				aux.p.add(new Monom(c1, p1));
			} else {

				int p3 = pol.p.get(ipow).getPutere();
				double c3 = pol.p.get(ipow).getCoef();
				aux.p.add(new Monom(c1+c3, p3));

				pol.p.remove(ipow);
			}
		}

		for(int j=0;j<pol.p.size();j++) {
			int p2 = pol.p.get(j).getPutere();
			double c2 = pol.p.get(j).getCoef();
			aux.p.add(new Monom(c2, p2));
		}
		aux.clean();
	return aux;
	}
```

Scaderea se va face prin modificarea coeficientilor polinomului al 2 lea , inversanduse semnul prin inmultire cu -1.Apoi este apelata metoda de adunare descrisa mai sus,parametrul fiind un nou polinom cu coeficienti “inversati”. Rezultatul se stocheaza in variabila locala aux care este returnata.

```
public Polinom scaderePol(Polinom pol) {
		Polinom aux;

		for(int i=0;i<pol.p.size();i++) {
			pol.p.get(i).setCoef(pol.p.get(i).getCoef()*(-1)) ;
```

```
            }

            aux = this.adunarePol(pol);
            aux.clean();
            return aux;
      }
```

Derivarea si integrarea se fac cu ajutorul metodelor din clasa Monom,care efectueaza aceste operatii pe monoame individuale,deriv () si integ () si returneaza monoamele modificate direct in ArrayList-ul polinomului care apeleaza metoda .

```
public Polinom derivarePol() {
            for(int i=0;i<this.p.size();i++) {
                  this.p.get(i).deriv();
            }
            Return this;
      }

      public Polinom integPol() {
            for(int i=0;i<this.p.size();i++) {
                  this.p.get(i).integ();
            }
            return this;
      }
```

Inmultirea polinoamelor este descrisa de metoda inmulPol() care cu ajutorul metodei times() descrisa mai sus in clasa Monom salveaza in variabila auxiliara de tip Polinom rezultatul inmultirii monoamelor primului polinom cu al celui de al 2 lea. Aceasta variabila locala este si returnata mai apoi.

```
public Polinom inmulPol(Polinom poli) {
            Polinom aux = new Polinom();

            for(int i=0;i<this.p.size();i++) {
                  Monom m1 = this.p.get(i);
                  for(int j=0;j<poli.p.size();j++) {
                        Monom m2 = poli.p.get(j);
                        Monom m3 = m1.times(m2);
                        aux.p.add(m3);
                  }
            }

            return aux;
      }
```

Pe langa aceste metode de baza mai avem diferite metode utile in diferite situatii,cum ar fi metoda clean() care elimina monoamele cu coeficienti 0,imbunatatind estetica iesirii.

```
private void clean() {
            for(int i=0;i<p.size();i++) {
                  if(p.get(i).getCoef() == 0)
                        p.remove(i);
            }
      }
```

Inca o metoda utilizata in clasa Polinom este getIndexPutere() folosita la adunare pentru gasirea monoamelor cu aceeasi putere , caz in care se returneaza valoarea 1 , iar daca nu a fost gasit un monom cu puterea respectiva se va returna valoarea 0.

```
private int getIndexPutere(int putere) {
            for(int i=0;i<p.size();i++) {
                  Monom mon = p.get(i);
                  if(putere == mon.getPutere())
                        return i;
            }
            return -1;
      }
```

Clasa Panel:

Implementeaza interfata grafica,imparte frame-ul in mai multe panouri:pentru intrari,pentru operatii,pentru iesire.Pentru asezare pe ecran am ales un stil de tip GridLayout deoarece pune elementele intr-o ordine logica,de la intrare,la procesare si in final la iesire.

```
frame.setLayout(new GridLayout(4,1));
```

Am ales ca modalitate de introducere si afisare a datelor folosirea campurilor de text,iar pentru operatii,o serie de butoane etichetate sugestiv.

```
public JTextField po1 = new JTextField(40);

public JTextField po2 = new JTextField(40);

plus = new JButton("+");

minus = new JButton("-");

impartire = new JButton("/");

inmultire = new JButton("*");
der = new JButton("Derivare");
integ = new JButton("Integrare");
```

Pentru a elimina anumite ambiguitati am folosit alaturi de fiecare camp text un label care sa specifice datele asteptate de intrare.

```
public JLabel textAfis1 = new JLabel("Rezultatul operatiei");

public JLabel p1 = new JLabel("Polinomul 1");
public JLabel p2 = new JLabel("Polinomul 2");
```

Marimea aleasa de mine pentru fereastra principala este de 600x400 deoarece ofera un spatiu de ajuns in cazul in care utilizatorul introduce un polinom cu multe monoame.

```
frame.setSize(600,400);
```

Clasa Selector:

Este clasa cea mai importanta din aplicatie deoarece in interiorul ei declar ascultatorii butoanelor,care simultan citesc datele de intrare,efectueaza operatii pe ele si afiseaza rezultatul.

```
p.plus.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                  p.textAfis.setText("");
                  pol1 = p.po1.getText();
                  pol2 = p.po2.getText();

                  crearePol();
                  Polinom rAdun = polinomFinal1.adunarePol(polinomFinal2);
                  Collections.reverse(rAdun.getP());

                  for (Monom f : rAdun.getP()){
                        if (f.getCoef() > 0){
                              p.textAfis.setText(p.textAfis.getText()+ "+"
+String.valueOf(f.getCoef())+ "x^" + String.valueOf(f.getPutere()) + " ");
                              }
                              else{
                                    p.textAfis.setText(p.textAfis.getText()
+String.valueOf(f.getCoef())+ "x^" + String.valueOf(f.getPutere()) + " ");
                              }
                  }

            }
      });
```

Procesarea datelor de intrare se face in metoda crearePol(),care manipuleaza stringurile de intrare si le transforma prin instructiuni succesive in blocuri de date care pot fi introduse in sirul de monoame din clasa polinom.Am ales sa citesc,procesez si afisez date in aceleasi metode din cauza faptului ca generaza un cod final mai usor de citit si inteles.Pentru a putea procesa monoamele cu semn negative intr-un numar mic de linii de cod am inlocuit cu ajutorul metodei replace() toate semnele “+” cu semne “+-“.

```
pol1 = pol1.replace("-", "+-");
```

Astfel ca la inca o apelare a metodei split() avem coeficienti fara semn la cei pozitivi si coeficienti precedati de “-“ la cei negativi.

```
pol1 = pol1.replaceAll("\\s" ,"");
            pol2 = pol2.replaceAll("\\s" ,"");

            int i ;
            String[] buff = null;
            String[] buff1 = null;

            monoame1 = pol1.split("[+]");
            monoame2 = pol2.split("[+]");

            int f = monoame1.length;
            int f1 = monoame2.length;

            for(i = 0;i < f;i++){
```

```
        monoame1[i] = monoame1[i].replaceAll("\\^", "");

        buff = monoame1[i].split("x");
        buff[1] = buff[1].replaceAll("\\^", "");

        polinomFinal1.polAdd(new
        Monom((Double.parseDouble(buff[0])),Integer.parseInt(buff[1])));
```

3.1. Testare

Testarea va fi unitara,adica voi testa individual metodele pentru operatii aritmetice din clasa Polinom cu ajutorul uneltei de testare JUnit . Voi testa respectivele metode doar verificand daca in urma operatiei rezultatul nu mai este NULL deoarece rezultatul propriu-zis se poate vedea la iesirea din interfata grafica.

```
@Test
public void testAdun() {
        pol1.polAdd(new Monom(5, 2));
        pol1.polAdd(new Monom(3, 1));
        pol1.polAdd(new Monom(-4, 0));
        pol2.polAdd(new Monom(7, 4));
        pol2.polAdd(new Monom(-2, 2));
        pol2.polAdd(new Monom(4, 1));
        pol2.polAdd(new Monom(8, 0));
        Polinom rez = pol1.adunarePol(pol2);
        assertNotNull(rez);
}


@Test
public void testScad() {
        pol1.polAdd(new Monom(5, 2));
        pol1.polAdd(new Monom(3, 1));
        pol1.polAdd(new Monom(-4, 0));
        pol2.polAdd(new Monom(7, 4));
        pol2.polAdd(new Monom(-2, 2));
        pol2.polAdd(new Monom(4, 1));
        pol2.polAdd(new Monom(8, 0));
        Polinom rez = pol1.scaderePol(pol2);
        assertNotNull(rez);
}

@Test
public void testInmul() {
        pol1.polAdd(new Monom(5, 2));
        pol1.polAdd(new Monom(3, 1));
        pol1.polAdd(new Monom(-4, 0));
        pol2.polAdd(new Monom(7, 4));
        pol2.polAdd(new Monom(-2, 2));
        pol2.polAdd(new Monom(4, 1));
        pol2.polAdd(new Monom(8, 0));
        Polinom rez = pol1.inmulPol(pol2);
        assertNotNull(rez);
}

@Test
public void testDeriv() {
```

```
        pol1.polAdd(new Monom(5, 2));
        pol1.polAdd(new Monom(3, 1));
        pol1.polAdd(new Monom(-4, 0));
        Polinom rez = pol1.derivarePol();
        assertNotNull(rez);
    }

    @Test
    public void testInteg() {
        pol1.polAdd(new Monom(5, 2));
        pol1.polAdd(new Monom(3, 1));
        pol1.polAdd(new Monom(-4, 0));
        Polinom rez = pol1.integPol();
        assertNotNull(rez);
    }
```

4.Rezultate

Ca rezultate am obtinut un calculator de polinoame de o singura variabila , cu exactitate mai mare ca in cerinta initiala , modificand tipul coeficientului din intreg in double astfel permitand efectuarea cu precizie a operatiilor de integrare , inmultire si impartire.Datele de iesire sunt aranjate intr-o ordine intuitiva , in ordine descrescatoare in functie de gradul monomului. Eliminand monoamele cu coeficienti nuli din ArrayList obtinem o estetica placuta a iesirii in limita oferita de timp si de swing. Prin citirea datelor de intrare la fiecare eveniment de buton ne asiguram ca utilizatorul va primi la iesire rezultatul asteptat indiferent de schimbarile aduse datelor de intrare pe parcursul executiei programului. Tot odata prin comprimarea iesirilor intr-un singur camp de text care se updateaza in functie de operatia selectata cream un design minimalist , usor de inteles indiferent de cunostintele tehnice ale utilizatorului.

5. Concluzii,dezvoltari ulterioare

In concluzie, nu as putea spune ca abordarea mea este cea mai corecta, fiind loc de imbunatatiri evidente , in principal separarea clasei Selector in 2 clase , una pentru interpretarea si crearea polinomului propriu-zis si alta pentru a lega operatiile de butoanele respective din interfata grafica.Cu toate acestea , aplicatia este functionala si dispune de o anumita libertate in introducerea datelor , ordinea monoamelor neefind importanta. Pe langa acestea , datele de iesire sunt relative usor de inteles si interpretat astfel ca am putea considera implementarea aleasa de mine un relativ succes.

Ca dezvoltari ulterioare , pentru a fi o aplicatie cu adevarat user-friendly sunt nevoie de imbunatatiri mari la capitolul interfata cu utilizatorul si o flexibilitate mai mare a modului de introducere a datelor de intrare. Ca imbunatatiri posibile s-ar putea observa nevoia de a calcula valoarea polinomului intr-un punct x dat si eventual generarea unui grafic pe baza datelor de intrare.

Incercand sa implementez o solutie la aceasta tema am aflat importanta modelarii de la inceput a problemei bazandu-te pe principiile programarii orientate pe obiecte. Impartind diferitele functionalitati ale aplicatiei in clasele si metodele corecte , ajungem la o solutie eficienta , lizibila si usor de depanat si mentinut.

6.Bibliografie

http://www.mkyong.com/tutorials/junit-tutorials/

http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

http://users.utcluj.ro/~igiosan/teaching_poo.html

https://google.github.io/styleguide/javaguide.html