

Tema 1

Calculator Polinoame

Student: Chiras Valentin-Fanica
Grupa: 30223
Profesor laborator: Dorin Moldovan

Cuprins

1. Cerinte functionale

.....
.....3

2. Obiective

.....
.....3

3. Analiza Problemei

.....
.....4

4. Proiectare

.....
..... 4

5. Test

.....
.....17

6. Posibilitati de dezvoltare si imbunatatire

.....
.....18

1. Cerințe Functionale

Propuneți, proiectați și implementați un sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi.

Operațiile pe care le suportă programul sunt:

- Citirea unui polinom de la tastatură sub forma de String cu formatul

$$a_n x^n + a_{n-1} x^{(n-1)} + \dots + a_1 x^1 + a_0 x^0$$

- Adunarea a două polinoame : $P(x) + Q(x)$
- Scaderea a două polinoame : $P(x) - Q(x)$
- Înmulțirea a două polinoame: $P(x) * Q(x)$
- Derivata unui polinom: $P'(x)$
- Aflarea valorii polinomului într-un punct anume: $P(x)$, x este un parametru introdus de la tastatură

2. Obiective

Obiectivul principal al proiectului este de a crea o aplicație care să implementeze un sistem de procesare a polinoamelor. Aceste polinoame sunt construite cu ajutorul unor termeni numiți **monoame**, care sunt alcătuite din 2 constante (coeficient și exponent).

Obiective secundare:

Obiectiv Secundar	Descriere	Capitol
Dezvoltarea de use case-uri și scenarii	Într-un sistem software un “use-case” este o listă de acțiuni care definesc în mod obișnuit interacțiunile dintre un rol (UML) și un sistem în atingerea unui obiect.	3
Alegerea structurilor de date	Structurile de date folosite pentru a duce la capăt obiectivul principal.	4.1
Împartirea pe clase	Folosirea MVC pentru a putea forma un GUI pentru clasele utilizate: Polinom, Monom etc...	4.2
Dezvoltarea algoritmilor	Vor fi descrise structurile de date necesare pentru atingerea obiectivului principal, schema UML precum și algoritmi folosiți pentru realizarea operațiilor.	4.3
Implementarea soluției	Vor fi descrise fiecare clasă cu variabilele și metodele importante precum și descrierea interfeței utilizator;	4.4
Testare	Vor fi descrise câteva scenarii de testare	5

3. Analiza Problemei

3.1 Use case-uri

Utilizarea programului presupune introducerea de către utilizator a celor două polinoame ce vor urma a fi folosite pentru efectuarea anumitor operații (adunare, scădere, înmulțire, derivare, find value). Rezultatul va fi afișat într-un nou JTextField după apăsarea unor Butoane ce au semnificativ operațiilor pe care le execută.

Deoarece e posibil ca unii utilizatori să introducă polinoame gresite, de fiecare dată când un polinom introdus în JTextField are un format greșit față de cel normal al unui polinom, la apăsarea butonului de operație va apărea o fereastră de dialog în care scrie faptul că formatul polinomului este de forma $a_n x^n + a_{n-1} x^{(n-1)} + \dots + a_1 x^1 + a_0 x^0$. Totodată dacă este introdus un polinom dezordonat, adică formatul este corect dar gradele sunt puse în ordine aleatoare, acest lucru nu va afecta rularea programului și va fi luat ca valid, iar în TextField-ul rezultatului va fi ordonat.

4. Proiectare

4.1 Alegerea structurilor de date

Principală structură de date folosită este un ArrayList de Monoame. Acest lucru se datorează faptului că un polinom este o înșiruire de monomae, acest lucru putând fi rezolvat foarte ușor cu un ArrayList.

```
List<Monom> polinom = new ArrayList<Monom>();
```

Fiecare Monom în parte este format dintr-un coeficient, o variabilă (x- în cazul de față) și un exponent (de exemplu: $3x^2$, unde 3 este coeficientul, x-variabilă, 2-exponentul).

Monomul va fi în Java o clasă separată care conține ca variabile instanță doi întregi, reprezentat de putere și coeficient.

4.2 Impartirea pe clase (MVC)

Prima clasă care a fost implementată este clasa pentru Monom(int coeficient, int putere). Această metodă are setere și getere pentru a fi ușor de utilizat pe viitor.

De asemenea există o metodă numită toString() care face Override la metoda toString() superclasa Object(). Această metodă are ca folosință transformarea polinomului final într-un string cât mai apropiat de formatul unui polinom. De exemplu dacă rezultatul operației efectuate pe polinoame rezultă coeficient 1 și putere 1 acesta va fi scris sub forma "x" dacă puterea este 0 atunci va fi scris doar coeficientul și dacă coeficientul este 0 acesta nu va mai fi scris deloc.

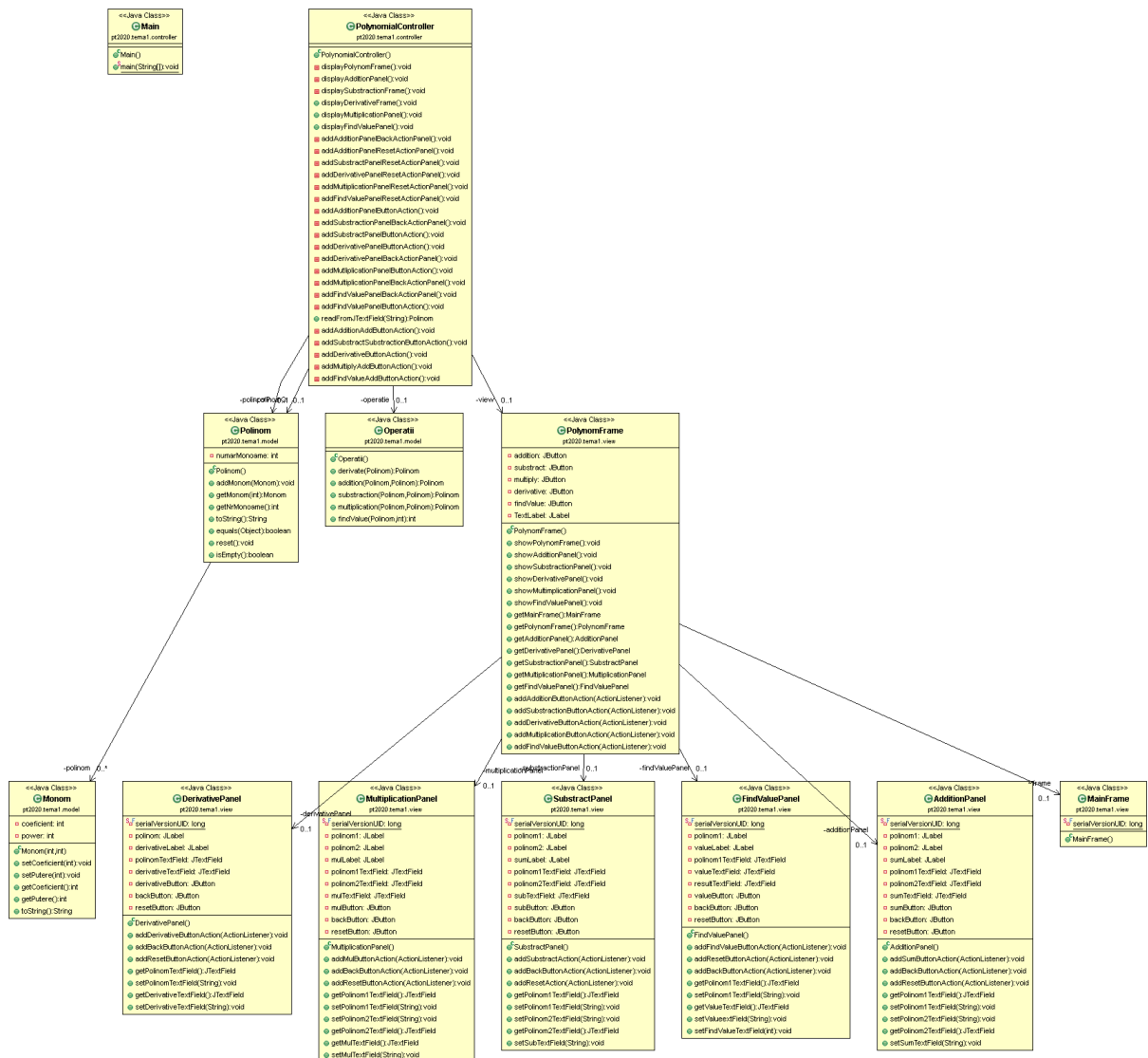
A doua clasă implementată a fost Polinom() care conține o listă de monoame.

Clasa Operatii() conține metode cu operațiile efectuate pe polinoame : adunare, scădere, înmulțire, derivare și aflarea valorii polinomului în punctul x.

Clasele de mai sus fac parte din Model. Clasele din partea de View sunt AdditionPanel(), DerivativePanel(), FindValuePanel(), MainFrame(), MultiplicationPanel(), PolynomFrame(), SubtractPanel() sunt clase ce creează pentru fiecare operație în parte un panou nou în care se va efectua operația dorită.

Clasa PolynomialController() face parte din componenta Controller din MVC care leagă Model și View și creează ActionListeneri pentru butoane.

4.3 Dezvoltarea algoritmilor



C clasele Monom si Polinom sunt folosite pentru a crea datele cu care vom lucra.

Clasa **Operatii()** este clasa care se ocupa cu prelucrarea datelor cum ne dorim: sa adunam, scadem, inmultim sau derivam polinoamele.

Prima metoda este metoda de derivare:

```
public Polinom derivate(Polinom p) {
    Polinom derivative = new Polinom();
    int i = 0;
    try {
        while (i < p.getNrMonoame()) {
            if (p.getMonom(i).getPutere() != 0) {
                Monom m = new Monom(p.getMonom(i).getCoeficient() *
p.getMonom(i).getPutere(),
                                p.getMonom(i).getPutere() - 1);
                derivative.addMonom(m);
            }
            i++;
        }
    } catch (Exception e) {
    }
    ;
    return derivative;
}
```

Aceasta metoda, ca toate celelalte din aceasta clasa contine un try-catch in care se calculeaza derivate Polinomului p introdus ca parametru. Am folosit o bucla **while** merge pe post de iteratoator si acceseaza fiecare monom existent in polinomul curent si-l atribuie monomului 'm', monom ce contine ca si coeficient rezultatul operatiei coeficient*putere, iar puterea se decrementeaza cu 1.

Aceste monom 'm' este adaugat in Polinomul 'derivative' la fiecare iteratie in parte, la final rezultand un alt polinom egal cu derivate polinomului introdus.

Metoda de inmultire:

```
public Polinom multiplication(Polinom p, Polinom q) {
    Polinom mul = new Polinom();

    int i = 0, j = 0;
    Monom m;
    try {
        while (i < p.getNrMonoame()) {
            while (j < q.getNrMonoame()) {
                m = new Monom(p.getMonom(i).getCoeficient() *
q.getMonom(j).getCoeficient(),
                                p.getMonom(i).getPutere() +
q.getMonom(j).getPutere());
                mul.addMonom(m);
                j++;
            }
            j = 0;
            i++;
        }
    } catch (Exception e) {
    }
    ;

    return mul;
}
```

Metoda de inmultire primeste doi parametri de tip Polinom(), mai exact Polinoamele care vor fi inmultite. Contine un bloc de tip try-catch in care se incearca cu ajutorul interatorilor i si j din while sa se inmulteasca fiecare monom in parte din cele doua polinoame. Astfel monumul 'm' primeste ca si coeficient rezultatul operatie de inmultire dintre cei doi coeficienti din monoamele lui p si q iar puterile se aduna, rezultat monamele necesare crearii polinomului 'mul' care este rezultatul inmultirii celor doua polinoame. Aceasta metoda poate fi imbunatatita deoarece acesta nu aduna la final monoamele care au aceasi putere astfel daca la un momenta dat polinmul final contine doua monoame cu aceasi putere aceste sunt scrise ca monoame diferite.

Metoda de adunare

```
public Polinom addition(Polinom p, Polinom q) {
    Polinom sum = new Polinom();
    int i = 0, j = 0;
    Monom m;
    try {
        while (i < p.getNrMonoame() && j < q.getNrMonoame()) {
            if (p.getMonom(i).getPutere() == q.getMonom(j).getPutere()) {
                m = new Monom(p.getMonom(i).getCoeficient() +
                    q.getMonom(j).getCoeficient(),
                        p.getMonom(i).getPutere());
                i++;
                j++;
            } else if (p.getMonom(i).getPutere() >
                q.getMonom(j).getPutere()) {
                m = new Monom(p.getMonom(i).getCoeficient(),
                    p.getMonom(i).getPutere());
                i++;
            } else {
                m = new Monom(q.getMonom(j).getCoeficient(),
                    q.getMonom(j).getPutere());
                j++;
            }
            sum.addMonom(m);
        }
        while (i < p.getNrMonoame()) {
            m = new Monom(p.getMonom(i).getCoeficient(),
                p.getMonom(i).getPutere());
            i++;
            sum.addMonom(m);
        }
        while (j < q.getNrMonoame()) {
            m = new Monom(q.getMonom(j).getCoeficient(),
                q.getMonom(j).getPutere());
            j++;
            sum.addMonom(m);
        }
    } catch (Exception e) {
    }
    ;

    return sum;
}
```

Aceasta metoda inainte de aduna coeficientii monoamelor verifica daca puterile sunt egale [**if** (p.getMonom(i).getPutere() == q.getMonom(j).getPutere())]. Daca aceasta conditie

este verificata este creat noul monom si este adugat in polinomul 'sum'. Daca nu se verifica care monom are puterea mai mare pentru a fi adaugat in noul polinom. La finalul executiei metode respective se obtine noul polinom care reprezinta suma celor doua polinoame

Metoda de scadere

```
public Polinom substraction(Polinom p, Polinom q) {
    Polinom subtraction = new Polinom();
    int i = 0, j = 0;
    Monom m;
    try {
        while (i < p.getNrMonoame() && j < q.getNrMonoame()) {
            if (p.getMonom(i).getPutere() == q.getMonom(j).getPutere()) {
                m = new Monom(p.getMonom(i).getCoeficient() -
                    q.getMonom(j).getCoeficient(),
                    p.getMonom(i).getPutere());
                i++;
                j++;
            } else if (p.getMonom(i).getPutere() >
                q.getMonom(j).getPutere()) {
                m = new Monom(p.getMonom(i).getCoeficient(),
                    p.getMonom(i).getPutere());
                i++;
            } else {
                m = new Monom(-q.getMonom(j).getCoeficient(),
                    q.getMonom(j).getPutere());
                j++;
            }
            subtraction.addMonom(m);
        }
        while (i < p.getNrMonoame()) {
            m = new Monom(p.getMonom(i).getCoeficient(),
                p.getMonom(i).getPutere());
            i++;
            subtraction.addMonom(m);
        }
        while (j < q.getNrMonoame()) {
            m = new Monom(-q.getMonom(j).getCoeficient(),
                q.getMonom(j).getPutere());
            j++;
            subtraction.addMonom(m);
        }
    } catch (Exception e) {
    }
    ;

    return subtraction;
}
```

Aceasta metoda este identica cu metoda de adunare doar ca coeficientii monomelor din polinomul q care nu au nici o putere egala cu monoamele din q sunt inmultite cu un '1'.

Metoda de Find Value

```
public int findValue(Polinom p, int valueOfx) {
    int i = 0;
    int value = 0;
    try {
        while (i < p.getNrMonoame()) {
            value += p.getMonom(i).getCoeficient() * Math.pow(valueOfx,
p.getMonom(i).getPutere());
            i++;
        }
    } catch (Exception e) {
    }
    ;
    return value;
}
```

Aceasta metoda nu este ceruta, de aceea voi fi scurt. Calculeaza $P(x)$ in punctul x trimis ca parametru "valueOfx"

Clasa de PolynomFrame() contine partea de grafica a aplicatie.

Metoda **showPolynomFrame()**:

```
public void showPolynomFrame() {
    frame.setTitle("Polynomial Calculator");
    frame.getContentPane().removeAll();
    frame.revalidate();
    frame.setLayout(new FlowLayout());
    frame.add(addition);
    frame.add(subtract);
    frame.add(derivative);
    frame.add(multiply);
    frame.add(findValue);
    TextLabel.setFont(new Font("BOSKO", Font.BOLD, 12));
    TextLabel.setForeground(Color.red);
    frame.add(TextLabel);
    JLabel image= new JLabel();
    image.setIcon(new
ImageIcon("C://Users//valen//Documents//pt2020_30223_chiras_valentin-
fanica_assignment_1//Tema1Project//mathematics.jpg"));
    frame.add(image);
    frame.setBounds(400, 100, 600, 550);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
}
```

Aceasta metoda cotine Panoul principal.

Polynomial Calculator
— □ ×

Addition Polynoms

Substract Polynoms

Derivative Polynom

Multiply Polynoms

Find Value

Polinoamele trebuie sa fiu de forma $a_n x^n + \dots + a_1 x^1 + a_0 x^0$

$$\frac{\partial}{\partial \theta} \int_{R_n} I'(x) f(x, \theta) dx = \int_{R_n} \frac{\partial}{\partial \theta} I'(x) f(x, \theta) dx$$

$$\sigma^2(\xi_1) = \frac{(\xi_1 - a)^2}{\sigma^2} f_{a, \sigma^2}(\xi_1)$$

Fiecare operatie in parte un panou separate. Codul urmator este pentru panoul in adunam

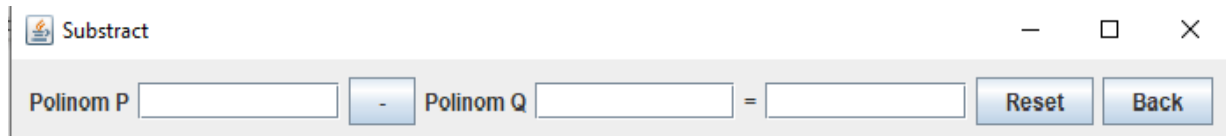
```
public AdditionPanel() {
    setLayout(new FlowLayout());
    polinom1 = new JLabel("Polinom P");
    add(polinom1);
    polinom1TextField = new JTextField(10);
    add(polinom1TextField);
    sumButton = new JButton("+");
    add(sumButton);
    polinom2 = new JLabel("Polinom Q");
    add(polinom2);
    polinom2TextField = new JTextField(10);
    add(polinom2TextField);
    sumLabel = new JLabel("=");
    add(sumLabel);
    sumTextField = new JTextField(10);
    add(sumTextField);
    resetButton = new JButton("Reset");
    add(resetButton);
    backButton = new JButton("Back");
    add(backButton);
    setVisible(true);
}
```

Polinoamele.



Codul este asemanator si pentru celelate panel-uri.

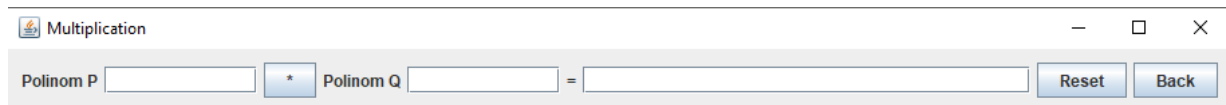
Scaderea Polinoamelor.



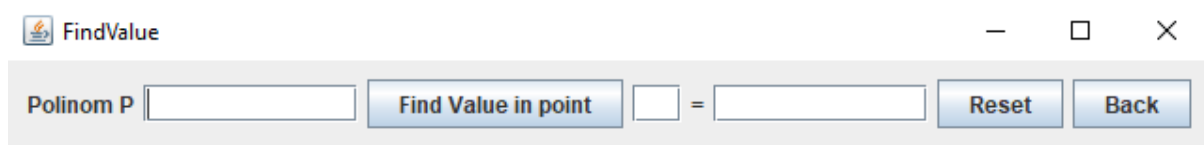
Derivarea Polinoamelor.



Inmultirea Polinoamelor.



Find Value



Una din clasele principale este **PolynomialController()**. Aceasta metoda leaga toate celelalte clase si tot odata aici se creaza ActionListener pentru fiecare buton.

```
private void addAdditionPanelButtonAction() {
    view.getPolynomFrame().addAdditionButtonAction(new
    ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayAdditionPanel();
        }
    });
}
```

Aceasta metoda este pentru butonul "Addition Polynoms" de pe panoul principal care ne duce in panoul rezervat adunarii.

Aceiasi logica este folosita si la celelalte butoane de pe panoul principal : "Subtract Polynoms", "Derivative Polynom", "Multiply Polynoms", "Find Value".

```
public Polinom readFromJTextField(String Polinom) {
    Polinom polinom = new Polinom();
    Monom m;
    int coefficient = 0;
    int power = 0;
    String[] numbersP = Polinom.replace("^",
    "").split("(?=\\+)|(?=\\-)|x");
    int i = 0;
    do {
        coefficient = Integer.parseInt(numbersP[i]);
        power = Integer.parseInt(numbersP[i + 1]);
        m = new Monom(coefficient, power);
        polinom.addMonom(m);
        i += 2;
    } while (i < numbersP.length);
    return polinom;
}
```

Aceasta metoda are scopul de a citi din TextFiled si a converti dintr-un String intr-un Polinom.

Acest lucru a fost posibil datorita Regex-ului problema la acesta metoda este faptul ca este necesara scrierea obligatorie a puterii si coeficientului. Mai exact "3x" trebuie scris "3x^1" iar '3' trebuie scris "3x^0".

```
private void addAdditionAddButtonAction() {
    view.getAdditionPanel().addSumButtonAction(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            try {
                polinomP.reset();
                polinomQ.reset();
                String stringPolinomP =
view.getAdditionPanel().getPolinom1TextField().getText().toString();
                String stringPolinomQ =
view.getAdditionPanel().getPolinom2TextField().getText().toString();
                polinomP = readFromJTextField(stringPolinomP);
                polinomQ = readFromJTextField(stringPolinomQ);

                view.getAdditionPanel().setSumTextField(operatie.addition(polinomP,
polinomQ).toString());

            } catch (Exception e1) {
                JOptionPane.showMessageDialog(new JFrame(),
                    "Polinoamele trebuie sa fiu de
forma anx^bn+...+a1x^1+a0x^0 ");
            }
        }
    });
}
```

Aceasta metoda este logica folosita pentru butonul "+" din panoul "Additional".

Aceasta metoda atribuie JTextField-ului pentru rezultat suma celor doua polinoame.

In cazul in care se introduce un format gresit apare o fereastră de dialog cu mesajul "Polinoamele trebuie sa fiu de forma $anx^{bn}+...+a_1x^1+a_0x^0$ ".

Aceasi logica este folosita si la celelalte metode pentru butoane singurul lucru care difera este operatiia utilizata.

Ficarii polinom ii este atribuit polinomul introdus de la tastatura cu ajutorul metodei "readFromJTextField". Acest lucru este valabil la toate metodele de acest gen

```
private void addSubtractSubtractionButtonAction() {
    view.getSubtractionPanel().addSubtractAction(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            try {
                polinomP.reset();
                polinomQ.reset();
                String stringPolinomP =
view.getSubtractionPanel().getPolinom1TextField().getText().toString();
                String stringPolinomQ =
view.getSubtractionPanel().getPolinom2TextField().getText().toString();
                polinomP = readFromJTextField(stringPolinomP);
                polinomQ = readFromJTextField(stringPolinomQ);

                view.getSubtractionPanel().setSubTextField(operatie.subtraction(polinomP,
polinomQ).toString());

            } catch (Exception e1) {
                JOptionPane.showMessageDialog(new JFrame(),
                    "Polinoamele trebuie sa fiu de forma
anx^bn+...+a1x^1+a0x^0 ");
            }
        }
    });
}
```

Aceasta metoda este logica folosita pentru butonul “-” din panoul “**Subtract**”.

```
private void addDerivativeButtonAction() {
    view.getDerivativePanel().addDerivativeButtonAction(new
ActionListener() {
        public void actionPerformed(ActionEvent e) {

            try {
                polinomP.reset();

                String stringPolinomP =
view.getDerivativePanel().getPolinomTextField().getText().toString();
                polinomP = readFromJTextField(stringPolinomP);

                view.getDerivativePanel().setDerivativeTextField(operatie.derivate(polin
omP).toString());

            } catch (Exception e1) {
                JOptionPane.showMessageDialog(new JFrame(),
                    "Polinoamele trebuie sa fiu de
forma anx^bn+...+a1x^1+a0x^0 ");
            }
        }
    });
}
```

Aceasta metoda este logica folosita pentru butonul "Derivate" din panoul "Derivative".

```
private void addMultiplyAddButtonAction() {
    view.getMultiplicationPanel().addMulButtonAction(new
    ActionListener() {
        public void actionPerformed(ActionEvent e) {

            try {
                polinomP.reset();
                polinomQ.reset();
                String stringPolinomP =
                view.getMultiplicationPanel().getPolinom1TextField().getText().toString();
                String stringPolinomQ =
                view.getMultiplicationPanel().getPolinom2TextField().getText().toString();
                polinomP = readFromJTextField(stringPolinomP);
                polinomQ = readFromJTextField(stringPolinomQ);

                view.getMultiplicationPanel().setMulTextField(operatie.multiplication(po
                linomP, polinomQ).toString());

            } catch (Exception e1) {
                JOptionPane.showMessageDialog(new JFrame(),
                "Polinoamele trebuie sa fiu de
                forma  $ax^bn+...+a1x^1+a0x^0$  ");
            }
        }
    });
}
```

Aceasta metoda este logica folosita pentru butonul "*" din panoul "Multiplication".

```
package pt2020.tema1.controller;

import static org.junit.Assert.*;

import org.junit.Test;

import pt2020.tema1.model.Monom;
import pt2020.tema1.model.Operatii;
import pt2020.tema1.model.Polinom;

public class PolynomialOperatoinTest {

    @Test
    public void test() {

        Polinom p = new Polinom();

        Monom m1 = new Monom(3,2);

        Monom m2 = new Monom(4,1);

        Monom m3 = new Monom(5,0);

        Operatii op =new Operatii();

        p.addMonom(m1);

        p.addMonom(m2);

        p.addMonom(m3);

        // p = 3x^2+4x+5

        Polinom q = new Polinom();

        Monom m4= new Monom(4,4);

        Monom m5= new Monom(3,2);

        Monom m6= new Monom(5,1);

        q.addMonom(m4);

        q.addMonom(m5);

        q.addMonom(m6);

        //q = 4x^4+3x^2+5x

        assertEquals("4x^4+6x^2+9x+5",op.addition(p, q).toString());

        assertEquals("-4x^4-x+5",op.substraction(p, q).toString());

        assertEquals(25,op.findValue(p, 2));

        assertEquals("16x^3+6x+5",op.derivate(q).toString());

    }

}
```


JUnit verifica rezultatul operatiilor cu Stringul introdus, String care reprezinta rezultatul sigur al operatiei respective.

7. Posibilitati de dezvoltare si imbunatatire/Concluzii

In primul rand la final mi-am dat seama ca poate m-am complicat facand pentru fiecare operatie un nou panou...

In aceasta tema am invatat in primul rand cum transform o problema in subprobleme tot mai simple de implementat ca la final sa ajung la rezolvarea problemei initiale.

Aplicatia facuta de mine poate fi imbunatatita deoarece lipsesc 2 operatii cea de integrare si si impartire. Metodele pentru operatii pot fi optimizate, in special din punct de vedere al cantitatii de cod.

8. Bibliografie

- <https://ro.wikipedia.org/wiki/Polinom>
- https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm
- <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>