DOCUMENTATIE TEMA 1

CALCULATOR DE POLINOAME

Nume Student: Buzilă-Gârda Andra-Maria
Grupa: 30229
Profesor de laborator: assist. Antal Marcel

Cuprins:

1. Cerinte functionale:

- Creati un calculator de polinoame utilizand interfata grafica prin care utilizatorul poate sa introduca polinoame, sa selecteze operatia matematica pe care calculatorul sa o efectueze intre cele 2 polinoame(suma, diferenta, inmultire, impartire, derivare, integrare), iar apoi sa afiseze rezultatul/rezultatele.
- Pentru realizarea acestui calculator va trebui respectati urmatoarele reguli:
    - Utilizati un limbaj de programare orientat pe obiecte(folositi incapsularea si impartiti problema in clase cu nume pertinente);
    - Utilizati liste in loc de siruri;
    - Utilizati „foreach” in loc de „for(int i=0..)”;
    - Implementati o interfata grafica utilizand Java Swing sau JavaFX;
    - Implementati clase cu maxim 300 linii(exceptand clasele interfatei cu utilizatorul) si metode cu cel mult 30 linii;
    - Folositi nume pertinente conform conventiei din java;
    - Folositi arhitectura Model-View-Controller;
    - Implementati operatia de adunare;
    - Implementati operatia de scadere;
    - Implementati operatia de inmultire;
    - Implementati operatia de impartire;
    - Implementati operatia de derivare;
    - Implementati operatia de integrare;
    - Folositi regular expressions si pattern matching atunci cand extrageti coeficientii;
    - Folositi Junit pentru testare;
    - Redactarea documentatiei, continand minim 2000 de cuvinte.

2. Obiective

2.1. Obiectiv principal:

Proiectarea si implementarea unui sistem de procesare a polinoamelor de o singura variabila si cu coeficienti reali.

2.2. Obiective secundare:

Obiectiv	Descriere	Capitol
Dezvoltarea de use case-uri si scenarii	Dorim sa cunoastem ce se intampla in momentul apasarii fiecarui buton	3
Alegerea structurilor de date	Modul de reprezentare in memorie	4
Impartirea problemei pe clase	Cream clasele de care avem nevoie	4
Implementarea solutiilor	Modul in care am creat fiecare clasa si fiecare metoda in parte	5
Testare	Verificarea functionalitatii	6

3. Analiza problemei

Utilizatori:

- Elevi;
- Profesori;
- Contabili;
- Programator;
- Administrator de retea.

Pre-conditii:

- Utilizatorul a introdus primul polinom in formatul cerut;
- Utilizatorul a introdus al doilea polinom in formatul cerut.

Post-conditii:

- Se realizeaza suma celor 2 polinoame;
- Se realizeaza diferenta celor doua polinoame;
- Se realizeaza produsul dintre cele 2 polinoame;
- Se deriveaza primul polinom;
- Se integreaza primul polinom.

Mod de functionare:

1. Utilizatorul introduce de la tastatura polinoamele asupra carora doreste sa se efectueze operatiile dupa formatul precizat inaintea acelor textField-uri pentru polinoame;
2. Cazuri:
    - 2.1. Cazul 1:
        - 2.1.1. Utilizatorul a apasat pe butonul „Adunare” ;
        - 2.1.2. Se afiseaza rezultatul adunarii polinoamelor in textField-ul pentru rezultat.
    - 2.2. Cazul 2:
        - 2.2.1. Utilizatorul a apasat pe butonul „Scadere” ;
        - 2.2.2. Se afiseaza rezultatul scaderii polinoamelor in textField-ul pentru rezultat.
    - 2.3. Cazul3:
        - 2.3.1. Utilizatorul a apasat pe butonul „Impartire”;
        - 2.3.2. Se afiseaza catul impartirii polinoamelor in primul textField din rezultat si restul impartirii, in cel de-al doilea.
    - 2.4. Cazul 4:
        - 2.4.1. Utilizatorul a apasat pe butonul „Inmultire” ;

2.4.2. Se afiseaza rezultatul inmultirii polinoamelor in textField-ul pentru rezultat.

2.5. Cazul 5:

2.5.1. Utilizatorul a apasat pe butonul „Derivare” ;

2.5.2. Se afiseaza rezultatul derivarii primului polinom in textField-ul pentru rezultat.

2.6. Cazul 6:

2.6.1. Utilizatorul a apasat pe butonul „Integrare” ;

2.6.2. Se afiseaza rezultatul derivarii primului polinom in textField-ul pentru rezultat.

3. In momentul in care utilizatorul apasa butonul clear, textField-urile pentru polinomul1, polinomul2 si rezultat se golesc.

! In cazul in care utilizatorul a introdus gresit vreunul dintre polinoame sau chiar amandoua, va fi atentionat printr-o exceptie afisata in consola.

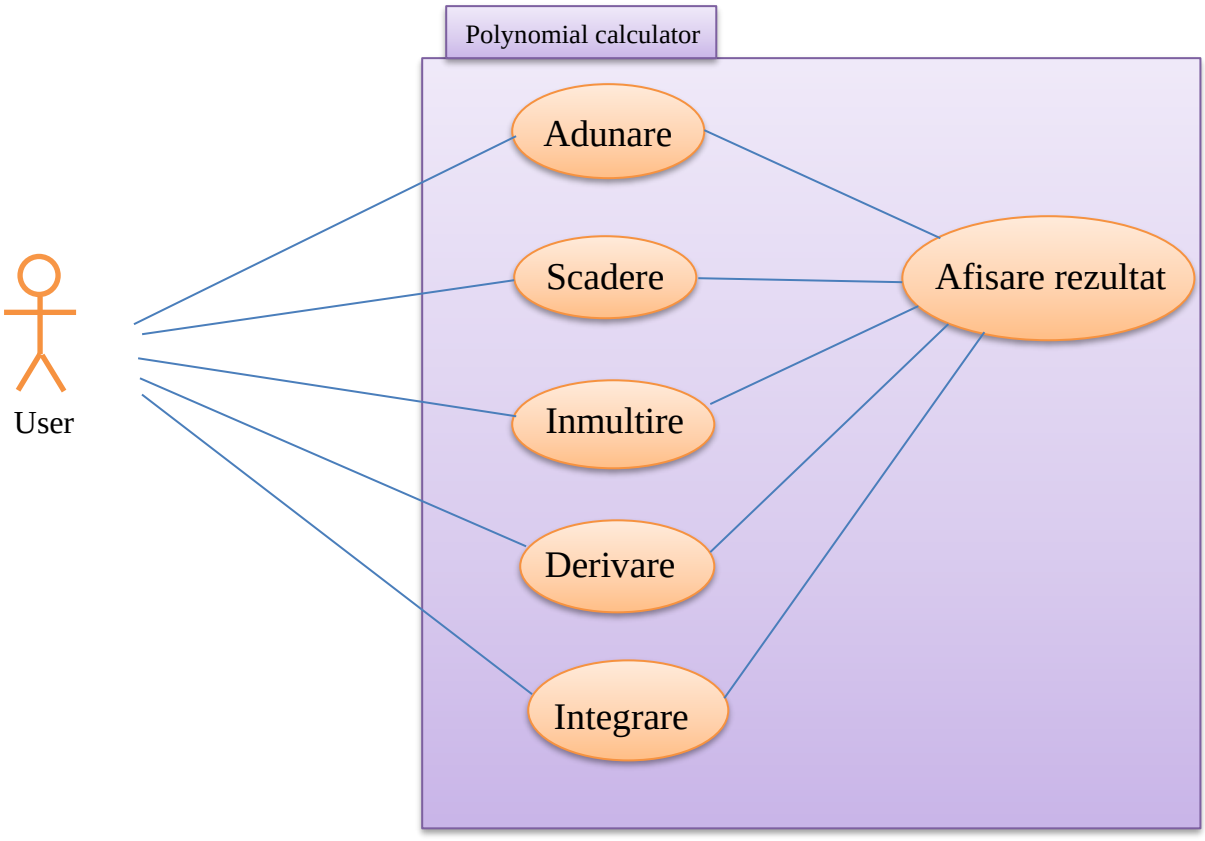


Figura 1: REPREZENTARE DIAGRAMA USE CASE

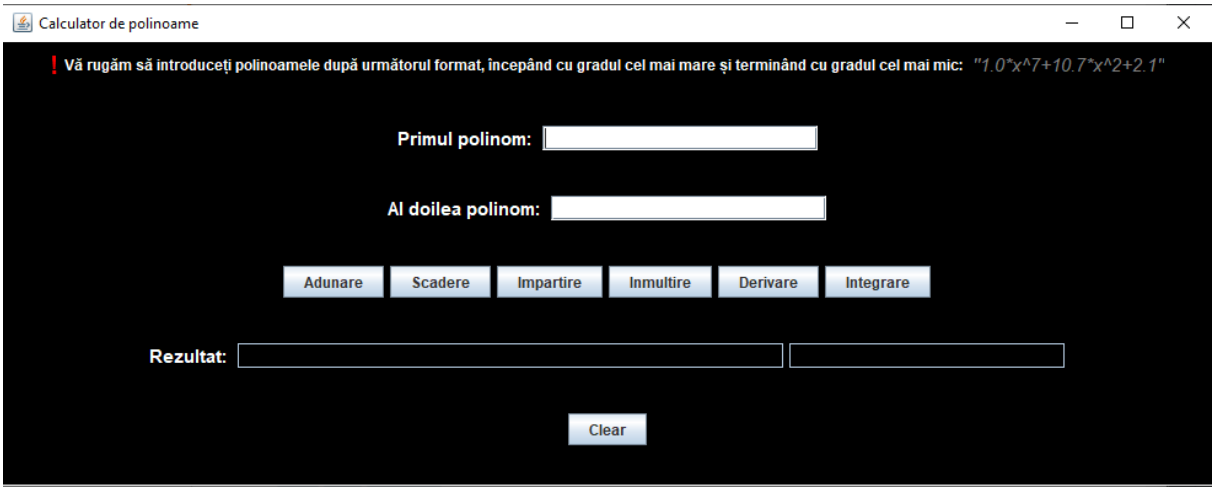


Figura 2: CALCULATORUL VAZUT DE CATRE UTILIZATOR

4. Proiectare

4.1. Alegerea structurilor de date

Polinomul am ales sa il reprezint ca si o lista de monoame, fiecare monom continand un intreg pentru puterea variabilei x si un numar real pentu coeficientul acesteia.

4.2. Diagrama de clase

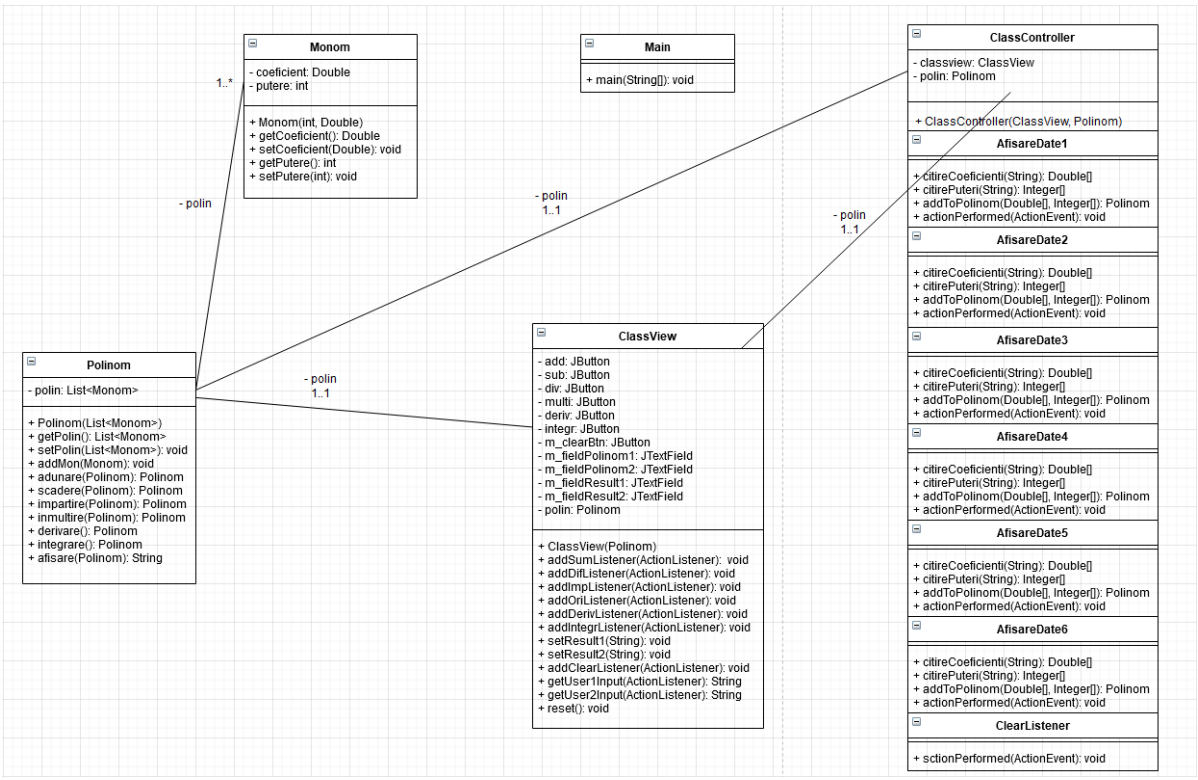


Figura 3: DIAGRAMA UML

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele și specificatii pentru software. UML ofera o larga gama de diagrame pentru modelarea diferitelor situatii in cadrul unui proiect de dezvoltare software.

Diagrama de clasa este folosita pentru reprezentarea vizuala a claselor ai a interdependentelor, taxonomiei si a relatiilor de multiplicitate dintre ele. Aceste diagrame sunt folosite si pentru reprezentarea concreta a unor instante de clasa, asadar obiecte, si a legaturilor concrete dintre acestea.

Am creat aceasta aplicatie luandu-ma dupa structura Model-View-Controller(MVC) si anume: am facut clasa ClassView in care am schitat imaginea de ansamblu a calculatorului, cu butoane, textField-uri, label-uri, apoi am construit clasa ClassController, unde am proiectat functionarea calculatorului, iar Modelul este reprezentat de clasele Monom si Polinom si, de asemenea, am creat si clasa main.

5. Implementare

CLASA MONOM

Pentru inceput am creat clasa Monom. Aceasta clasa contine 2 variabile instanta: putere si coeficient, numere care insotesc variabila x din polinom. Pentru aceste variabile instanta am generat metodele getters si setters, de care ne vom folosi cand vom dori sa le invocam dintr-o alta clasa. Am mai creat o metoda, compareTo(Monom m), care returneaza diferenta de puteri a 2 polinoame, metoda pe care o voi folosi cand sortez monoamele pentru a le afisa, cu ajutorul functiei Collection.sort() care sorteaza o lista.

```
import java.util.*;

public class Monom implements Comparable<Monom>{
    private int putere;
    private Double coeficient;

    public Monom(int p, Double c){
        this.putere = p;
        this.coeficient = c;
    }

    public int getPutere() {
        return putere;
    }

    public void setPutere(int putere) {
        this.putere = putere;
    }

    public Double getCoeficient() {
        return coeficient;
    }

    public void setCoeficient(Double coeficient) {
        this.coeficient = coeficient;
    }

    public int compareTo(Monom m) { return this.getPutere()-m.getPutere(); }


}
```

CLASA POLINOM

Am construit apoi clasa Polinom in care am luat o lista de monoame, avand in vedere ca si in matematica polinomul este alcatuit din unul sau mai multe monoame. Am generat metoda getter si metoda setter pentru aceasta lista in aceeasi clasa. Tot aici am implementat metoda de adaugare de monom in polinom (addMon(Monom m)), metoda in care pur si simplu am adaugat un monom in lista de polinoame a polinomului curent. Tot in clasa polinom am implementat operatiile de adunare de polinoame (adunare (Polinom p)), scadere de polinoame (scadere (Polinom p)), impartire de polinoame (impartire (Polinomp)), inmultire de polinoame (inmultire (Polinom p)), derivare (derivare()) si integrare (integrare()) de polinom. Toate aceste metode returneaza un polinom ca rezultat al respectivei operatii. Ultima metoda implementata in clasa Polinom este metoda de afisare, care, pe langa faptul ca afiseaza la consola polinomul rezultat, mai returneaza un string, care este tot polinomul rezultat, care va fi afisat in textField-ul pentru rezultat. In continuare am sa descriu metodele implementate pentru operatii:

- In metoda de adunare am creat o instanta de polinom, care este de fapt polinomul rezultat, de asemenea am creat si o lista de monoame care se vor repeta in polinomul rezultat in urma algoritmului, pe care le-am eliminat in final. Am parcurs monoamele din lista de monoame a polinomul trimis ca parametru in functie si puterea fiecaruia am comparat-o cu puterea fiecarui monom din lista de monoame a polinomului curent. In cazul in care am gasit monoame cu puteri egale, le-am adunat coeficientii si am creat un nou monom, avand coeficientul rezultat si puterea monoamelor gasite. Pe acest monom l-am adaugat intr-un polinom nou, denumit generic: egal, acesta continand doar astfel de monoame. In polinomul pe care il voi returna am adaugat atat monoamele polinomului trimis ca parametru cat si monoamele polinomului curent. Am comparat apoi puterile monoamelor din polinomul rezultat cu puterile monoamelor din polinomul egal deoarece in polinomul rezultat pot exista monoame cu aceleasi puteri, iar acestea cu siguranta se vor afla in polinomul egal. Daca am gasit astfel de monoame in polinomul rezultat le-am adaugat in lista de monoame pe care le vom sterge. In final am adaugat si monoamele din egal in rezultat si apoi am sters tot ceea ce trebuia sters. Am ordonat apoi monoamele cu functia Collection.sort.
- Metoda scadere am implementat-o in acelasi mod ca si metoda adunare, doar ca in momentul in care am gasit moname cu aceleasi puteri in cele doua polinoame le-am scazut coeficientii, iar daca am gasit vreun monom avand coeficientul 0.0 l-am adaugat in lista de monoame de eliminat la final. De asemenea, atunci cand am adaugat monoamele polinomului trimis ca parametru in polinomul rezultat, le-am schimbat coeficientul din +coeficient in -coeficient si astfel l-am adaugat in lista. Dupa aceea am urmat pasii de la metoda adunare pana in final. Cand am scazut coeficientii, am folosit un format pentru rezultat astfel incat rezultatul sa fie afisat doar cu o zecimala.
- Metoda de impartire va returna o lista de 2 polinoame alcatuita din cat reprezentat printr-un polinom si restul, tot un polinom. Pentru inceput am creat un polinomul rest pe care l-am egalat cu polinomul curent. Am comparat puterea termenului cel mai mare din polinomul rest cu puterea termenului cel mai mare din polinomul la care se face impartirea si cat timp acesta este mai mare sau egal, se genereaza pe rand monoamele polinomului cat in felul urmator: fiecare monom din acest polinom va avea puterea egala cu diferenta de puteri dintre cele 2 polinoame: rest si polinomul trimis ca parametru, cel la care se face impartirea, iar coeficientul egal cu coeficientul restului (acest algoritm porneste de la polinomul curent deoarece am egalat restul cu acesta), iar pe urma se inmulteste monomul curent din cat cu polinomul la care se face impartirea si acest rezultat se va scadea din rest. Polinomul rest se va sorta descrescator dupa eliminarea din el, astfel incat termenul cu cea mai mare putere sa fie mereu primul. Bucla se va incheia atunci cand practic gradul polinomului rest va fi mai mic decat gradul polinomului impartitor. Am sortat polinomul rest si polinomul cat ascendent dupa putere si le-am adaufat in lista de polinoame pe care am returnat-o.
- In metoda inmultire am „inmultit" fiecare monom din polinomul curent cu fiecare monom din polinomul trimis ca parametru, adica le-am adunat puterile si le-am inmultit coeficientii (am folosit si aici un tip de format care imi permite sa vad doar 2 zecimale). Pe urma am verificat daca in polinomul rezultat exista monoame care au aceeasi putere. In cazul in care am gasit astfel de monoame le-am adunat coeficientii, am pastrat puterea si le-am adaugat intr-un alt monom pe care l-am pus in rezultat. Monoamele anterior gasite le-am adaugat in lista cu monoamele ce trebuie eliminate.
- In metoda derivare am luat din nou un polinom pe care o sa il returnez la final. Acestui polinom i-am atribuit valoarea polinomului curent. Am luat pe rand fiecare monom din polinom si i-am setat coeficientul la coeficient*putere si puterea i-am setat-o la putere-1. Am folosit, din nou, un decimal format pentru a se afisa doar 1 zecimala. Daca am gasit monom care are coeficientul 0.0 l-am eliminat in polinom.
- Metoda integrare am implementat-o in acelasi mod ca si cea precedenta, doar ca atunci cand am setat coeficientul i-am atribuit valoarea coeficient/putere, iar puterea am setat-o la putere+1 deoarece asa se procedeaza si atunci cand se face integrarea matematica. (am folosit acelasi decimal format pentru coeficienti).

```java
import java.text.DecimalFormat;
import java.util.*;

public class Polinom {
    private List<Monom> polin=new ArrayList();

    public Polinom(List<Monom> poli){ this.polin = poli; }

    public List<Monom> getPolin() { return polin; }

    public void setPolin(List<Monom> polin) { this.polin = polin; }

    public void addMon(Monom m){
            this.polin.add(m);}

    public Polinom adunare(Polinom p) {
        Polinom result = new Polinom(new ArrayList());
        List<Monom> toRemove1 = new ArrayList();
        Polinom egal = new Polinom(new ArrayList());
        for (Monom m1 : p.getPolin())
            for(Monom m2 : polin)
                if(m1.getPutere() == m2.getPutere()){
                Double coef = m1.getCoeficient() + m2.getCoeficient();
                Monom m = new Monom(m1.getPutere(), coef);
                egal.addMon(m);}

        for(Monom m1 : polin)
            result.getPolin().add(m1);
        for(Monom m2 : p.getPolin())
            result.getPolin().add(m2);

        for(Monom m1 : egal.getPolin())
            for(Monom m2: result.getPolin())
                if(m1.getPutere() == m2.getPutere())
                    toRemove1.add(m2);
        result.getPolin().removeAll(toRemove1);
        result.getPolin().addAll(egal.getPolin());
        Collections.sort(result.getPolin());
        return result;
    }

    public Polinom scadere(Polinom p) {
        Polinom result = new Polinom(new ArrayList());
        List<Monom> toRemove1 = new ArrayList(), toRemove2 = new ArrayList();
        Polinom egal = new Polinom(new ArrayList());
        for (Monom m1 : p.getPolin())
            for(Monom m2 : polin)
                if(m1.getPutere() == m2.getPutere()){
                    DecimalFormat df = new DecimalFormat( pattern: "#.##");
                    Double coef = Double.valueOf(df.format( number: m2.getCoeficient() - m1.getCoeficient()));
                    Monom m = new Monom(m1.getPutere(), coef);
                    egal.addMon(m);
                    if(coef == 0.0)
                        toRemove1.add(m);}
        for(Monom m1 : polin)
            result.getPolin().add(m1);
        for(Monom m2 : p.getPolin()){
            Monom m3 = new Monom(m2.getPutere(),-m2.getCoeficient());
            result.getPolin().add(m3); }
        for(Monom m1 : egal.getPolin())
            for(Monom m2: result.getPolin())
                if(m1.getPutere() == m2.getPutere())
                    toRemove2.add(m2);
        result.getPolin().removeAll(toRemove2);
        result.getPolin().addAll(egal.getPolin());
        result.getPolin().removeAll(toRemove1);
        Collections.sort(result.getPolin());
        return result;
```

```
69      }
70
71  @   public List<Polinom> impartire(Polinom p){
72          Polinom resultCat = new Polinom(new ArrayList());
73          List<Polinom>  result = new ArrayList<~>();
74          Polinom p1 = new Polinom(new ArrayList());
75          Polinom inmul = new Polinom(new ArrayList());
76          Polinom asta = new Polinom(new ArrayList());
77          asta = this;
78          while(asta.getPolin().get(0).compareTo(p.getPolin().get(0))>=0){
79              Monom mon = new Monom( p: asta.getPolin().get(0).getPutere()-p.getPolin().get(0).getPutere(), asta.getPolin().get(0).getCoeficient());
80              resultCat.addMon(mon);
81              p1.addMon(mon);
82              inmul = p.inmultire(p1);
83              p1.getPolin().remove(mon);
84              asta=asta.scadere(inmul);
85              Collections.sort(asta.getPolin(), Collections.reverseOrder());
86          }
87          Collections.sort(resultCat.getPolin());
88          result.add(resultCat);
89          result.add(asta);
90          return result;
91      }
92
93      public Polinom inmultire(Polinom p){
94          int i=0,cnt[] = new int[100];
95          DecimalFormat df = new DecimalFormat( pattern: "#.##");
96          Polinom result = new Polinom(new ArrayList());
97          List<Monom> toRemove = new ArrayList();
98          for(Monom m1:polin)
99              for(Monom m2:p.getPolin()){
100                 Monom m3 = new Monom( p: m1.getPutere()+m2.getPutere(), Double.valueOf(df.format( number: m1.getCoeficient()*m2.getCoeficient())));
101                 result.addMon(m3);
102             }
```

```
103         for(Monom m1: result.getPolin()){
104             for(Monom m2: result.getPolin()) {
105                 if (m1.getPutere() == m2.getPutere())
106                     cnt[i]++;
107                 if (cnt[i] >= 2) {
108                     m1.setCoeficient(Double.valueOf(df.format( number: m1.getCoeficient() + m2.getCoeficient())));
109                     cnt[i]--;
110                     toRemove.add(m2);
111                 }
112             }
113             i++;}
114         result.getPolin().removeAll(toRemove);
115         Collections.sort(result.getPolin());
116         return result;
117     }
118
119     public Polinom derivare(){
120         Polinom result = this;
121         List<Monom> toRemove = new ArrayList();
122         for(Monom m: result.getPolin()){
123             DecimalFormat df = new DecimalFormat( pattern: "#.##");
124             m.setCoeficient(Double.valueOf(df.format( number: m.getCoeficient()*m.getPutere())));
125             m.setPutere(m.getPutere()-1);
126         }
127         for(Monom m: result.getPolin()){
128             if(m.getCoeficient() == 0.0){
129                 toRemove.add(m);
130             }
131         }
132         result.getPolin().removeAll(toRemove);
133         Collections.sort(result.getPolin());
134         return result;
135     }
136
```

```
137     public Polinom integrare(){
138         Polinom result = this;
139         for(Monom m: result.getPolin()){
140             DecimalFormat df = new DecimalFormat( pattern: "#.##");
141             m.setCoeficient(Double.valueOf(df.format( number: m.getCoeficient()/(m.getPutere()+1))));
142             m.setPutere(m.getPutere()+1);
143         }
144         Collections.sort(result.getPolin());
145         return result;
146     }
147
148 @   public String afisare(Polinom p){
149         String s ="";
150         for(Monom m : p.getPolin()){
151             if(m.getCoeficient()>0){
152             s=s+ "+"+m.getCoeficient()+"*x^"+m.getPutere()+"";}
153             else
154                 s=s+ m.getCoeficient()+"*x^"+m.getPutere()+"";
155             System.out.println(m.getCoeficient()+"*x^"+m.getPutere()+"");
156         }
157         System.out.println();
158         return s;
159     }
160 }
```

CLASA MAIN

In clasa main exista metoda main in care am creat o intanta de polinom pe care am trimis-o ca parametru in momentul in care am creat instante de ClassView si de ClassController. Am setat vizibilitatea vederii la true pentru a fi afisat alculatorul.

```
1   import java.util.ArrayList;
2   import java.util.List;
3
4   public class Main {
5           public static void main(String[] args) {
6
7               Polinom p = new Polinom(new ArrayList());
8               ClassView view = new ClassView(p);
9               ClassController controller = new ClassController(view, p);
10              view.setVisible(true);
11          }
12
13  }
```

CLASA CLASSVIEW

In aceasta clasa am facut vederea, si anume ceea ce vede utilizatorul in momentul in care deschide calculatorul de polinoame. Am facut instante de butoane si de spatii de text si o instanta de polinom. In metoda ClassView am creat panel-uri pentru a delimita butoanele de textFielduri, am creat fonturi pentru label-uri si, de asemenea, am setat culoarea background-ului. In primul panel am adaugat un label in care am scris dupa ce format ar trebui sa introduca utilizatorul polinoamele, in al doilea panel am scris ca este vorba despre primul polinom si i-am pus textField-ul, in al doilea panel am pozitionat cel de-al doilea polinom, in al treilea panel am pus butoanele, in al patrulea, rezultatele (adica 2 textField-uri deoarece la impartire avem si rest), si am setat ca in textfield-ul aferent rezultatelor sa nu se poata introduce nimic de catre utilizator, acolo se vor afisa strict rezultatele in urma operatiilor, iar in ultimul label am pus butonul de clear. Am implementat cate o metoda pentru fiecare buton pe care le-am apelat in ClassController, unde am implementat ce se va intampla la apasarea fiecarui buton. Tot aici

mai exista 2 metode care returneaza datele introduse de persoana care foloseste calculatorul, si anume, cele 2 polinoame si inca 2 pentru a seta rezultatele operatiilor in cele 2 spatii de text pentru ele (aceste metode se vor apela in clasa ClassController). Am facut si o metoda de reset care seteaza fiecare textField la valoarea null.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class ClassView extends JFrame{
    private JButton add = new JButton( text: "Adunare");
    private JButton sub = new JButton( text: "Scadere");
    private JButton div = new JButton( text: "Impartire");
    private JButton multi = new JButton( text: "Inmultire");
    private JButton deriv = new JButton( text: "Derivare");
    private JButton integr = new JButton( text: "Integrare");
    private JButton m_clearBtn = new JButton( text: "Clear");
    private JTextField m_fieldPolinom1 = new JTextField( columns: 20);
    private JTextField m_fieldPolinom2 = new JTextField( columns: 20);
    private JTextField m_fieldResult1 = new JTextField( columns: 40);
    private JTextField m_fieldResult2 = new JTextField( columns: 20);
    private Polinom polin;


    ClassView(Polinom p1){
        polin = p1;

        Font f = new Font( name: "TimesRoman",Font.BOLD, size: 15);
        Font f1 = new Font( name: "TimesRoman",Font.BOLD, size: 20);
        Font f2 = new Font( name: "TimesRoman",Font.ITALIC, size: 15);
        Font f3 = new Font( name: "TimesRoman",Font.BOLD, size: 12);
        JPanel panel0 = new JPanel();
        JPanel panel1 = new JPanel();
        JPanel panel2 = new JPanel();
        JPanel panel3 = new JPanel();
        JPanel panel4 = new JPanel();
        JPanel panel5 = new JPanel();

        JLabel label0 = new JLabel( text: "!");
        label0.setFont(f1);
        label0.setForeground(Color.red);
        JLabel label = new JLabel( text: "Vă rugăm să introduceți polinoamele după următorul format, începând cu gradul cel mai mare și terminând cu gradul cel mai mic: ");
        label.setFont(f3);
```

```
        label.setForeground(Color.white);
        JLabel l = new JLabel( text: "''1.0*x^7+10.7*x^2+2.1''");
        l.setForeground(Color.gray);
        l.setFont(f2);
        panel0.add(label0);
        panel0.add(label);
        panel0.add(l);
        panel0.setLayout(new FlowLayout());

        JLabel label1 = new JLabel( text: "Primul polinom: ");
        label1.setForeground(Color.white);
        label1.setFont(f);
        panel1.add(label1);
        panel1.add(m_fieldPolinom1);
        panel1.setLayout(new FlowLayout());

        JLabel label2 = new JLabel( text: "Al doilea polinom: ");
        label2.setForeground(Color.white);
        label2.setFont(f);
        panel2.add(label2);
        panel2.add(m_fieldPolinom2);
        panel2.setLayout(new FlowLayout());

        panel3.add(add);
        panel3.add(sub);
        panel3.add(div);
        panel3.add(multi);
        panel3.add(deriv);
        panel3.add(integr);
        panel3.setLayout(new FlowLayout());

        JLabel label3 = new JLabel( text: "Rezultat: ");
        label3.setForeground(Color.white);
        label3.setFont(f);
        panel4.add(label3);
        panel4.add(m_fieldResult1);
        m_fieldResult1.setOpaque(false);
        panel4.add(m_fieldResult2);
```

```
        m_fieldResult2.setOpaque(false);
        panel4.setLayout(new FlowLayout());

        m_fieldResult1.setEnabled(false);
        m_fieldResult2.setEnabled(false);

        panel5.add(m_clearBtn);
        panel5.setLayout(new FlowLayout());


        JPanel p = new JPanel();
        p.add(panel0);
        p.add(panel1);
        p.add(panel2);
        p.add(panel3);
        p.add(panel4);
        p.add(panel5);
        panel0.setBackground(Color.black);
        panel1.setBackground(Color.black);
        panel2.setBackground(Color.black);
        panel3.setBackground(Color.black);
        panel4.setBackground(Color.black);
        panel5.setBackground(Color.black);

        p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
        //setSize(670,700);
        setPreferredSize(new Dimension( width: 1000, height: 400));
        this.setContentPane(p);
        this.pack();

        this.setTitle("Calculator de polinoame");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
    void addSumListener(ActionListener aal) { add.addActionListener(aal); }

    void addDifListener(ActionListener aal) { sub.addActionListener(aal); }

```

```
    void addImpListener(ActionListener aal){ div.addActionListener(aal); }

    void addOriListener(ActionListener aal) { multi.addActionListener(aal); }

    void addDerivListener(ActionListener aal) { deriv.addActionListener(aal); }

    void addIntegrListener(ActionListener aal) { integr.addActionListener(aal); }

    void setResult1(String result1) { m_fieldResult1.setText(result1); }

    void setResult2(String result2){ m_fieldResult2.setText(result2); }

    void addClearListener(ActionListener cal) { m_clearBtn.addActionListener(cal); }


    String getUser1Input(){return m_fieldPolinom1.getText();}

    String getUser2Input(){return m_fieldPolinom2.getText();}

    void reset(){
        m_fieldPolinom1.setText(null);
        m_fieldPolinom2.setText(null);
        m_fieldResult1.setText(null);
        m_fieldResult2.setText(null);
    }

}
```

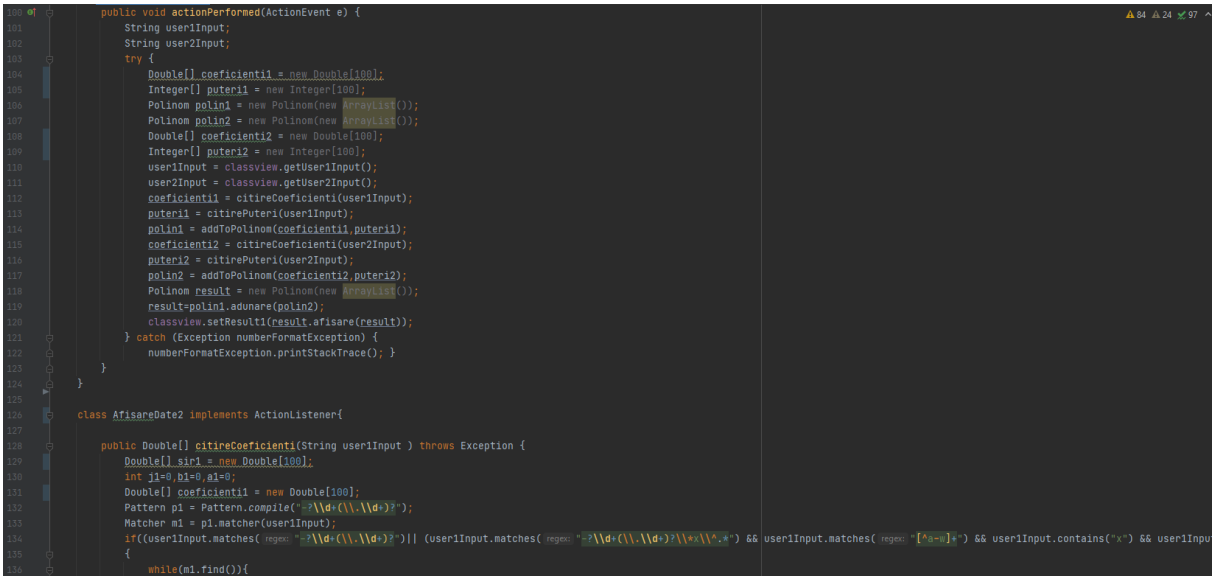
CLASA CLASSCONTROLLER

In clasa ClassController am 2 variabile instanta, una de Polinom si una de ClassView. Am creat un constructor in care am atribuit apasarii fiecarui buton cate o clasa. M-am folosit de interfata ActionListener cand am creat clasele pentru butoane. In fiecare dintre cele 6 clase aferente fiecarei operatii am implementat 4 metode, 3 dintre ele sunt identice deci am sa le explic doar o singura data, iar a 4-a are acelasi nume peste tot doar ca diferaimplementarea.

Prima clasa este pentru butonul de adunare si avem pentru inceput metoda citireCoeficienti(String). Dupa cum sugereaza si titlul, se vor citi coeficientii din string-ul trimis ca parametru care este de fapt fiecare polinom introdus, pe rand. Am luat un sir de numere reale in care am introdus atat coeficientii cat si puterile din String cu ajutorul clasei Pattern si expresiilor regulate: am folosit expresia „-?\\d+(\\.\\d+)?” pentru a extrage numere reale atat pozitive cat si negative, dar inainte de a extrage coeficientii si puterile am verificat daca Stringul introdus se potriveste formatului impus, fara alte caractere in plus, tot cu ajutorul clasei Pattern din pachetul regex. In cazul in care a fost introdus gresit polinomul, se afiseaza un mesaj de eroare. Daca este corect se vor citi coeficientii si puterile pe rand in sir, ordinea fiind: coeficient, putere, coeficient, putere s.a.m.d. Folosindu-ma de un foreach(..) am extras din sir doar coeficientii, adica elementele de pe pozitiile pare si le-am adaugat intr-un array de coeficienti pe care l-am returnat. Metoda urmatoare este cea de citirePuteri(String). Am trimis ca parametru tot polinomul introdus dee utilizator, doar ca de data asta am extras puterile. Am procedat in acelasi mod ca la metoda anterioare, insa, in loc sa pun elementele de pe pozitia para din sir, le-am ales pe cele de pe pozitia impara, le-am convertit la int, deoarece puterile sunt numere intregi si le-am adaugat intr-un array de puteri. In aceasta metoda am verificat daca puterile sunt puse in ordinea ceruta, adica incepand cu puterea cea mai mare si terminand cu cea mai mica, daca nu, se afiseaza un mesaj de eroare. Urmatoarea metoda este addToPolinom(Double[], Integer[]) unde am trimis ca parametrii sirul de coeficienti si sirul de puteri. Aici am creat un polinom si am adaugat in lista sa de monoame cate un nou monom avand coeficientul si puterea aflate pe aceeasi pozitie in sirul fiecaruia deoarece le-am adaugat in ordine in sirurile respective. Este posibil sa existe cu un coeficient in plus fata de numarul de puteri deoarece acel coeficient corespunde puterii 0, de aceea, atunci cand am creat monomul corespunzator acelui coeficient, am adaugat la putere numarul 0. Ultima metoda din aceasta clasa este actionPerformed, adica cea in care se implementeaza functionarea la apasarea butonului. Am creat array-uri pentru coeficientii si puterile fiecarui polinom pe care i-am adaugat apeland functiile citireCoeficienti si citirePuteri si le-am adaugat in 2 polinoame diferite. Am creat 2 Stringuri in care am plasat polinoamele introduse de utilizator, apeland functiile care returneaza ceea ce exista in textField-urile corespunzatoare din classview. Am creat un polinom in care am adaugat rezultatul adunarii apeland metoda adunare din primul polinom in care am trimis ca parametru al doilea polinom. Am apelat apoi metoda afisare din polinomul rezultat, metoda care returneaza un String, pe care l-am fixat in textField-ul pentru rezultatul1.

In a doua clasa, cea pentru butonul de scadere, am procedat in acelasi mod, metodele au aceleasi nume, singura diferenta este ca in metoda action performed, in polinomul rezultat am apelat functia scadere din primul polinom si l-am trimis pe al doilea ca parametru, dupa aceea am afisat rezultatul in acelasi textField. In urmatoarea, cea pentru butonul de impartire, la fel, doar ca in action performed am apelat din primul polinom functia de impartire, cu parametrul, al doilea polinom, rezultatul l-am pus intr-o lista de polinoame. Acest rezultat contine catul pe prima pozitie si restul pe a doua. Catul l-am pus in primul textField de la rezultat, iar restul in al doilea. In a patra clasa, cea pentru inmultire, am lucrat la fel doar ca am afisat polinoamele inmutite ca si rezultat. In clasa pentru butonul de derivare am citit coeficientii si puterile primului polinom introdus si am apelat functia derivare pentru el. La fel am facut si pentru butonul de integrare, doar ca am afisat integrala din primul polinom. Ultima clasa este pentru butonul clear in care am creat metoda actionPerformed pentru apasarea butonului clear si in acel moment se apeleaza metoda reset din classview care goleste toate textField-urile.

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class ClassController {
    private ClassView classview;
    private Polinom polin;

    ClassController(ClassView view, Polinom p){

        classview = view;
        polin = p;
        classview.addSumListener(new AfisareDate1());
        classview.addDifListener(new AfisareDate2());
        classview.addImpListener(new AfisareDate3());
        classview.addOriListener(new AfisareDate4());
        classview.addDerivListener(new AfisareDate5());
        classview.addIntegrListener(new AfisareDate6());
        classview.addClearListener(new ClearListener());


    }

    class AfisareDate1 implements ActionListener{

        public Double[] citireCoeficienti(String user1Input ) throws Exception {
            Double[] sir = new Double[100];
            int j1=0,b1=0,a1=0;
            Double[] coeficienti1 = new Double[100];
            Pattern p1 = Pattern.compile("-?\\d+(\\.\\d+)?");
            Matcher m1 = p1.matcher(user1Input);
            if((user1Input.matches( regex: "-?\\d+(\\.\\d+)?")|| (user1Input.matches( regex: "-?\\d+(\\.\\d+)?\\*x\\^.*") && user1Input.matches( regex: "[^a-u]+") && user1Input.contains("x") && user1Input.
            {
                while(m1.find()){
                    sir[j1]=Double.parseDouble(m1.group());
                    j1++; }
                int[] numbers1 =new int[j1];
                while(b1<j1){
                    numbers1[b1]=b1;
                    b1++; }
                for(int i: numbers1){
                    if(i%2==0){
                        coeficienti1[a1]=sir[i];
                        a1++; }}
            return coeficienti1;}
            else
                throw new Exception("DidntRespectTheRequirementsException");
        }

        public Integer[] citirePuteri(String user1Input ) throws Exception {
            Double[] sir1 = new Double[100];
            int j1=0,b1=0,a1=0,put1=0,cnt1=0;
            Integer[] puteri1 = new Integer[100];
            Pattern p1 = Pattern.compile("-?\\d+(\\.\\d+)?");
            Matcher m1 = p1.matcher(user1Input);
            if((user1Input.matches( regex: "-?\\d+(\\.\\d+)?")|| (user1Input.matches( regex: "-?\\d+(\\.\\d+)?\\*x\\^.*") && user1Input.matches( regex: "[^a-u]+") && user1Input.contains("x") && user1Inp
            {   while(m1.find()){
                    sir1[j1]=Double.parseDouble(m1.group());
                    j1++; }
                int[] numbers1 =new int[j1];
                while(b1<j1){
                    numbers1[b1]=b1;
                    b1++; }
                for(int i: numbers1){
                    if(i%2==1){
                        puteri1[a1]=sir1[i].intValue();
                        a1++; }}
                int[] puteri = new int[a1];
                while(put1<a1-1){
                     puteri[put1]=put1;
                    put1++; }
                for(int i1: puteri){
                    if(puteri1[i1+1]!=null)
                        if(puteri1[i1]<puteri1[i1+1])
                            cnt1++;
                    else return puteri1;}
                if(cnt1==0) return puteri1;
                else throw new Exception("DidntRespectTheRequirementsException"); }
            else throw new Exception("DidntRespectTheRequirementsException"); }

        public Polinom addToPolinom(Double[] coeficienti, Integer[] puteri){
            int i=0;
            Polinom p = new Polinom(new ArrayList());

                while(puteri[i]!=null){
                    Monom mon = new Monom(puteri[i],coeficienti[i]);
                    p.addMon(mon);
                    i++;
                }if(coeficienti[i]!=null){
                Monom mon = new Monom( pi: 0,coeficienti[i]);
                p.addMon(mon);}
            return p;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            String user1Input;
            String user2Input;
            try {
                Double[] coeficienti1 = new Double[100];
                Integer[] puteri1 = new Integer[100];
                Polinom polin1 = new Polinom(new ArrayList());
                Polinom polin2 = new Polinom(new ArrayList());
                Double[] coeficienti2 = new Double[100];
                Integer[] puteri2 = new Integer[100];
                user1Input = classview.getUser1Input();
                user2Input = classview.getUser2Input();
                coeficienti1 = citireCoeficienti(user1Input);
                puteri1 = citirePuteri(user1Input);
                polin1 = addToPolinom(coeficienti1,puteri1);
```



```
    public void actionPerformed(ActionEvent e) {
        String user1Input;
        String user2Input;
        try {
            Double[] coeficienti1 = new Double[100];
            Integer[] puteri1 = new Integer[100];
            Polinom polin1 = new Polinom(new ArrayList());
            Polinom polin2 = new Polinom(new ArrayList());
            Double[] coeficienti2 = new Double[100];
            Integer[] puteri2 = new Integer[100];
            user1Input = classview.getUser1Input();
            user2Input = classview.getUser2Input();
            coeficienti1 = citireCoeficienti(user1Input);
            puteri1 = citirePuteri(user1Input);
            polin1 = addToPolinom(coeficienti1,puteri1);
            coeficienti2 = citireCoeficienti(user2Input);
            puteri2 = citirePuteri(user2Input);
            polin2 = addToPolinom(coeficienti2,puteri2);
            Polinom result = new Polinom(new ArrayList());
            result=polin1.adunare(polin2);
            classview.setResult1(result.afisare(result));
        } catch (Exception numberFormatException) {
            numberFormatException.printStackTrace(); }
    }
}

class AfisareDate2 implements ActionListener{

    public Double[] citireCoeficienti(String user1Input ) throws Exception {
        Double[] sir1 = new Double[100];
        int j1=0,b1=0,a1=0;
        Double[] coeficienti1 = new Double[100];
        Pattern p1 = Pattern.compile("-?\\d+(\\.\\d+)?");
        Matcher m1 = p1.matcher(user1Input);
        if((user1Input.matches( regex: "-?\\d+(\\.\\d+)?")|| (user1Input.matches( regex: "-?\\d+(\\.\\d+)?\\*x\\^.*") && user1Input.matches( regex: "[^a-w]+") && user1Input.contains("x") && user1Inpu
        {
            while(m1.find()){
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        String user1Input;
        String user2Input;
        try {
            Double[] coeficienti1 = new Double[100];
            Integer[] puteri1 = new Integer[100];
            Polinom polin1 = new Polinom(new ArrayList());
            Polinom polin2 = new Polinom(new ArrayList());
            Double[] coeficienti2 = new Double[100];
            Integer[] puteri2 = new Integer[100];
            user1Input = classview.getUser1Input();
            user2Input = classview.getUser2Input();
            coeficienti1 = citireCoeficienti(user1Input);
            puteri1 = citirePuteri(user1Input);
            polin1 = addToPolinom(coeficienti1,puteri1);
            coeficienti2 = citireCoeficienti(user2Input);
            puteri2 = citirePuteri(user2Input);
            polin2 = addToPolinom(coeficienti2,puteri2);
            Polinom result = new Polinom(new ArrayList());
            result=polin1.scadere(polin2);
            classview.setResult1(result.afisare(result));
        } catch (Exception numberFormatException) {
            numberFormatException.printStackTrace();
        }
    }
}
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        String user1Input;
        String user2Input;
        try {
            Double[] coeficienti1 = new Double[100];
            Integer[] puteri1 = new Integer[100];
            Polinom polin1 = new Polinom(new ArrayList());
            Polinom polin2 = new Polinom(new ArrayList());
            Double[] coeficienti2 = new Double[100];
            Integer[] puteri2 = new Integer[100];
            user1Input = classview.getUser1Input();
            user2Input = classview.getUser2Input();
            coeficienti1 = citireCoeficienti(user1Input);
            puteri1 = citirePuteri(user1Input);
            polin1 = addToPolinom(coeficienti1,puteri1);
            coeficienti2 = citireCoeficienti(user2Input);
            puteri2 = citirePuteri(user2Input);
            polin2 = addToPolinom(coeficienti2,puteri2);
            List<Polinom> result = new ArrayList<Polinom>();
            result=polin1.impartire(polin2);
            classview.setResult1(result.get(0).afisare(result.get(0)));
            classview.setResult2(result.get(1).afisare(result.get(1)));
        } catch (Exception numberFormatException) {
            numberFormatException.printStackTrace();
        }
    }
}
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            String user1Input;
            String user2Input;
            try {
                Double[] coeficienti1 = new Double[100];
                Integer[] puteri1 = new Integer[100];
                Polinom polin1 = new Polinom(new ArrayList());
                Polinom polin2 = new Polinom(new ArrayList());
                Double[] coeficienti2 = new Double[100];
                Integer[] puteri2 = new Integer[100];
                user1Input = classview.getUser1Input();
                user2Input = classview.getUser2Input();
                coeficienti1 = citireCoeficienti(user1Input);
                puteri1 = citirePuteri(user1Input);
                polin1 = addToPolinom(coeficienti1,puteri1);
                coeficienti2 = citireCoeficienti(user2Input);
                puteri2 = citirePuteri(user2Input);
                polin2 = addToPolinom(coeficienti2,puteri2);
                Polinom result = new Polinom(new ArrayList());
                result=polin1.inmultire(polin2);
                classview.setResult1(result.afisare(result));
            } catch (Exception numberFormatException) {
                numberFormatException.printStackTrace();
            }
        }
    }
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            String user1Input;
            try {
                Double[] coeficienti1 = new Double[100];
                Integer[] puteri1 = new Integer[100];
                Polinom polin1 = new Polinom(new ArrayList());
                user1Input = classview.getUser1Input();
                coeficienti1 = citireCoeficienti(user1Input);
                puteri1 = citirePuteri(user1Input);
                polin1 = addToPolinom(coeficienti1,puteri1);
                Polinom result = new Polinom(new ArrayList());
                result=polin1.derivare();
                classview.setResult1(result.afisare(result));
            } catch (Exception numberFormatException) {
                numberFormatException.printStackTrace();
            }
        }
    }
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            String user1Input;
            try {
                Double[] coeficienti1 = new Double[100];
                Integer[] puteri1 = new Integer[100];
                Polinom polin1 = new Polinom(new ArrayList());
                user1Input = classview.getUser1Input();
                coeficienti1 = citireCoeficienti(user1Input);
                puteri1 = citirePuteri(user1Input);
                polin1 = addToPolinom(coeficienti1,puteri1);
                Polinom result = new Polinom(new ArrayList());
                result=polin1.integrare();
                classview.setResult1(result.afisare(result));
            } catch (Exception numberFormatException) {
                numberFormatException.printStackTrace();
            }

        }
    }
class ClearListener implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent arg0) { classview.reset(); }

}

}
```

6. Concluzii si dezvoltari ulterioare

Se pot implementa butoane pentru cifre si pentru simbolurile folosite in introducerea polinoamelor, astfel incat interfata cu utilizatorul sa arate asemenea unui claculator.

De asemenea, acest calculator se poate transforma intr-unul de calculare a radacinilor polinomului sau intr-un calculator de determinare a celui mai mare divizor comun dintre 2 polinoame.

In concluzie, aceste limbaje OOP sunt foarte usor de folosit si la indemana oricui stie paradigmele OOP de baza. Mi-am perfectionat modul de programare, atat aplicand tehnici noi cat si repetandu-le pe cele deja invatate.

7. Bibliografie

- Documentele puse la dispozitie semestrul trecut de catre facultate: cursuri, documente de la laborator;
- https://www.w3schools.com/java/java_regex.asp
- https://www.javatpoint.com/encapsulation
- https://www.javatpoint.com/java-swing
- YouTube