DOCUMENTATIE

TEMA 1

NUME STUDENT: ERNST ROBERT
GRUPA: 30226

CUPRINS

1. Obiectivul temei

Obiectivul temei este de a dezvolta o aplicatie Java+Swing pentru a ajuta utilizatorul sa realizeze calculele basic ale unor polinoame. Pentru a indeplini obiectivul, este necesara creeri interfetei in Swing (text field-urile si butoanele), implementarea transformarii inputului de la text field la polinoame in structure de date convenabile, si creerea algoritmilor pentru a realiza calculele basic(adunare, scadere, inmultire, impartire, derivare, intergrare).

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

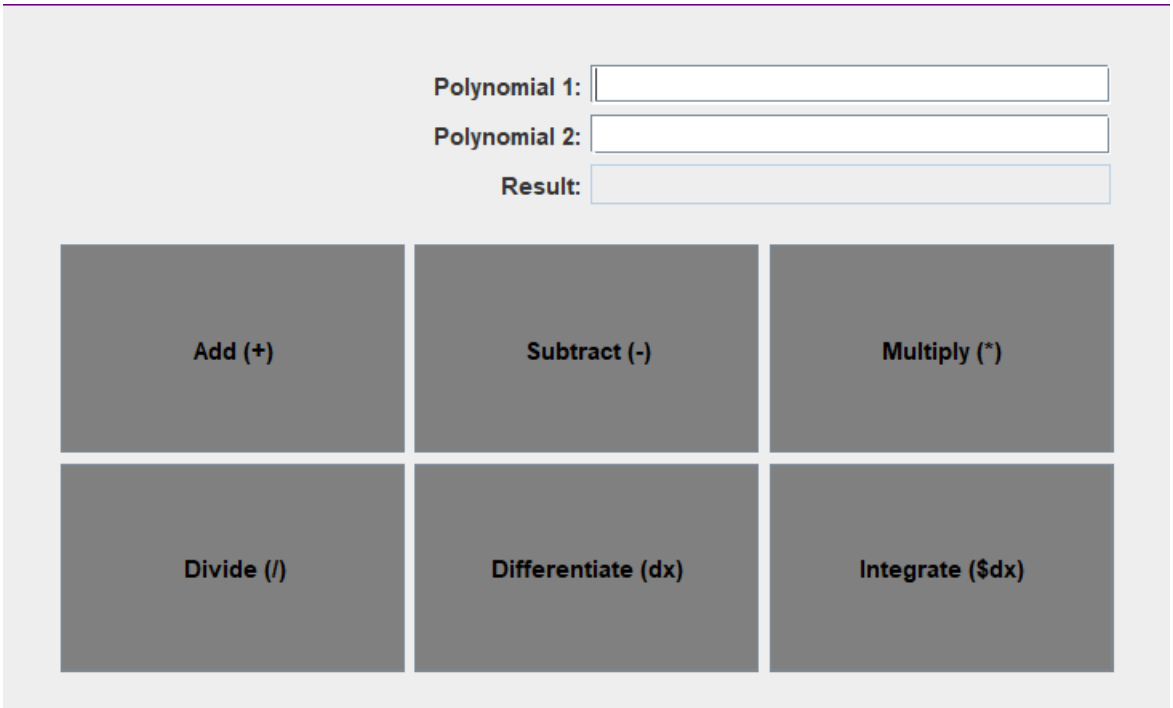
Utilizatorul va putea introduce polinoamele sub forma predefinita (a*x^2+b*x+c), in cele 2 text-field-uri, si dupa apasarea butonului de operatie dorit, in al treilea text-field marcat cu ‘Result’ se va afisa rezultatul corespunzator.

3. Proiectare

Pentru implementarea structurilor de date, am folosit o clasa Polynom, care continue un HashMap de tipul cheie<exponent, coefficient>, si gradul polinomului pentru a fi mai usor la afisare si la calcule.

Cealalta clasa, Monom, este folosita pentru a tine coefficientul si exponentul intr-un singur loc, in timpul folosirii calculelor.

Pentru operatii, am folosit o clasa Operations, unde folosim cate o functie statica pentru fiecare tip de calcul. In acelasi package, avem si o clasa pentru a converti inputul in polynom.



Pentru frontend, am folosit o clasa window, ce extinde JFrame, unde avem toate componentele cerute:

-titlul primului input

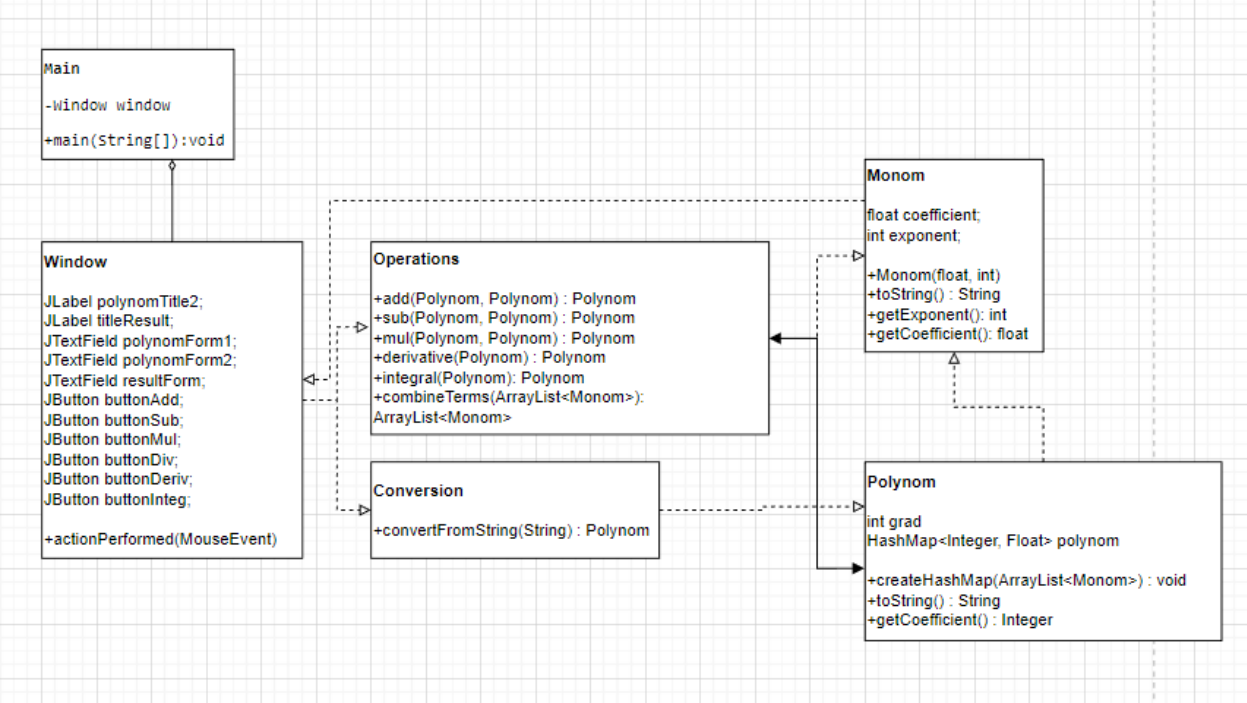
-text field pentru primul input

-titlul la al doilea input

-text field pentru al doilea input

-cele 6 butoane pentru operatii intr-un GridView de 2: 3

Structurile de date native folosite sunt ArrayList pentru stocarea monoamelor si HashMap<Integer, Float> pentru polynom;

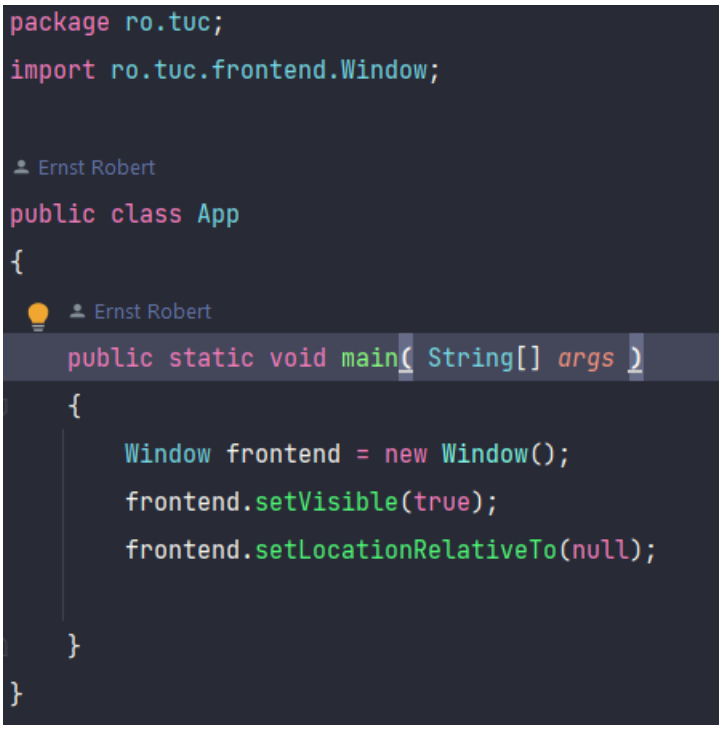


-diagrama UML a aplicatiei

4. Implementare

Clasa Main:

-este prima clasa pe care o executa compilerul si are doar referenta catre clasa Window, pentru a porni frontend-ul



Clasa Polynom:

-contine HashMap-ul discutat mai devreme cu toate datele matematice despre polynom

-contine grad-ul polynomului intr-un Integer

-contine o functie care populeaza HashMap-ul dintr-un ArrayList de monoame

-contine gettere, si toString

```java
package ro.tuc.model;

import java.util.ArrayList;
import java.util.HashMap;

25 usages  Ernst Robert
public class Polynom {
    2 usages
    private final HashMap<Integer, Float> polynom = new HashMap();
    3 usages
    private int grad=-1;
    6 usages  Ernst Robert
    public Polynom(ArrayList<Monom> monoms) { createHashMap(monoms); }
    1 usage  Ernst Robert
    private void createHashMap(ArrayList<Monom> monoms){
        for(Monom monom: monoms){
            polynom.put(monom.getExponent(), monom.getCoefficient());
            if(monom.getExponent() > grad){
                grad = monom.getExponent();
            }
        }
    }
```

```java
public Float getCoefficient(Integer exponent){
    return polynom.get(exponent);
}

Ernst Robert
@Override
public String toString() {
    int isFirst = 0;
    String toReturn = "";
    for(int i=getGrad();i>=0;i--){
        if(getCoefficient(i)!=null){
            isFirst++;
            Monom monom = new Monom(getCoefficient(i), i);
            if(isFirst!=1) toReturn+="+";

            toReturn+= monom.toString();
        }
    }
    return toReturn;
}

13 usages  Ernst Robert
public int getGrad() { return grad; }
```

Clasa Monom

-contine un int fiind exponentul

-contine un float fiind coefficientul

-contine gettere si toString

```java
package ro.tuc.model;

Ernst Robert
public class Monom {
    5 usages
    private final float coefficient;
    4 usages
    private final int exponent;

    20 usages  Ernst Robert
    public Monom(float coefficient, int exponent) {
        this.coefficient = coefficient;
        this.exponent = exponent;
    }
                                int exponent
                                PT2022_Test_Project
    4 usages  Ernst Robert
    public float getCoefficient() { return coefficient; }

    14 usages  Ernst Robert
    public int getExponent() { return exponent; }
    Ernst Robert
```

```java
4 usages  Ernst Robert
public float getCoefficient() { return coefficient; }

14 usages  Ernst Robert
public int getExponent() { return exponent; }
Ernst Robert
@Override
public String toString() {
    String exponent ="";
    if(this.exponent≠0 && this.exponent ≠ 1){
        exponent=String.valueOf(getExponent());
    }

    return (coefficient % 1 == 0 ? Integer.toString((int)coefficient ): coefficient)+(getExponent() == 0 ? "" : "x" )
            + (getExponent() ≠0 && getExponent() ≠ 1 ? "^" : "")
            + exponent;
}
```

Clasa Conversion

-contine o functie statica care converteste inputul intr-un ArrayList de monoame, si returneaza un Polynom nou

-folosim regex pentru a converti String-ul de input intr-un array de String

-parcurgem array-ul si verificam cele 3 conditii ale monom-ului: are grad 0, are grad 1 sau are grad>1

```
13 usages  Ernst Robert
public class Conversion {
    12 usages  Ernst Robert
    public static Polynom convertFromString(String input){
        String[] terms = input.split( regex: "(?=[-+])|\\s+");
        System.out.println(Arrays.toString(terms));
        ArrayList<Monom> monoms = new ArrayList<>();

        for (String term : terms) {
            Monom monom;
            if (term.contains("x^")) {
                String[] parts = term.split( regex: "\\*x\\^");
                monom = new Monom(Float.parseFloat(parts[0].trim()), Integer.parseInt(parts[1].trim()));

            }
            else if (term.contains("x")) {
                monom = new Monom(Float.parseFloat(term.replace( target: "*x",  replacement: "").trim()),  exponent: 1);
            }
            else {
                monom = new Monom( Float.parseFloat(term.trim()),  exponent: 0);
            }
            monoms.add(monom);
        }
        return new Polynom(monoms);
    }
}
```

Clasa Operations

-contine o functie add care ia doi Polynom si returneaza un al treilea ca rezultatul adunarii

-parcurgem intr-un loop cu doi parametri cei doi polinomi, si verificam care este mai mare. Daca nu sunt egali, va fi adaugat in ArrayList ul rezultat, monomul aferent iteratorului mai mare. Daca sunt egali, vom adauga un monom rezultat adunarii celor doua.Dupa parcurgem intr-un alt loop pentru fiecare iterator, in cazul in care gradele polinoamelor nu sunt egale

```
public static Polynom add(Polynom firstPolynom,Polynom secondPolynom){
    ArrayList<Monom> result = new ArrayList<Monom>();
    int i = 0;
    int j = 0;
    while (i ⩽ firstPolynom.getGrad() && j ⩽ secondPolynom.getGrad()) {
        if (i > j){
            if(firstPolynom.getCoefficient(i)≠null) result.add(new Monom(firstPolynom.getCoefficient(i), i));
            i++;
        } else if (j > i){
            if(secondPolynom.getCoefficient(j)≠null) result.add(new Monom(secondPolynom.getCoefficient(j), j));
            i++;
        } else {
            float firstCoefficient = firstPolynom.getCoefficient(i)≠null ? firstPolynom.getCoefficient(i) : 0;
            float secondCoefficient = secondPolynom.getCoefficient(j)≠null ? secondPolynom.getCoefficient(j) : 0;
            float sumOfCoefficients = firstCoefficient + secondCoefficient;
            if(sumOfCoefficients≠0) result.add(new Monom(sumOfCoefficients, i));
            i++;
            j++;
        }
    }
    while (i ⩽ firstPolynom.getGrad()) {
        if(firstPolynom.getCoefficient(i)≠null) result.add(new Monom(firstPolynom.getCoefficient(i), i));
        i++;
    }
    while (j ⩽ secondPolynom.getGrad()) {
        if(secondPolynom.getCoefficient(j)≠null) result.add(new Monom(secondPolynom.getCoefficient(j), j));
        j++;
    }
    return new Polynom(result);
```

-contine o functie sub care ia doi Polynom si returneaza un al treilea ca rezultatul scaderii

-folosim acelasi principiu ca la adunare, doar scadem in loc sa adunam monomii

```java
public static Polynom sub(Polynom firstPolynom, Polynom  secondPolynom){
    ArrayList<Monom> result = new ArrayList<Monom>();
    int i = 0;
    int j = 0;
    while (i <= firstPolynom.getGrad() && j <= secondPolynom.getGrad()) {
        if (i > j){
            if(firstPolynom.getCoefficient(i)!=null) result.add(new Monom(firstPolynom.getCoefficient(i), i));
            i++;
        } else if (j > i){
            if(secondPolynom.getCoefficient(j)!=null) result.add(new Monom(secondPolynom.getCoefficient(j), j));
            i++;
        } else {
            float firstCoefficient = firstPolynom.getCoefficient(i)!=null ? firstPolynom.getCoefficient(i) : 0;
            float secondCoefficient = secondPolynom.getCoefficient(j)!=null ? secondPolynom.getCoefficient(j) : 0;
            float sumOfCoefficients = firstCoefficient - secondCoefficient;
            if(sumOfCoefficients!=0) result.add(new Monom(sumOfCoefficients, i));
            i++;
            j++;
        }
    }
    while (i <= firstPolynom.getGrad()) {
        if(firstPolynom.getCoefficient(i)!=null) result.add(new Monom(firstPolynom.getCoefficient(i), i));
        i++;
    }
    while (j <= secondPolynom.getGrad()) {
        if(secondPolynom.getCoefficient(j)!=null) result.add(new Monom(secondPolynom.getCoefficient(j), j));
        j++;
    }
    return new Polynom(result);
```

-contine o functie mul care ia doi Polynom si returneaza un al treilea ca rezultatul inmultirii

-aici parcurgem in doua foruri, deoarece inmultirea se face pe fiecare dintre monomii celor doua polinoame

-apelam apoi combineTerms pentru a aranja polinomul rezultat

```java
1 usage  Ernst Robert
public static Polynom mul(Polynom firstPolynom, Polynom secondPolynom) {
    ArrayList<Monom> productPoly = new ArrayList<Monom>();

    for (int i=0; i<=firstPolynom.getGrad(); i++) {
        for (int j=0; j<=secondPolynom.getGrad(); j++) {
            float firstCoefficient = firstPolynom.getCoefficient(i) != null ? firstPolynom.getCoefficient(i) : 0f;
            float secondCoefficient = secondPolynom.getCoefficient(j) != null ? secondPolynom.getCoefficient(j) : 0f;

            float prodCoefficient = firstCoefficient * secondCoefficient;
            int prodExponent = i + j;
            productPoly.add(new Monom(prodCoefficient, prodExponent));
        }
    }

    combineTerms(productPoly);
    return new Polynom(productPoly);
}
```

-contine o functie derivative care ia un Polynom si returneaza un al doilea ca rezultatul derivarii

-contine o functie integral care ia un Polynom si returneaza un al doilea ca rezultatul integrarii

```java
1 usage  Ernst Robert
public static Polynom derivative(Polynom polynom){
    ArrayList<Monom> result = new ArrayList<Monom>();

    for(int i=0; i<= polynom.getGrad(); i++){
        if(polynom.getCoefficient(i)!=null){
            Monom monom = new Monom(polynom.getCoefficient(i), i);
            result.add(new Monom( coefficient: monom.getCoefficient()* monom.getExponent(),  exponent: monom.getExponent()-1));
        }
    }

    return new Polynom(result);
}

1 usage  Ernst Robert
public static Polynom integral(Polynom polynom){
    ArrayList<Monom> result = new ArrayList<Monom>();
    for(int i=1; i<= polynom.getGrad(); i++){
        if(polynom.getCoefficient(i) != null){
            Monom monom = new Monom(polynom.getCoefficient(i), i);
            result.add(new Monom( coefficient: 1/((monom.getExponent()+1)* monom.getCoefficient()),  exponent: monom.getExponent()+1));
        }
    }
    return new Polynom(result);
}
```

-contine o functie combineTerms care ia un arrayList de Monom dezordonat si il reordoneaza dupa exponent, folosind un Map pentru a stoca monoamele si fiind un helper al inmultirii.

```
1 usage   Ernst Robert
public static void combineTerms(ArrayList<Monom> polynom) {
    Map<Integer, Float> map = new HashMap<>();
    for (Monom mono : polynom) {
        int exponent = mono.getExponent();
        float coefficient = mono.getCoefficient();
        if (map.containsKey(exponent)) {
            map.put(exponent, map.get(exponent) + coefficient);
        }
        else {
            map.put(exponent, coefficient);
        }
    }

    polynom.clear();

    for (Map.Entry<Integer, Float> entry : map.entrySet()) {
        int exponent = entry.getKey();
        float coefficient = entry.getValue();
        if (coefficient != 0) {
            polynom.add(new Monom(coefficient, exponent));
        }
    }

    Collections.sort(polynom, (Comparator) (monom1, monom2) -> {
            return monom2.getExponent() - monom1.getExponent();
    });
}
```

Si Clasa Window

-contine totul legat de frontend (2 panel-uri, unul pentru butoane, unul pentru textfield-uri)

-contine o functie action listener pentru onClick performed, unde se apeleaza functiile aferente operatiilor din Operations file.

```
public class Window extends JFrame implements ActionListener
    2 usages
    private final JLabel polynomTitle1;
    2 usages
    private final JLabel polynomTitle2;
    2 usages
    private final JLabel titleResult;
    3 usages
    private final JTextField polynomForm1;
    3 usages
    private final JTextField polynomForm2;
    8 usages
    private final JTextField resultForm;
    6 usages
    private final JButton buttonAdd;
    6 usages
    private final JButton buttonSub;
    6 usages
    private final JButton buttonMul;
    6 usages
    private final JButton buttonDiv;
    6 usages
    private final JButton buttonDeriv;
    6 usages
    private final JButton buttonInteg;
    1 usage   Ernst Robert
    public Window() {
        setTitle("Polynom Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
JPanel inputPanel = new JPanel(new GridLayout( rows: 3,  cols: 2,  hgap: 5,  vgap: 5));
inputPanel.setBorder(BorderFactory.createEmptyBorder( top: 20,  left: 20,  bottom: 10,  right: 20));
inputPanel.setBackground(new Color( r: 238,  g: 238,  b: 238));

polynomTitle1 = new JLabel( text: "Polynomial 1:", SwingConstants.RIGHT);
inputPanel.add(polynomTitle1);
polynomForm1 = new JTextField( columns: 20);
inputPanel.add(polynomForm1);
polynomTitle2 = new JLabel( text: "Polynomial 2:", SwingConstants.RIGHT);
inputPanel.add(polynomTitle2);
polynomForm2 = new JTextField( columns: 20);
inputPanel.add(polynomForm2);
titleResult = new JLabel( text: "Result:", SwingConstants.RIGHT);
inputPanel.add(titleResult);
resultForm = new JTextField( columns: 20);
resultForm.setEditable(false);
inputPanel.add(resultForm);

JPanel buttonPanel = new JPanel(new GridLayout( rows: 2,  cols: 3,  hgap: 5,  vgap: 5));
buttonPanel.setBorder(BorderFactory.createEmptyBorder( top: 10,  left: 20,  bottom: 20,  right: 20));
buttonPanel.setBackground(new Color( r: 238,  g: 238,  b: 238));
buttonAdd = new JButton( text: "Add (+)");
buttonAdd.setBackground(Color.GRAY);
buttonAdd.setForeground(Color.BLACK);
buttonPanel.add(buttonAdd);

buttonSub = new JButton( text: "Subtract (-)");
buttonSub.setBackground(Color.GRAY);
buttonSub.setForeground(Color.BLACK);
```

```
buttonMul = new JButton( text: "Multiply (*)");
buttonMul.setBackground(Color.GRAY);
buttonMul.setForeground(Color.BLACK);
buttonPanel.add(buttonMul);

buttonDiv = new JButton( text: "Divide (/)");
buttonDiv.setBackground(Color.GRAY);
buttonDiv.setForeground(Color.BLACK);
buttonPanel.add(buttonDiv);

buttonDeriv = new JButton( text: "Differentiate (dx)");
buttonDeriv.setBackground(Color.GRAY);
buttonDeriv.setForeground(Color.BLACK);
buttonPanel.add(buttonDeriv);

buttonInteg = new JButton( text: "Integrate ($dx)");
buttonInteg.setBackground(Color.GRAY);
buttonInteg.setForeground(Color.BLACK);
buttonPanel.add(buttonInteg);

JPanel panel = new JPanel(new BorderLayout());
panel.setBorder(BorderFactory.createEmptyBorder( top: 10,  left: 10,  bottom: 10,  right: 10));
panel.setBackground(new Color( r: 238,  g: 238,  b: 238));

panel.add(inputPanel, BorderLayout.PAGE_START);
panel.add(buttonPanel, BorderLayout.CENTER);

add(panel);
```

```
    buttonAdd.addActionListener( l: this);
    buttonSub.addActionListener( l: this);
    buttonMul.addActionListener( l: this);
    buttonDiv.addActionListener( l: this);
    buttonDeriv.addActionListener( l: this);
    buttonInteg.addActionListener( l: this);

    setVisible(true);
}
```

Ernst Robert

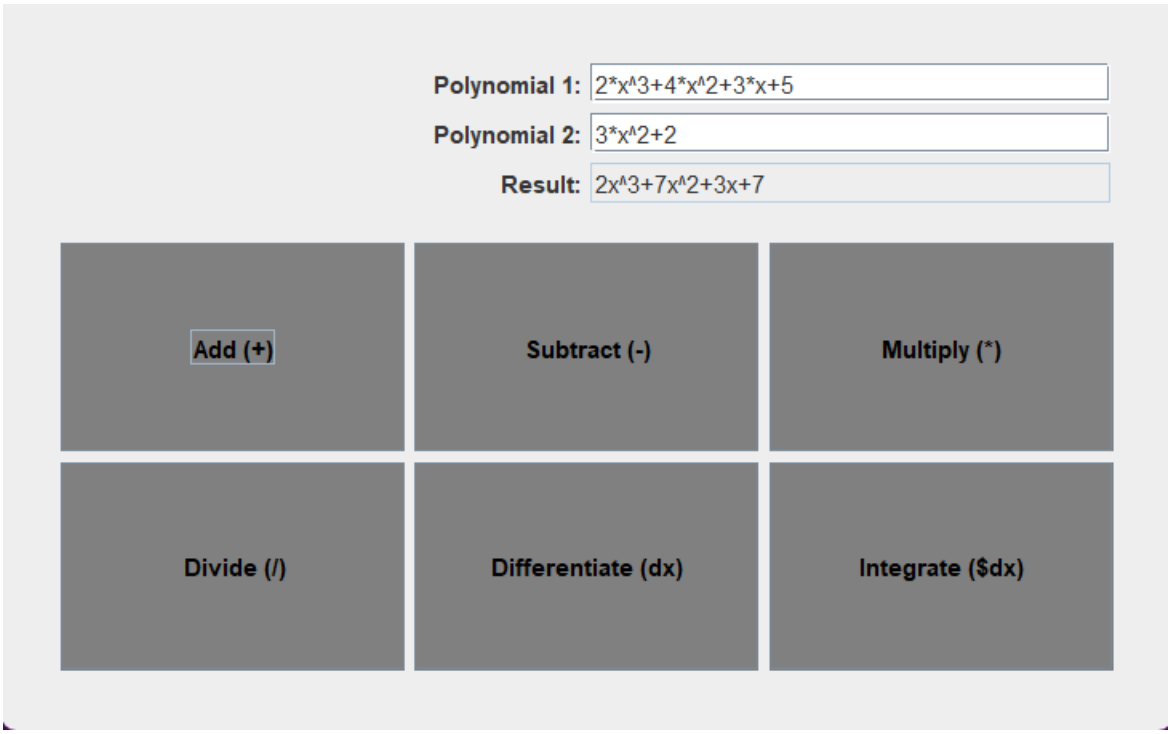
```
public void actionPerformed(ActionEvent e) {
    String input1 = polynomForm1.getText().trim();
    String input2 = polynomForm2.getText().trim();

    Polynom polynom1;
    Polynom polynom2;

    Object source = e.getSource();
    if (buttonAdd.equals(source)) {
        polynom1 = Conversion.convertFromString(input1);
        polynom2 = Conversion.convertFromString(input2);

        resultForm.setText(Operations.add(polynom1, polynom2).toString());
    }
    else if (buttonSub.equals(source)) {
        polynom1 = Conversion.convertFromString(input1);
        polynom2 = Conversion.convertFromString(input2);
```

```
        }
        else if (buttonMul.equals(source)) {
            polynom1 = Conversion.convertFromString(input1);
            polynom2 = Conversion.convertFromString(input2);

            resultForm.setText(Operations.mul(polynom1, polynom2).toString());
        }
        else if (buttonDiv.equals(source)) {
            polynom1 = Conversion.convertFromString(input1);
            polynom2 = Conversion.convertFromString(input2);
            //TODO division operation
        }
        else if (buttonDeriv.equals(source)) {
            polynom1 = Conversion.convertFromString(input1);
            polynom2 = Conversion.convertFromString(input2);

            resultForm.setText(Operations.derivative(polynom1).toString());
        }
        else if (buttonInteg.equals(source)) {
            polynom1 = Conversion.convertFromString(input1);
            polynom2 = Conversion.convertFromString(input2);

            resultForm.setText(Operations.integral(polynom1).toString());
        }
    }
}
```

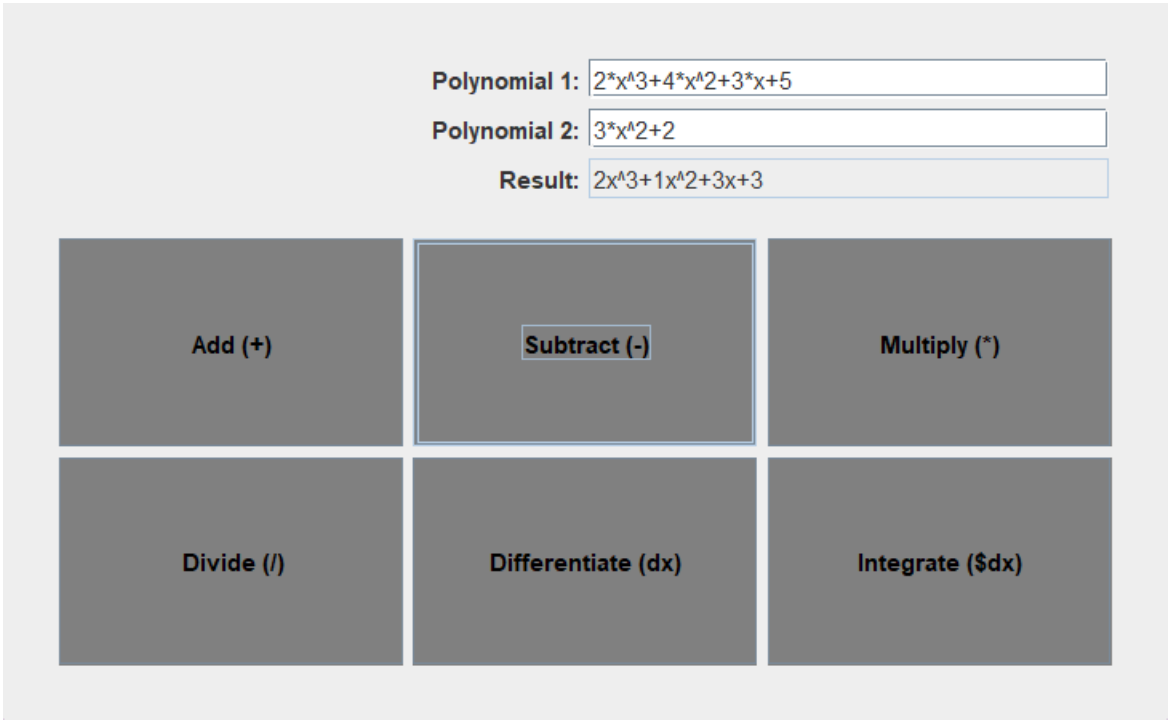
5. Rezultate

Vom testa aplicatia pentru fiecare operatie.

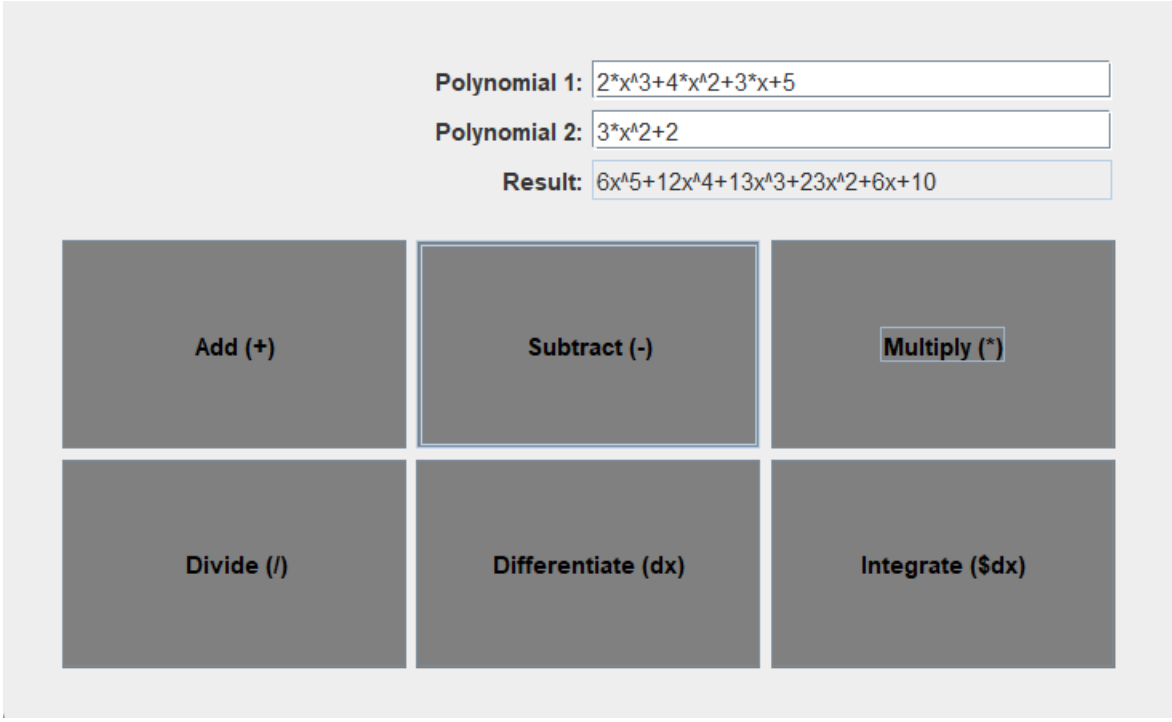
Adunarea:



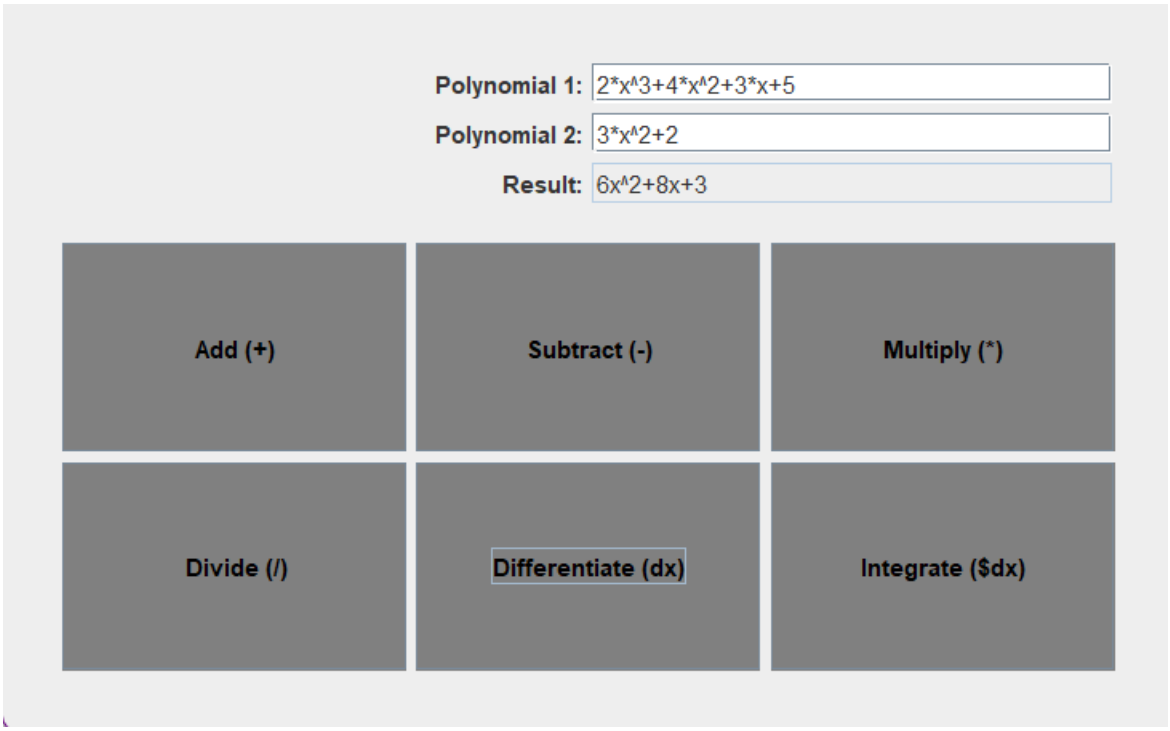
Scaderea:



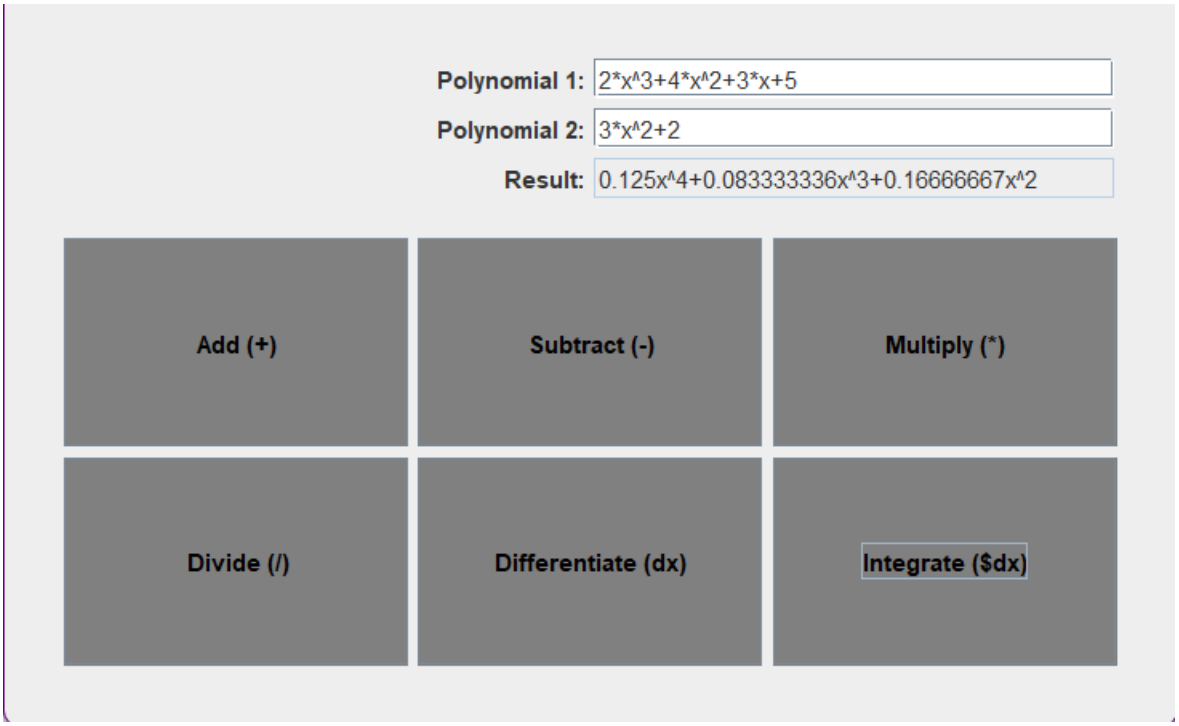
Inmultirea:



Derivarea primului polynom:



Integrarea primului polinom



Daca inputul nu respecta conditiile pentru regex, programul va arunca o eroare, iar textfield-ul de output este blocat pentru a nu putea interfera cu logica programului.

6. Concluzii

Proiectul a fost o problema challenging, la care este nevoie de o gandire matematica, si o vedere logica asupra sub-problemelor impuse de acesta, mai ales la debugging.

O posibila dezvoltare este aceea de a implmenta si impartirea, inputuri complexe sau mixte, si un frontend mai frumos, scris intr-un framework dedicate, e.g. React.

7. Bibliografie

Exemplu de proiect Java Swing: https://www.javatpoint.com/java-swing

Intelegere mai profunda a polinoamelor: https://en.wikipedia.org/wiki/Polynomial

Diagram Maker: https://app.diagrams.net/