

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Calculator de polinoame

Documentație

Ciubotaru Andrei-Mihai

Grupa: 30227

Cuprins

1. Obiectivul temei
2. Analiza problemei
3. Proiectare
 - 3.1 Diagrama de clase
4. Implementare
 - 4.1 Clase si pachete
 - 4.2 GUI
5. Testare
6. Rezultate
7. Concluzii
8. Bibliografie

1.Obiectivul temei

Obiectivul acestei teme este sa propunem, sa proiectam si sa implementam un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

Se doreste citirea de la tastatura a doua polinoame sub forma $5X^3+2X-7$, astfel incat sa se poata realiza asupra lor operatiile de adunare, scadere si inmultime. In plus, primului polinom citit i se poate aplica si operatia de derivare sau integrare. Practic se va implementa un calculator de polinoame.

Acesta va dispune de o interfata „User Friendly”, care poate fi utilizata de oricine.

2.Analiza problemei

Un polinom reprezinta o expresie construita din unul sau mai multi termeni numiti moname, care sunt alcatuite dintr-o constanta, numita coeficient inmultita cu o variabila(X), care poate avea un exponent intreg pozitiv.

Perluarea polinomului introdus de utilizator reprezinta o etapa foarte importanta in rezolvarea temei, acest lucru putandu-se realiza in mai multe feluri.

3.Proiectare

Am ales o metoda care elimina neajunsurile cum ar fi limitarea gradului unui polinom, astfel incat utilizatorul poate introduce orice polinom doreste. Am folosit un RegEx pentru a despartii string-ul citit in moname si pentru a prelua coeficientul si puterea. Aceasta metoda este mai simpla si mai eficienta decat verificarea string-ului caracter cu caracter.

Diagrama UML presupune modelarea unui sistem prin lucrurile care sunt importante pentru acesta. Aceste lucruri sunt modelate folosind clase.

3.1. Diagrama de clase

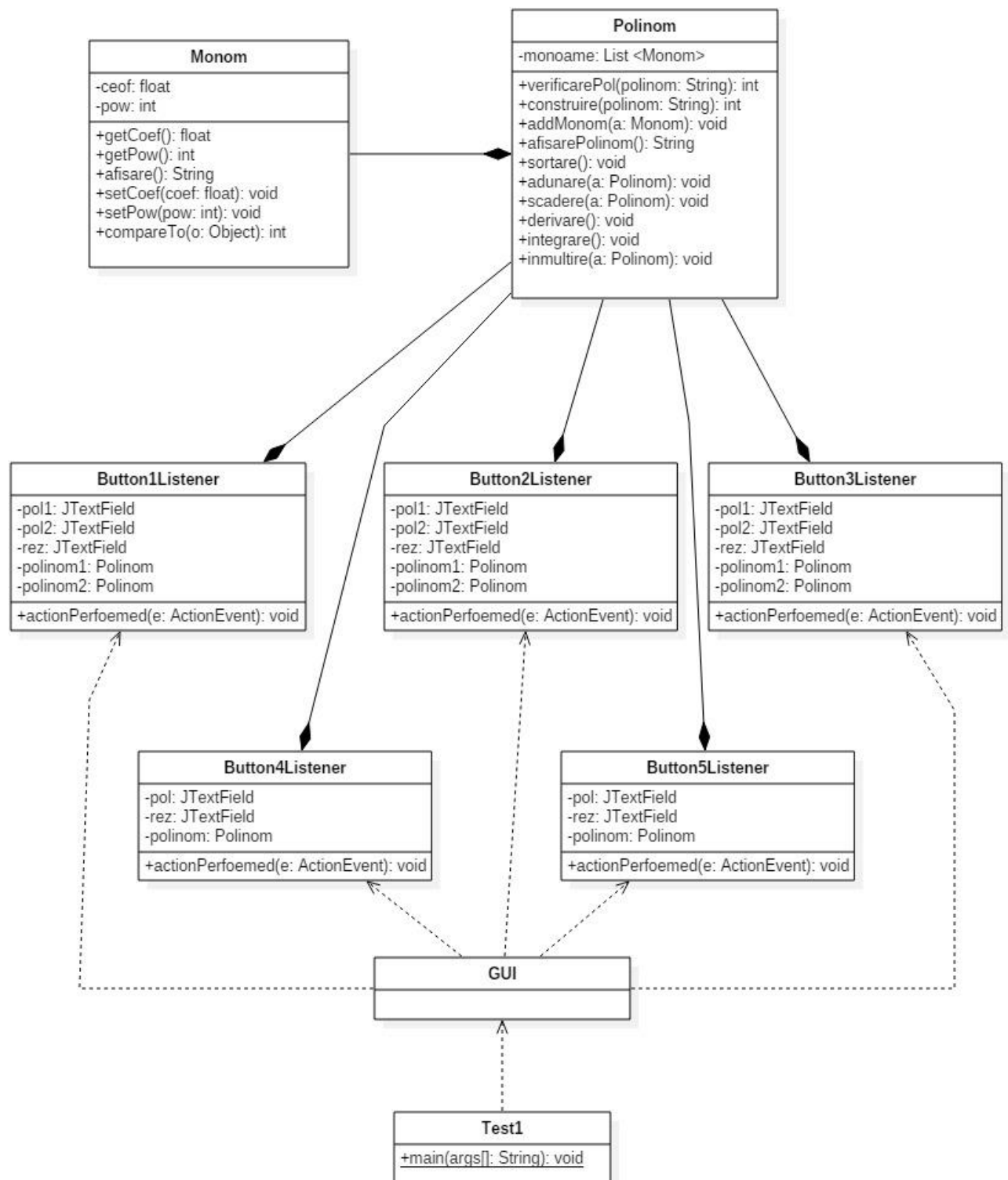


Diagrama Use-case

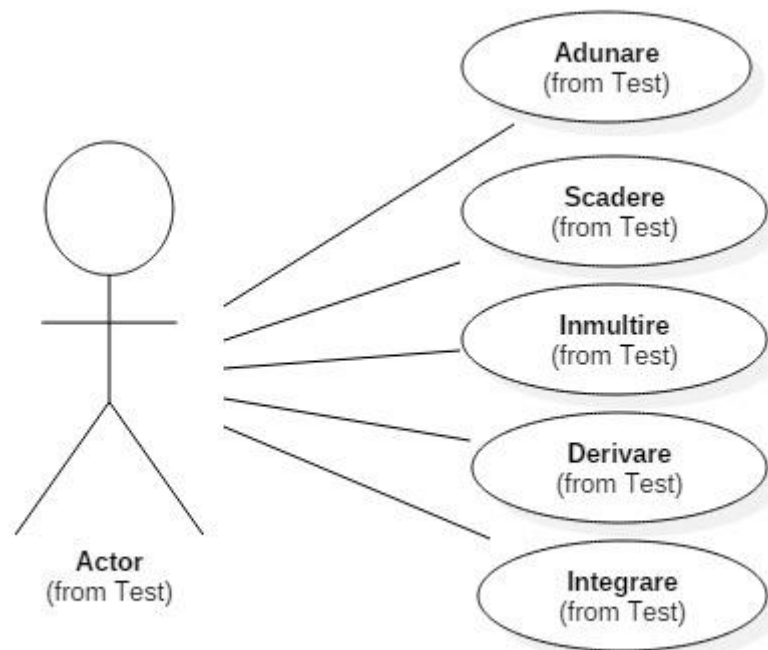
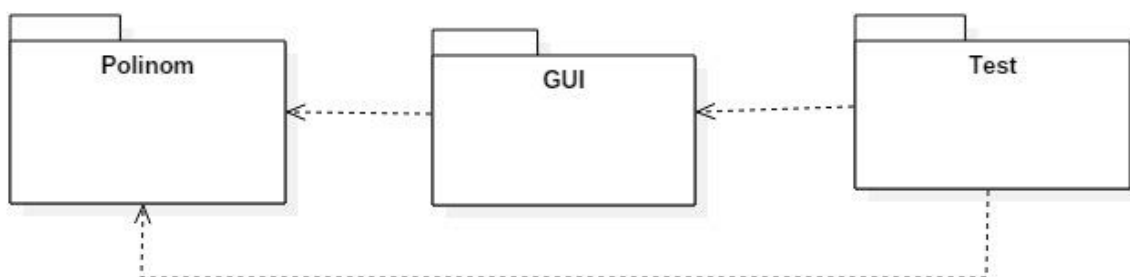


Diagrama Uuse-case prezinta o colectie de cazuri de utilizare si actori, oferind o descriere generala a modului in care va fi utilizat sistemul. Astfel actorul (utilizatorul) dupa introducerea polinoamelor poate cere sistemului una din cele cinci operatii. Operatiile sunt reprezentate de cazurile de utilizare. Atunci cand actorul alege un caz de utilizare sistemul va genera un raspuns.

Diagrama de pachete



4.Implementare

4.1.Clase si pachete

Am organizat proiectul in trei pachete: Polinom, GUI, Test.

In pachetul Polinom se gasesc clasele Monom si Polinom. Aceste clase au rolul de a verifica si construi efectiv polinomul. Clasa Monom are doua attribute coef si pow in care se stocheaza coeficientul si puterea unui monom. Clasa Polinom are ca atribut o lista de tipul Monom. Totodata in aceasta clasa se implementeaza si operatiile de adunare, scadere, inmultire, derivare si integrare.

In pachetul GUI se gasesc clasele cu ajutorul carora se constituie interfata. In clasele Button1Listener, Button2Listener, Button3Listener, Button4Listener, Button5Listener se implementeaza actiunile care se executa in momentul apasarii unui buton. Iar in clasa GUI vom gasi implementarea interfetei, care e alcatuita din panel-uri. Aceste panel-uri contin butoane, textField-uri si etichete.

In pachetul Test se gasesc doua clase, clasa Test in care sunt implementate testele cu JUnit si clasa Test1 in care e instantiat un obiect de tip GUI, iar din metoda main se lanseaza aplicatia.

Clasa Monom contine metodele getCoef(), getPow(), setCoef(float coef), setPow(int pow) metode cu ajutorul carora se preia si se seteaza la o anumita valoare coeficientul si puterea unui monom. Metoda afisare() returneaza un string care reprezinta monomul si trateaza toate cazurile care pot interveni la afisarea unui monom(ex: $2X^3$, $2X$, X^3 , 5). Cazul in care polinomul este 0 va fi tratat in metoda afisarePolinom din clasa Polinom. Aceasta clasa implementeaza interfata Comparable si suprascrie metoda compareTo() pentru a putea sorta rezultatul(polinomul) unei operatii in ordine descrescatoare dupa gradul monoamelor, astfel rezultatul va fi mai usor de vizualizat.

```
public String afisare() {
    if(pow!=0) {
        if (coef > 0)
            return "+" + coef + "X^" + pow;
        else if (coef == 0)
            return "";
        else
            return coef + "X^" + pow;
    }
    else {
        if (coef >= 0)
            return "+" + coef;
        else
            return "" + coef;
    }
}
```

```

public int compareTo(Object o) {
    Monom a = (Monom)o;
    return Integer.compare(a.pow,pow);
}

```

Clasa Polinom contine metoda verificarePol(String polinom) care verifica daca un string este cu adevarat un polinom, nu are caractere invalide(litere, caractere speciale, etc).

```

public int verificarePol(String polinom){
    String caractere="0123456789+-X^";
    int i,ok=1;
    for(i=0; i<polinom.length(); i++)
        if (caractere.indexOf(polinom.charAt(i))==-1)
            ok=0;
    String polinomRegex = "(-|\\+){2,}|(\\d+\\^)|(\\+|-)|(-\\+)|(X\\d+)|(\\^X)|(XX+)|(^X\\^+$)|(\\^\\^\\d*$)|(^(\\+|-)?(X|\\^)?(\\+|-)$)";
    Pattern pattern = Pattern.compile(polinomRegex);
    Matcher matcher = pattern.matcher(polinom);
    if(matcher.find() || polinom.equals(""))
        ok=0;
    System.out.println(ok);
    return ok;
}

```

Pentru inceput verific daca polinomul contine un caracter care nu se afla in sirul caractere(0123456789+-X^), iar apoi cu ajutorul polinomRegex testez cazurile nevalide cum ar fi X ^, -, +, ^2, etc si setez variabila ok la 1 daca string-ul primit este bun sau 0 in caz contrar.

```

public int construire(String polinom) {
    int pow, coef, ok;
    String polinomRegex = "(\\+|-)?(\\d*)(X)?\\^?(\\d*)";
    Pattern pattern = Pattern.compile(polinomRegex);
    Matcher matcher = pattern.matcher(polinom);
    if (this.verificarePol(polinom)==1) {
        while (matcher.find()) {
            ok = 1;
            String monom = matcher.group(0); //monomul
            String semn = matcher.group(1); //semnul
            String group2 = matcher.group(2); //coeficientul
            String x = matcher.group(3); //X
            String group4 = matcher.group(4); //gradul
            System.out.println(monom + ":" + semn + ", " + group2 + ", " + x + ", "
+ group4);
            if (monom.equals(""))
                ok = 0;
            if (ok == 1) {
                if (group2.equals(""))
                    coef = 1;
                else
                    coef = Integer.parseInt(group2);
                if (semn != null)
                    if (semn.equals("-"))
                        coef = -coef;
                if (x == null)
                    pow = 0;
                else if (group4.equals(""))
                    pow = 1;
                else
                    pow = Integer.parseInt(group4);
                System.out.printf("%d , %d \n", coef, pow);
                Monom p = new Monom(coef, pow);
                this.addMonom(p);
            }
        }
    }
}

```



```

    }
    return 1;
}
else return 0;
}

```

In metoda construire(), construiesc efectiv polinomul daca string-ul primit e ok. Cu ajutorul urmatorului Regex "((\\+|-)?(\\d*)(X)?\\^(\\d*))" impart monomul in grupuri si extrag coeficientul si puterea in variabilele coef si pow si cu ajutorul lor creez un nou monom si il adaug in polinom. Astfel primul grup este reprezentat de monom (matcher.group(0)), cel de-al doi-lea e reprezentat de semn + sau -, al trei-lea e reprezentat de coeficient, urmatorul e reprezentat de variabila X, iar ultimul e reprezentat de grad. La inceput verific ca grupul principal (monom) sa fie diferit de null, pentru ca tot timpul imi gaseste un monom „” (vid). Daca coeficientul exista e diferit de „” atunci cu ajutorul metodei parseInt() transform stringul intr-un int altfel daca de exemplu introducem doar X string-ul salvat in grup va fi „” si setez variabila coef la 1. Daca semnul este „-” atunci coef va fi -coef. Daca nu exista variabila X gradul va 0 altfel daca group4 in care e salvat gradul e „” pow ia valoarea 1 altfel transform string-ul in int. Metoda construiesc un nou polinom si returneaza 1 daca string-ul introdus de utilizator e corect sau 0 daca string-ul e incorect. Am utilizat o functie de tip int pentru a putea afisa un mesaj de eroare. Aceasta metoda este utilizata in clasele ButtonListener in metoda actionPerformed pentru a verifica input-ul.

Metoda AfisarePolinom()

```

public String afisarePolinom() {
    String a = "";
    if(monoame.size()==1 && monoame.get(0).getCoef()==0)
        a = a + "0.0";
    else {
        for (Monom i : monoame)
            if(i.getCoef()==0 && i.getPow()==0)
                a = a + "";
            else
                a = a + i.afisare();
    }
    return a;
}

```

Am tratat special cazul in care polinomul este 0 sau are un monom egal cu 0. Altfel concatenez la string-ul a afisarea fiecarui monom din polinom.

Metoda addMonom(Monom a)

```
public void addMonom(Monom a) {
    int ok = 0;
    for (Monom i : monoame) {
        if (a.getPow() == i.getPow()) {
            i.setCoef(i.getCoef() + a.getCoef());
            ok = 1;
        }
    }
    if (ok == 0)
        monoame.add(a);
}
```

Aceasta metoda adauga un monom la un polinom. Am luat in considerare cazul in care utilizatorul introduce doua moname cu acelasi grad ($3X+X^2+X+7$), astfel parcurg monoamele si daca noul monom pe care vreau sa il adaug are aceeasi putere cu unul din monamele existente atunci setez coeficientul.

Am ales sa implementez operatiile in clasa Polinom. Acestea sunt de tip void, astfel rezultatul va fi salvat in polinomul pe care se apeleaza metoda si nu va trebui sa creez un nou polinom

Adunarea

```
public void adunare(Polinom b) {
    for (Monom i : monoame) {
        for (Monom j : b.monoame)
            if (i.getPow() == j.getPow()) {
                i.setCoef(i.getCoef() + j.getCoef());
            }
    }
    int ok;
    for (Monom j : b.monoame) {
        ok = 0;
        for (Monom i : monoame) {
            if (j.getPow() == i.getPow())
                ok = 1;
        }
        if (ok == 0)
            monoame.add(new Monom(j.getCoef(), j.getPow()));
    }
    this.sortare();
}
```

Metoda primeste ca parametru un polinom b. Algoritmul consta in parcurgerea celor doua polinoame si daca gasesc moname cu grad egal atunci setez coeficientul primului monom ca fiind suma coeficientilor celor doua monoame. La sfarsit mai parcurg odata polinomul b pentru a adauga monoamele care au gradul diferit fata de cele din primul polinom.

In fiecare metoda care implementeaza o operatie apelez metoda sortare()

```
public void sortare() {
    Collections.sort(monoame);
}
```

pentru a sorta monoamele in ordine descrescatoare a gradului.

Operatia de scadere este asemanatoare

```
public void scadere(Polinom b) {
    for (Monom i : monoame) {
        for (Monom j : b.monoame)
            if (i.getPow() == j.getPow()) {
                i.setCoef(i.getCoef() - j.getCoef());
            }
    }
    int ok;
    for (Monom j : b.monoame) {
        ok = 0;
        for (Monom i : monoame) {
            if (j.getPow() == i.getPow())
                ok = 1;
        }
        if (ok == 0)
            monoame.add(new Monom((-1) * j.getCoef(), j.getPow()));
    }
    this.sortare();
}
```

Inmultirea

```
public void inmultire(Polinom b) {
    Polinom c = new Polinom();
    for (Monom i : monoame) {
        for (Monom j : b.monoame)
            c.addMonom(new
Monom(i.getCoef()*j.getCoef(),i.getPow()+j.getPow()));
    }
    monoame.removeAll(monoame);
    for (Monom i : c.monoame) {
        monoame.add(new Monom(i.getCoef(),i.getPow()));
    }
    this.sortare();
}
```

Creez un nou polinom c in care voi avea rezultatul. Parcurg cele doua polinoame si pentru fiecare monom adaug in c un nou monom care are coeficientul produsul coeficientilor celor doua monoaome, iar puterea este suma puterilor celor doua monoaome. Deoarece nu am facut o functie care returneaza un polinom la sfarsit sterg monomale din primul polinom si le adaug pe cele din polinomul c.

Derivarea

```
public void derivare() {
    for (Monom i : monoame) {
        if (i.getPow() != 0) {
            i.setCoef(i.getCoef() * i.getPow());
            i.setPow(i.getPow() - 1);
        }
        else
            i.setCoef(0);
    }
    this.sortare();
}
```

Parcureg monamele din polinom si pentru fiecare setez coeficientul ca fiind coeficient * putere, iar gradul il scad cu 1.

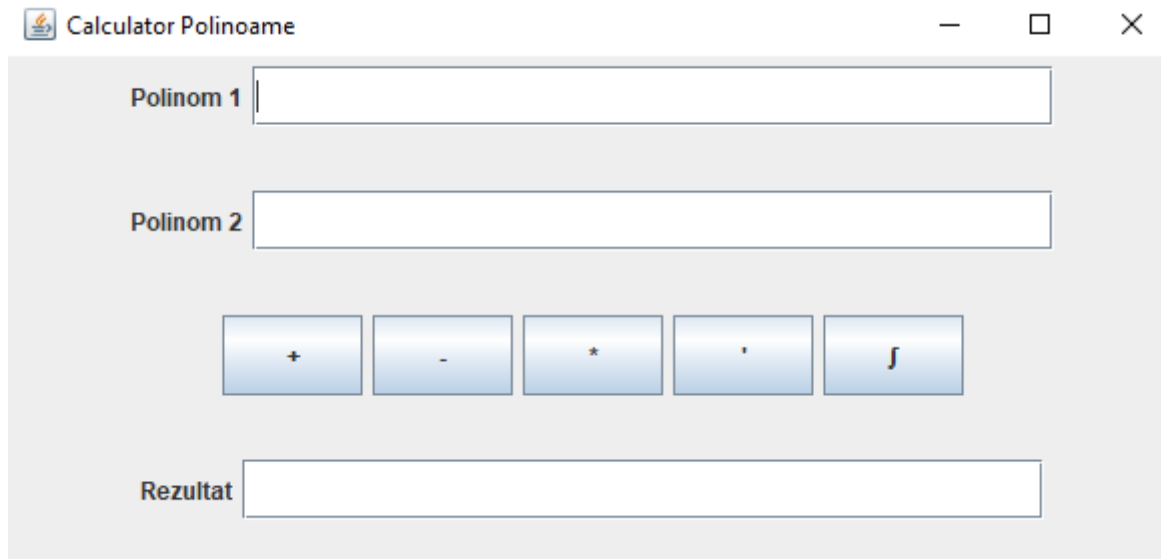
Integrarea

```
public void integrare() {  
    for(Monom i : monoame) {  
        if(i.getPow() != 0) {  
            i.setCoef(i.getCoef() / (i.getPow() + 1));  
            i.setPow(i.getPow() + 1);  
        }  
        else  
            i.setPow(1);  
    }  
    this.sortare();  
}
```

Parcureg monamele din polinom si pentru fiecare setez coeficientul ca fiind ca fiind coeficientul / (putere + 1), iar gradul il cresc cu 1. Daca gradul e 0, monomul e o constanta atunci setez gradul la 1.

4.2.GUI

Am facut o interfata simpla si usor de utilizat. Aceasta contine trei textField-uri, doua in care se introduc cele doua polinoame si unul in care se va afisa rezultatul si cinci butoane, unul pentru fiecare operatie.



In clasa GUI am creat un frame, iar apoi 4 panel-uri.

```
JFrame frame = new JFrame ("Calculator Polinoame");

JPanel panel1 = new JPanel();
JLabel l1 = new JLabel ("Polinom 1");
JTextField tf1 = new JTextField();
tf1.setPreferredSize(new Dimension(400,30));
panel1.add(l1);
panel1.add(tf1);
panel1.setLayout(new FlowLayout());

JPanel panel3 = new JPanel();
JButton b1 = new JButton("+");
b1.setPreferredSize(new Dimension(70,40));
JButton b2 = new JButton("-");
b2.setPreferredSize(new Dimension(70,40));
JButton b3 = new JButton("*");
b3.setPreferredSize(new Dimension(70,40));
JButton b4 = new JButton(".");
b4.setPreferredSize(new Dimension(70,40));
JButton b5 = new JButton("f");
b5.setPreferredSize(new Dimension(70,40));
panel3.add(b1);
panel3.add(b2);
panel3.add(b3);
panel3.add(b4);
panel3.add(b5);
panel3.setLayout(new FlowLayout());
```

In primul panel am adaugat o eticheta(Polinom1) si un textField in care se va introduce polinomul de la tastatura. La fel am facut si pentru Polinom2 si rezultat. In cel de-al trei-lea panel am adaugat cele cinci butoane.

Pentru fiecare buton am facut o clasa ButtonListener, care implementeaza interfata ActionListener, iar in metoda actionPerformed(ActionEvent e) am implementat operatia specifica acelui buton.

```
public class Button1Listener implements ActionListener{
    private JTextField pol1,pol2,rez;
    private Polinom polinom1,polinom2;
    Button1Listener(JTextField pol1,JTextField pol2,JTextField rez) {
        this.pol1=pol1;
        this.pol2=pol2;
        this.rez=rez;
    }

    public void actionPerformed(ActionEvent e){
        String polString1=pol1.getText();
        String polString2=pol2.getText();
        polinom1 = new Polinom();
        polinom2 = new Polinom();
        if(polinom1.construire(polString1)==1 &&
        polinom2.construire(polString2)==1) {
            polinom1.adunare(polinom2);
            rez.setText(polinom1.afisarePolinom());
        }
        else {
            rez.setText("Introduceti un polinom!!!");
        }
    }
}
```

5. Testare

Am folosit JUnit pentru a testa input-ul si operatiile.

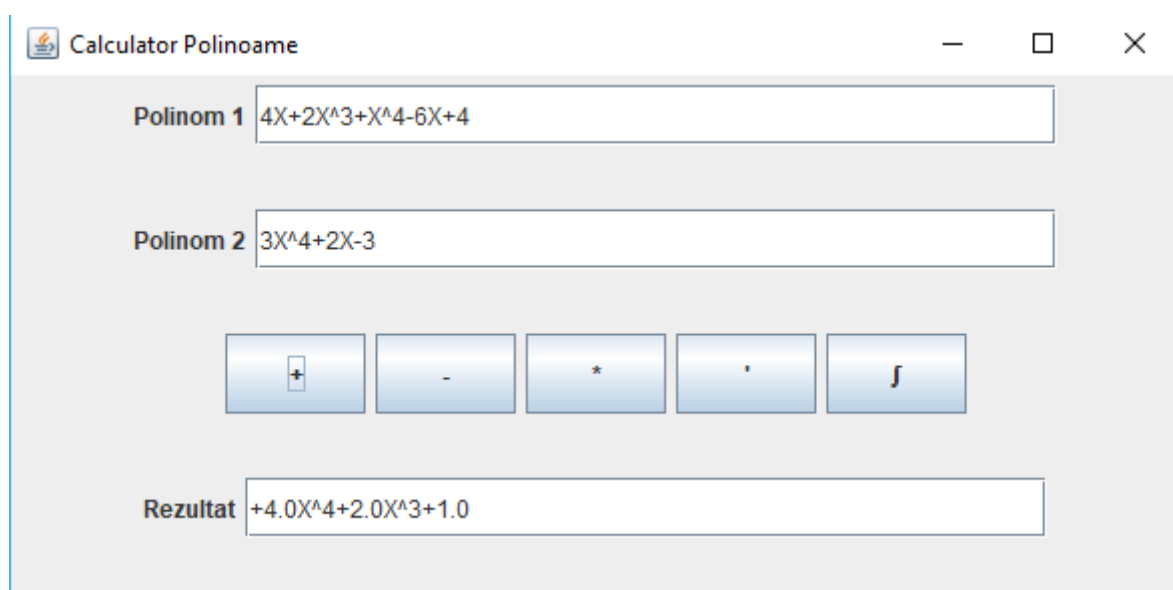
Pentru testInput folosesc metoda verificarePolinom(), iar pentru operatii creez trei polinoame, aplic operatia pe doua dintre ele si compar rezultatul cu cel de-al trei-lea polinom.

```
public class Test {
    Polinom polinom,a,b,c;
    @org.junit.Test
    public void testInput() {
        polinom = new Polinom();
        assertEquals(polinom.verificarePol("3X^4+2"),1);
        assertEquals(polinom.verificarePol("3"),1);
        assertEquals(polinom.verificarePol("X"),1);
        assertEquals(polinom.verificarePol("X^4"),1);
        assertEquals(polinom.verificarePol("-X^4+2"),1);
        assertEquals(polinom.verificarePol("3X^4dsa+2"),0);
        assertEquals(polinom.verificarePol("3X^4%+2"),0);
        assertEquals(polinom.verificarePol("3X^4-+2"),0);
        assertEquals(polinom.verificarePol("2^4+2X+1"),0);
        assertEquals(polinom.verificarePol("3X4+2"),0);
        assertEquals(polinom.verificarePol("-"),0);
        assertEquals(polinom.verificarePol("+"),0);
        assertEquals(polinom.verificarePol("X^^"),0);
    }
    @org.junit.Test
    public void testAdunare() {
        a = new Polinom();
        b = new Polinom();
        c = new Polinom();
        a.construire("3X^4+2X-5");
        b.construire("X^2+7X-3");
        c.construire("3X^4+X^2+9X-8");
        a.adunare(b);
        assertEquals(a.afisarePolinom(),c.afisarePolinom());
    }
    @org.junit.Test
    public void testScadere() {
        a = new Polinom();
        b = new Polinom();
        c = new Polinom();
        a.construire("3X^4+2X-5");
        b.construire("X^4+X^2+7X-3");
        c.construire("2X^4-X^2-5X-2");
        a.scadere(b);
        assertEquals(a.afisarePolinom(),c.afisarePolinom());
    }
    @org.junit.Test
    public void testDerivare() {
        a = new Polinom();
        b = new Polinom();
        a.construire("3X^4+7X+X^2-3X");
        b.construire("12X^3+2X+4");
        a.derivare();
        assertEquals(a.afisarePolinom(),b.afisarePolinom());
    }
}
```


6.Rezultate

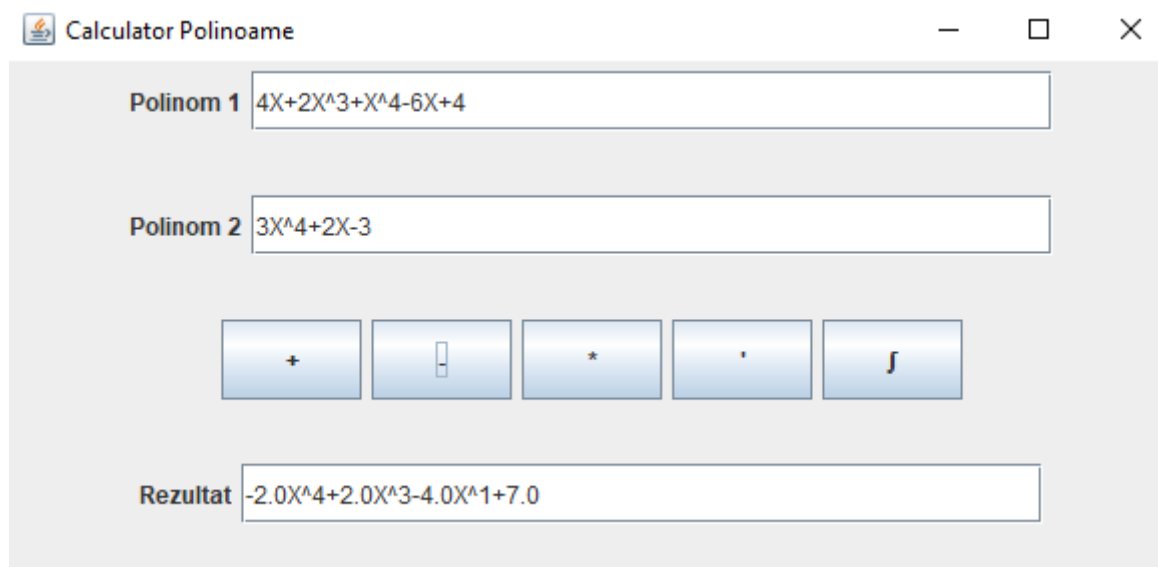
Pentru operatiile de adunare, scadere si inmultire se introduc polinoamele in cele doua textField-uri apoi se apasa pe butunoul specific operatiei, iar rezultatul va aparea in textField-ul specific.

Adunare



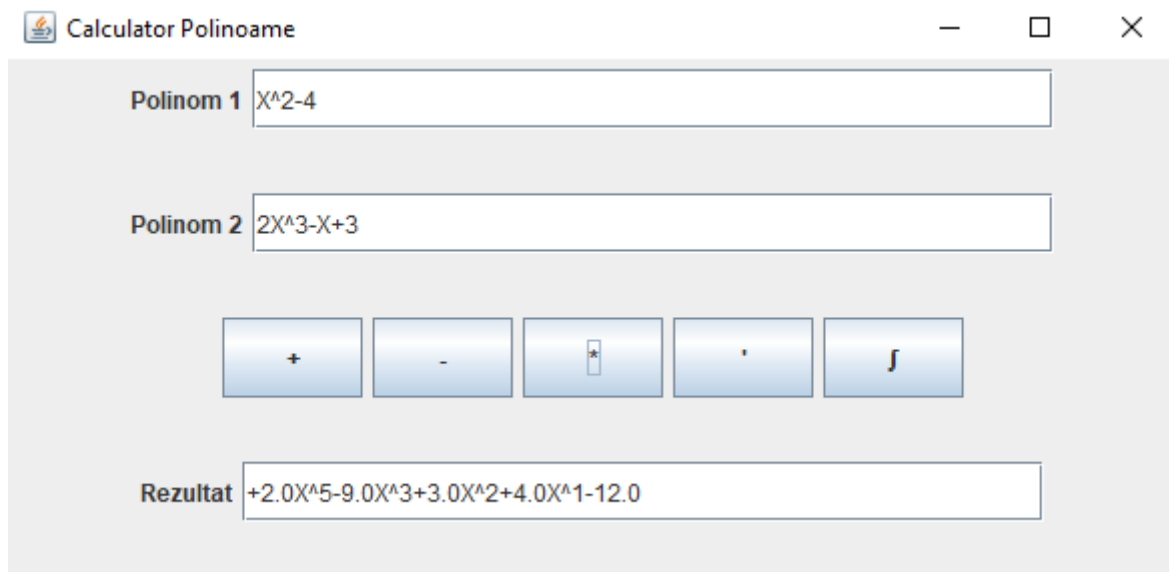
The screenshot shows a window titled "Calculator Polinoame". It contains two input fields: "Polinom 1" with the text $4X+2X^3+X^4-6X+4$ and "Polinom 2" with the text $3X^4+2X-3$. Below these fields is a row of five buttons: a plus sign (+), a minus sign (-), an asterisk (*), a period (.), and an integral symbol (∫). At the bottom, the "Rezultat" field displays $+4.0X^4+2.0X^3+1.0$.

Scadere



The screenshot shows the same "Calculator Polinoame" window. The input fields for "Polinom 1" and "Polinom 2" remain the same. The row of buttons now has the plus sign (+) button disabled and the minus sign (-) button highlighted. The "Rezultat" field displays $-2.0X^4+2.0X^3-4.0X^1+7.0$.

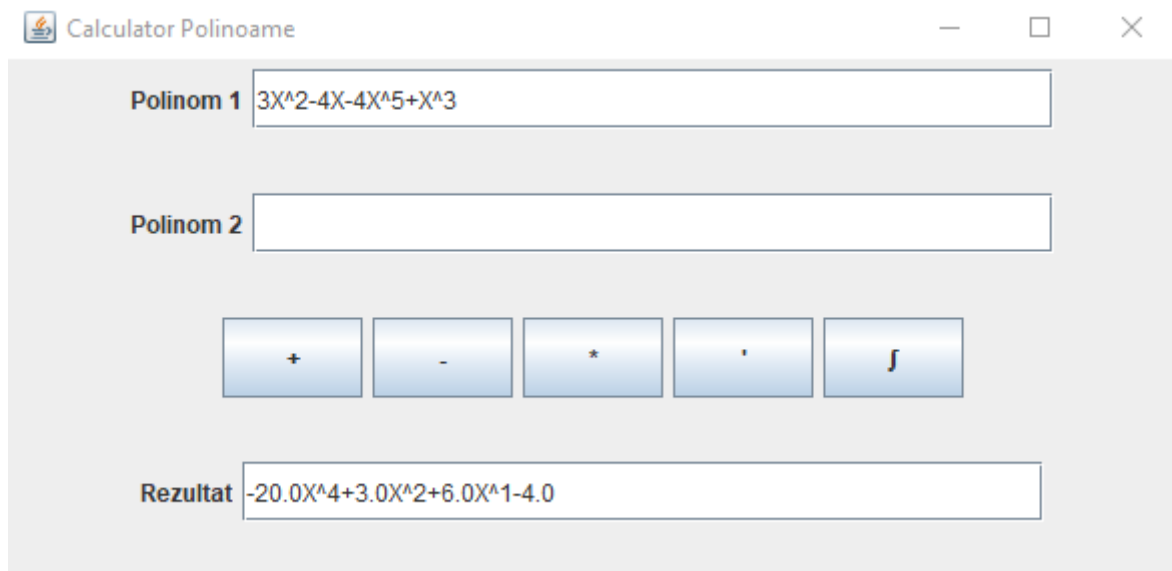
Inmultire



The screenshot shows a window titled "Calculator Polinoame" with standard window controls. It contains two input fields: "Polinom 1" with the text x^2-4 and "Polinom 2" with the text $2x^3-x+3$. Below these fields is a row of five buttons: a plus sign (+), a minus sign (-), a multiplication sign (*), a decimal point (.), and an integral sign (∫). At the bottom, a "Rezultat" field displays the result: $+2.0x^5-9.0x^3+3.0x^2+4.0x^1-12.0$.

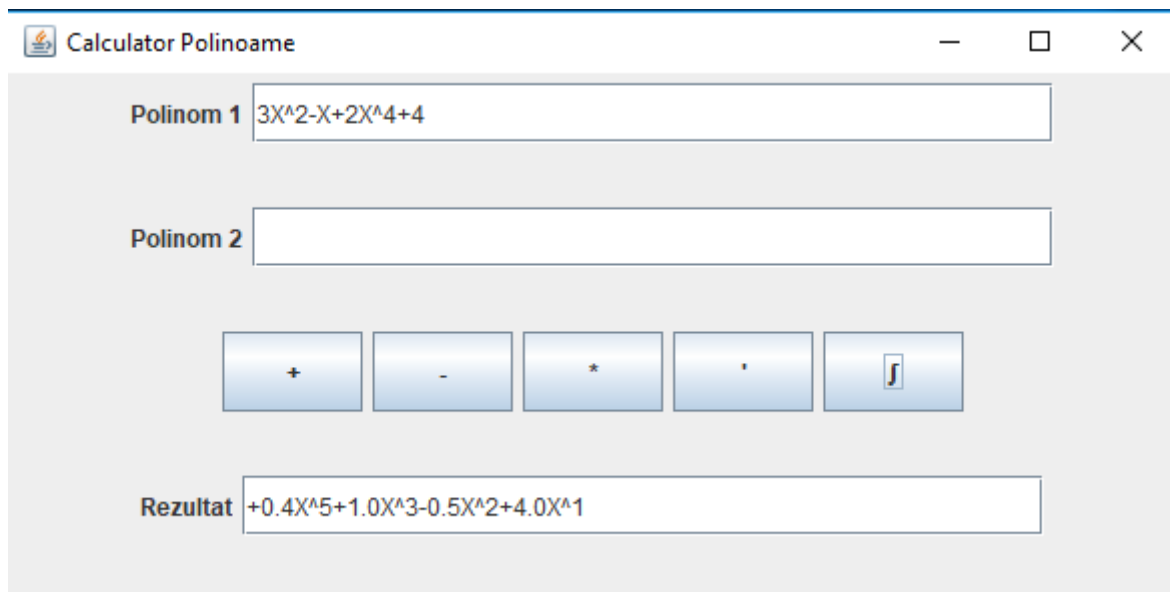
Pentru operatiile de derivare si integrare polinomul se va introduce in primul textField.

Derivare



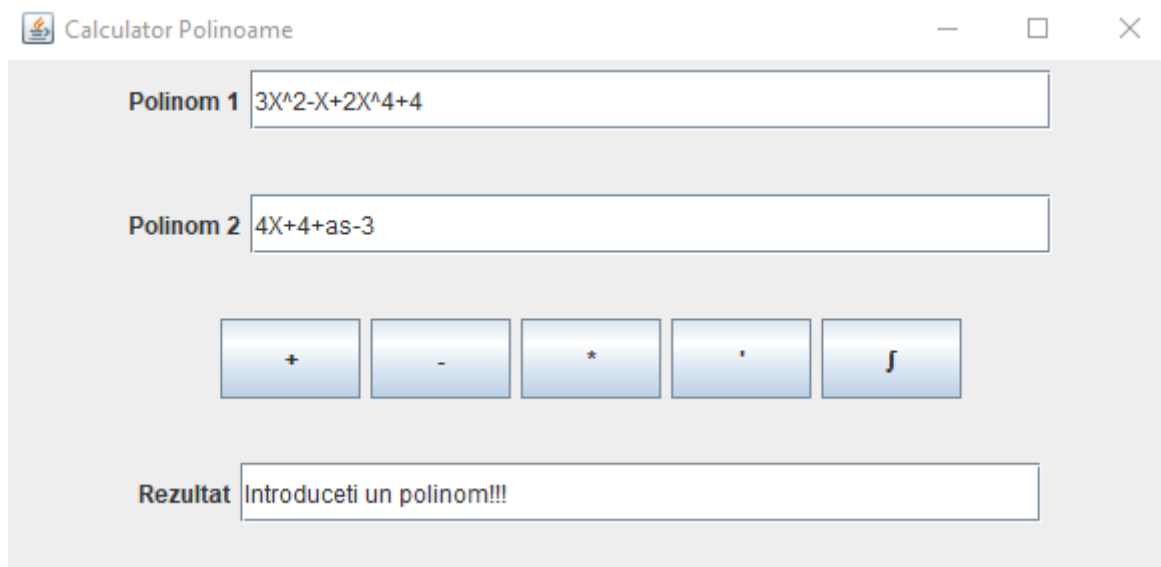
The screenshot shows the same "Calculator Polinoame" window. The "Polinom 1" field now contains the text $3x^2-4x-4x^5+x^3$. The "Polinom 2" field is empty. The buttons remain the same. The "Rezultat" field displays the derivative result: $-20.0x^4+3.0x^2+6.0x^1-4.0$.

Integrare



The screenshot shows a window titled "Calculator Polinoame". It has two input fields for polynomials. "Polinom 1" contains the expression $3X^2 - X + 2X^4 + 4$. "Polinom 2" is empty. Below the input fields are five buttons: "+", "-", "*", ".", and an integral symbol \int . The "Rezultat" field displays the integrated result: $+0.4X^5 + 1.0X^3 - 0.5X^2 + 4.0X^1$.

Eroare



The screenshot shows the same "Calculator Polinoame" window. "Polinom 1" still contains $3X^2 - X + 2X^4 + 4$. "Polinom 2" now contains the expression $4X + 4 + as - 3$. The buttons are the same. The "Rezultat" field now displays the error message "Introduceti un polinom!!!".

În cazul în care se introduce un polinom greșit în câmpul specific rezultatului va apărea mesajul „Introduceti un polinom!!!”.

7.Concluzii

Aceasta tema m-a ajutat sa aprofundez limbajul Java si paradigmele OOP si am invatat sa lucrez cu Regular Expressions. Ca si dezvoltare a aplicatie se poate implementa si operatia de impartire.

8.Bibliografie

1. https://www.tutorialspoint.com/java/java_regular_expressions
2. <https://www.regular-expressions.info>
3. <https://www.youtube.com>
- 4.Curs POO