DOCUMENTATIE

TEMA *1*
CALCULATOR DE POLINOAME

NUME STUDENT: Dimitriu David
GRUPA: 30218

CUPRINS

1. Obiectivul temei 3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3
3. Proiectare 3
4. Implementare 3
5. Rezultate 3
6. Concluzii 3
7. Bibliografie 3

1. Obiectivul temei

Obiectivul temei este acela ce a implementa un calculator de polinoame cu o interfața grafica dedicata in care utilizatorul poate insera polinoamele(string-uri) ,alege operația dorită(adunare, scădere, înmulțire, împărțire, derivare sau integrare) iar mai apoi sa vadă rezultatul într-un text field.

Vom folosi arhitectura mvc (Model,View,Controller) pentru implementarea temei. In pachetul models am implementat clasele Monomial si Polynomial, în View avem view-ul cu care utilizatorul interacționează atunci când folosește aplicația iar in Controller facem legătura intre Moldel si View.

Cerințele proiectului sunt :

Requirement	**Grading**
• Use an object-oriented programming design (use encapsulation, define • appropriate classes as a result of problem decomposition such as *Polynomial* and *Monomial*) • Use lists instead of arrays • Use *foreach* instead of *for*(*int i=0*…) • Implement a graphical user interface using Java Swing or JavaFX • Implement the addition and subtraction operation • Implement classes with maximum 300 lines (except the UI classes) and methods with maximum 30 lines • Use the Java naming conventions (see link) • Good quality documentation addressing all sections from the documentation structure.	5 points (**Minimum to pass**)
Use an architectural pattern (e.g., Model View Controller)	1 point
Implement the multiplication operation	0.5 points
Implement the division operation	1 point
Implement the derivative operation	0.5 points
Implement the integration operation	0.5 points

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

In fond, cerințele proiectului sunt implementarea operațiilor din cerința. Se considera ca input-ul sa fie corect. Consideram input corect polinomul 2x^3+4x^2+5x^1+6x^0 (puterea este reprezentata de ^ ,iar string-ul trebuie sa fie introdus fără spatii).In cazul in care polinomul nu este introdus corect, se va afișa un pop-up in care i se spune utilizatorului unde a greșit.

Este necesar urmarea următorilor pași :

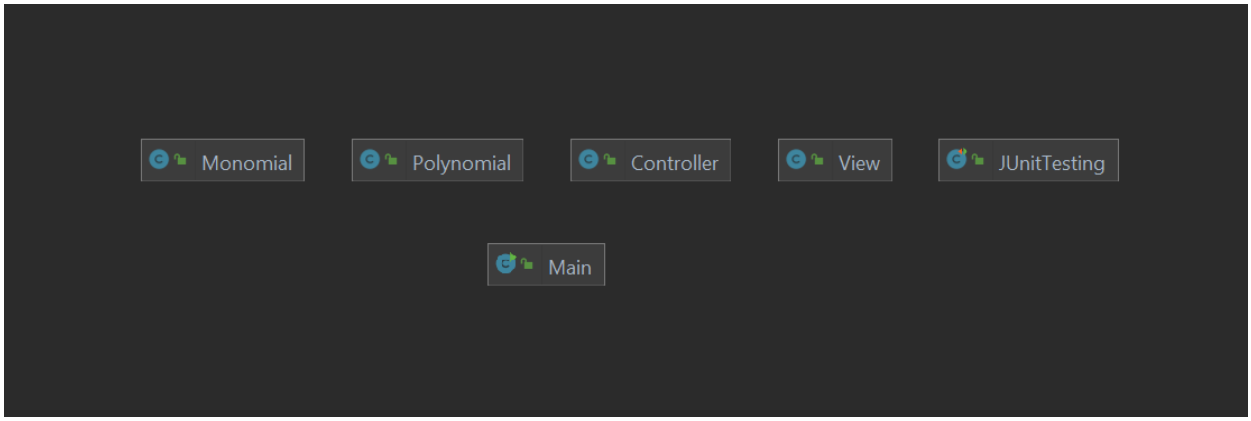
1) Introducerea primul polinom in text filed-ul corespunzător.
2) Introducerea celui de al doilea polinom in al doilea text field (excepție pentru operația de derivare si integrare in care vom folosi primul câmp).
3) Alegerea operației dorite, apăsând pe butonul corespunzător.
4) Afișarea rezultatului in text field ul destinat afișării.

Nu este necesar introducerea polinoamelor de fiecare data când dorim sa facem alta operație. Utilizatorul apasă pe butonul care reprezintă operația iar rezultatul se va schimba in funcție de asta.

După ce utilizatorul se folosește de aplicație poate închide apăsând pe label-ul x care va provoca un pop up de verificare .Rularea programului se oprește de la sine, nefiind nevoie sa fie închisa din Intelijj,

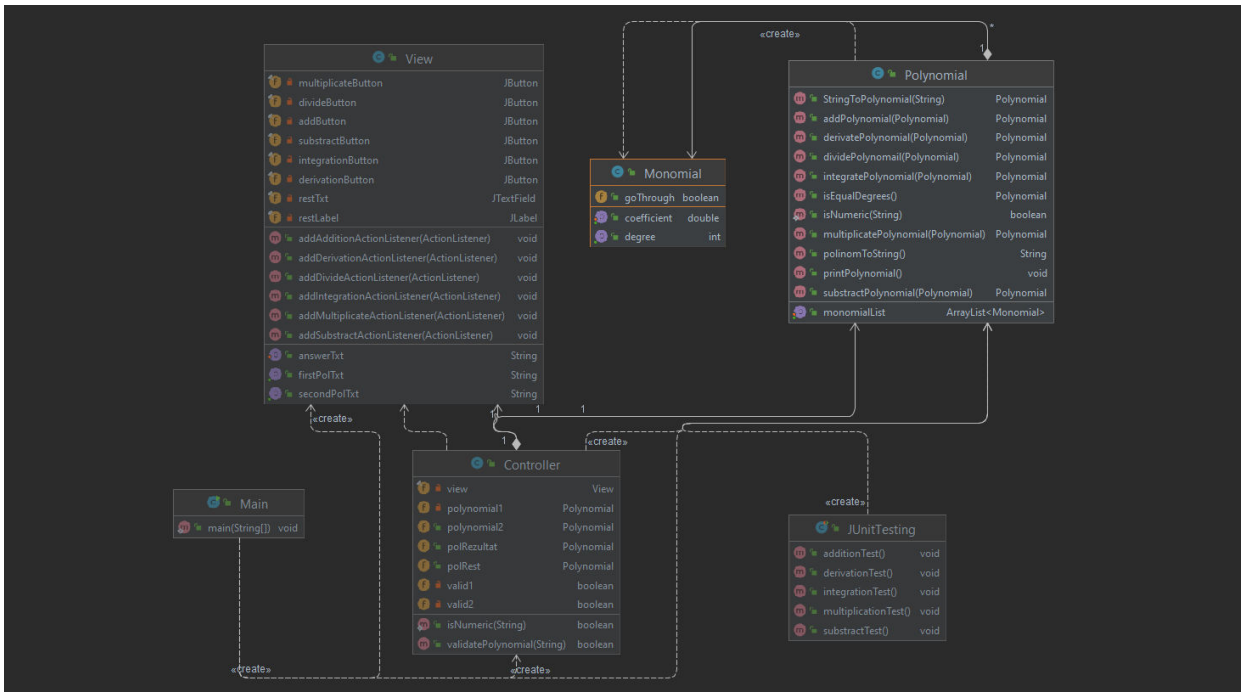
3. Proiectare

Proiectul este structurat in pachetele models, view si controller. Pentru realizarea calculatorului de polinoame a fost necesar implementarea următoarelor clase : Monomial , Polynomial din pachetul models , Controller din pachetul controller , view din pachetul View si clasa Main cu care simulam aplicația.

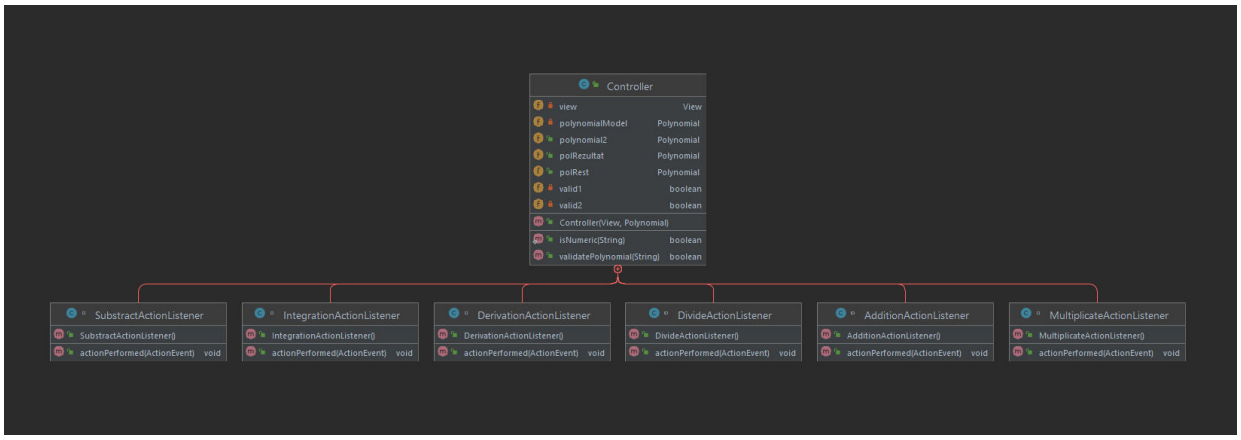


Pentru implementarea polinomului am folosit structura de date ArrayList,fiecare polinom este o lista de monoame.

Următoarea figură prezintă principalele proprietăți și metode ale unei clase printr-o diagramă UML, evidențiind în același timp conexiunile dintre ele:



Pentru implementarea controller-ului ne-am folosit de inner clasele cu care am facut legatura cu butoanele din view,am implementat functionalitatile butoanelor :



4. Implementare

Vom prezenta fiecare clasa cu câmpuri si metodele importante. Se va descrie implementarea interfeței utilizator.

Clasa Monomial are următoarele atribute : coefficient de tip double ,fiind private care reprezintă coeficientul monomului si degree de tip int , tot private, care reprezintă gradul monomului(puterea lui x).Mai apoi avem constructorul clasei ce are ca rol instanțierea unui monom cu atributele date. Metodele clasei sunt : public double getCoefficinet() care are ca rol returnarea coeficientului monomului, public void setCoefficient() pentru setarea coeficientului si public int getDegree() cu care obtinem gradul monomului.

Clasa Polynomial continue un singur atribut : rivate ArrayList<Monom> monomialList, care evidentiaza asocierea dintre cele doua clase din pachetul models,fiecare polinom este alcatuit din unul sau mai multe monoame.

Metodele clasei sunt : public ArrayList<Monomial> getMonomialList() care returneaza monoamele polinomului, public void setMonomialList(ArrayList<Monomial> monomialList) care seteaza monoamele polinomului.

In clasa Polynomial am implementat urmatoarele operatii :

1) Adunarea

Metoda public Polynomial addPolynomial(Polynomial polynomial) are rolul de a aduna polinomul polynomial la polinomul current.Cele doua polinoame vor fi parcurse de cate un for in care vom testa ,monom cu monom daca gradul monomului din polinomul curent este egal cu gradul monomlui din al doilea monom. In caz afirmativ,se vor aduna coeficientii monoamelor. In cazul in care exponentul primului polinom va fi mai mic decat exponentul celui de al doilea polinom si monomul din polinomul doi nu a fost adunat pana atunci se va efectua adaugarea in polinomul rezultat a unui monom cu cu coeficientul si exponetul celui de-al doilea monom. La iesirea din cele doua bucle va fi returnat polinomul rezultat.

Codul de la metoda care implementeaza adunarea :

```
public Polynomial addPolynomial(Polynomial polynomial) {

        Polynomial answer = new Polynomial();   //am creat un nou polinom care va fi raspunsul adunarii

        Polynomial expEgali;

        expEgali = this.isEqualDegrees();

        this.setMonomialList(expEgali.getMonomialList());


        for (Monomial temp : monomialList) {

            for (Monomial temp2 : polynomial.monomialList) {

                if (temp.getDegree() == temp2.getDegree()) { //daca polinoamele au gradele egale

                    answer.monomialList.add(new Monomial(temp.getCoefficient() + temp2.getCoefficient(), temp.getDegree())); //adunam coeficientii si gradul ramane la fel

                    temp2.goThrough = false;

                    temp.goThrough = false;

                    break;

                } else if (temp.getDegree() < temp2.getDegree() && temp2.goThrough) {      //daca gradul primului polinom este mai mic decat gradul celui de al doilea

                    answer.monomialList.add(new Monomial(temp2.getCoefficient(), temp2.getDegree()));      //luam gradul mai mare

                    temp2.goThrough = false;

                } else if (temp.getDegree() > temp2.getDegree() && temp2.goThrough) {      //invers

                    answer.monomialList.add(new Monomial(temp.getCoefficient(), temp.getDegree()));

                    temp.goThrough = false;
```

```
                    break;
                }
            }
        }
        return answer;
    }
```

2) Diferenta

Metoda public Polynomial substractPolynomial(Polynomial polynomial) are rolul de a face diferenta intre doua polinoame.Ne vom creea un polinom answer in care vom retine rezultatul diferentei iar apoi parcurgem doar al doilea polinom caruia ii setam coeficientii (-) iar mai apoi apelam operatia de adunare pe primul polinom.Pasam informatia catre polinomul answer iar apoi il returnam.

3) Inmultirea

Metoda public Polynomial multiplicatePolynomial(Polynomial polynomial) are rolul de a inmulti doua polinoame.Aici parcurgem ambele polinoame si inmultim fiecare cu fiecare coeficinet si adunam gradul respetiv. La iesirea din cele doua bucle for, vom apela metoda verificareExponentiEgali() pe polinomul answer care are ca rol adunarea monoamelor de grad egal din polinomul rezultat. La sfarsit polinomul rezultat va fi returat.

4) Derivare

Metoda public Polynomial derivatePolynomial(Polynomial polynomial) are ca rol derivarea unui polinom.Vom parcurge polinomul si coeficientul nou este gradul monomului inmultit cu coeficientul iar gradul va scadea cu 1.

5) Integrare

Metoda public Polynomial integratePolynomial(Polynomial polynomial) are ca rol integrarea unui polinom.In aceasta parcurgem polinomul si coeficientul e reprezentat de coeficient impartit la gradul monomului +1 ((temp.getCoefficient() /

(temp.getDegree() + 1)) iar gradului i se va aduna 1.

In clasa Polynomial mai avem metodele: public void printPolynomial() care are ca rol afisarea in consola a unui polinom, public Polynomial isEqualDegrees() in care verificam gradele polinomului sa fie egale si creăm un nou polinom cu gradele monoamelor in ordine, metoda public Polynomial StringToPolynomial(String polynomString) care are ca rol convertirea unui string in polynomial.Aici am preluat codul de pe site-ul stackoverflow. Am folosit regex pentru separarea string-ului. Metoda returnează polinomul answer. Metoda public static boolean isNumeric(String str) este folosita in StringToPolynomial si verifica daca un string este numeric sau nu.

Clasa View conține atributele : private JPanel, contentPane , private JTextField firstPolTxt, private JTextField secondPolTxt, private JButton multiplicateButton, private JButton divideButton, private JButton addButton, private JButton substractButton, private JButton integrationButton, private JButton derivationButton, private JTextField answerTxt, private JTextField restTxt, private JLabel restLabel.FirstPolTxt reprezinta campul in care va fi introdus primul polinom ,respectiv pentru secondPolTxt. Restul sunt butoanele corespunzătoare fiecărei operații.

Aici avem metodele public String getFirstPolTxt(), public String getSecondPolTxt(), public void setAnswerTxt(String text) cu care ne luam datele pe de view, respetiv setam câmpul in care vom afișa răspunsul la operația selectata.

Clasa View mai conține pentru fiecare buton cate o metoda ActionListener cu care facem legătura cu controller-ul. (exemplu pentru adunare : public void addMultiplicateActionListener(ActionListener actionListener)).

Am implementat o funcționalitate pentru butonul X, care afișează un pop up si întreabă utilizatorul daca este sigur ca vrea sa părăsească aplicația.

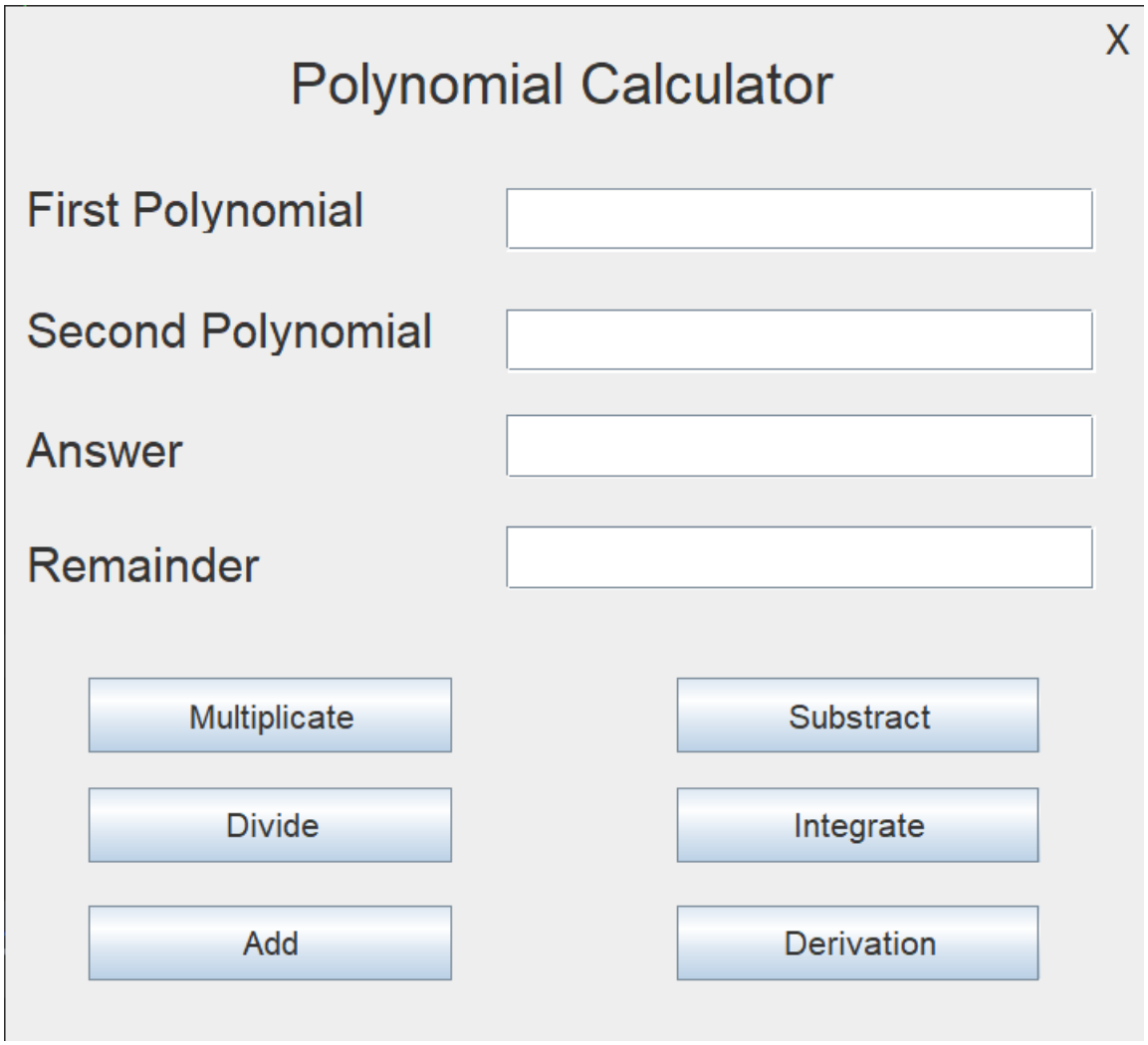
```
JLabel lblX = new JLabel("X");
        lblX.addMouseListener(new MouseAdapter() {
            @Override
```

```
            public void mouseClicked(MouseEvent e) {
                if (JOptionPane.showConfirmDialog(null, "Are you sure you want to close this application?", "Confirmation", JOptionPane.YES_NO_OPTION) == 0)
                    View.this.dispose();
            }

            @Override
            public void mouseEntered(MouseEvent e) {
                lblX.setForeground(Color.RED);
            }

            @Override
            public void mouseExited(MouseEvent e) {
                lblX.setForeground(Color.WHITE);
            }
        });
```

Utilizatorul va interacționa cu acest view :



Clasa Controller conține atributele view de tip View, polynomial1 , polynomial2(polinoamele introduse de utilizator in view), polRezultat, polRest, si valid1 si valid2 de tip boolean cu care verificam pentru fiecare câmp daca polinomul a fost introdus corect.

Constructorul clasei are rolul de a trea un obiect de timpul Controller.Aici legam si butoanele din view cu inner class-urile din controller, inner clase care dețin logic fiecărui buton.

Constructorul clasei Controller :

```
public Controller(View view, Polynomial polynomial1) {
```

```
        this.view = view;
        this.polynomial1 = polynomial1;
        this.polynomial2 = new Polynomial();
        this.polRezultat = new Polynomial();
        this.polRest = new Polynomial();
        this.view.addMultiplicateActionListener(new MultiplicateActionListener));
        this.view.addDivideActionListener(new DivideActionListener());
        this.view.addAdditionActionListener(new AdditionActionListener());
        this.view.addSubstractActionListener(new SubstractActionListener());
        this.view.addIntegrationActionListener(new IntegrationActionListener());
        this.view.addDerivationActionListener(new DerivationActionListener());

    }
```

După cum am menționați clasa Controller am implementat logica din spatele fiecărui buton cu care este selectata operația dorita de catre utilizator.
Spre exemplu, class MultiplicateActionListener implements ActionListener este inner clasa pentru butonul de inmultire in care avem doua string-uri,pol1 si pol2 in care reținem cele doua polinoame de pe view iar mai apoi transformam string-urile in polinoame pentru a putea apela metoda de înmulțire din clasa Polynomial(metoda multiplicatePolynomial este de tip Polynomial). Verificam prin cele 2 atribute,valid1 si valid2 daca input-ul este introdus corect, daca da,in polRezultat salvam rezultatul operației de inmultire, apoi in string-ul polRezString reținem polinomul sub forma de text si setam text field-ul in care afisam rezultatul cu acel polinom. In caz contrar(unul dintre valid1 sau valid2 e fals,a fisam un pop up in care spunem utilizatorului care polinom l-a introdus greșit.
Asemănător facem pentru toate butoanele, schimbam doar metoda pentru fiecare operație in parte. O mica diferența ar fi la integrare si derivare in care ne luam datele doar de pe primul text field si apelam metoda cu el însuși.

Tot in Controller avem metoda public boolean validatePolynomial(String

input) care seteaza atributele valid1 si valid2.Funtionalitatea ei este de a verifica input-ul introdus de user.

Exemplu de inner class folosita in clasa Controller pentru a implementa funcționalitatea butonului ADD.

```
class AdditionActionListener implements ActionListener {


    @Override
    public void actionPerformed(ActionEvent e) {
        String polRezString;
        String pol1 = view.getFirstPolTxt();
        String pol2 = view.getSecondPolTxt();


        polynomial1 = polynomial1.StringToPolynomial(pol1);
        polynomial1.printPolynomial();


        polynomial2 = polynomial2.StringToPolynomial(pol2);
        polynomial2.printPolynomial();
        valid1 = validatePolynomial(pol1);
        valid2 = validatePolynomial(pol2);


        if (valid1 && valid2) {
            polRezultat = polynomial1.addPolynomial(polynomial2);
            polRezString = polRezultat.polinomToString();
            view.setAnswerTxt(polRezString);
        } else {
```

```
                if (!valid1 && !valid2)
                    JOptionPane.showMessageDialog(null, "Invalid input for both polynomials!", "Error.", JOptionPane.INFORMATION_MESSAGE);
                else {
                    if (!valid1)
                        JOptionPane.showMessageDialog(null, "Invalid input for the first polynomial!", "Error.", JOptionPane.INFORMATION_MESSAGE);
                    else {
                        JOptionPane.showMessageDialog(null, "Invalid input for the second polynomial!", "Error.", JOptionPane.INFORMATION_MESSAGE);
                    }
                }
            }
        }
    }
```

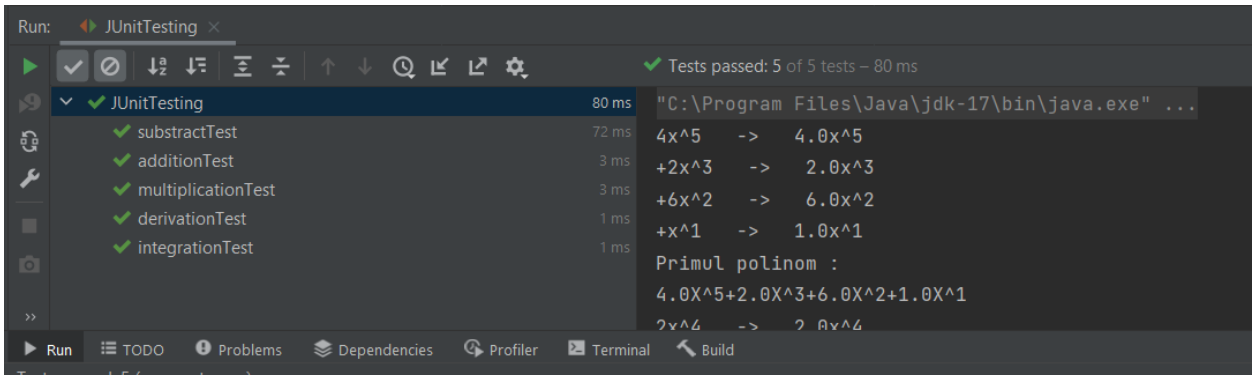
Clasa Main are rolul de a simula aplicația, declarând un view,un polinom si un controller.

```
public class Main {
    public static void main(String[] args) {
        View view = new View();
        Polynomial model = new Polynomial();
```

```
        Controller controller = new Controller(view, model);
    }
}
```

5. Rezultate

In clasa JUnitTesting am implementat cate un Test pentru fiecare operatie. Practic e aproximativ același cod in fiecare metoda, fiecare test, doar apelam metoda corespunzătoare testului. Ne-am declarat 3 string-uri,test1,test2,rezString.Test1 reprezintă primul polinom sub forma de string,respectiv test2 iar rezString e rezultatul. Convertim string-urile in Polynomial pentru a putea apela metoda „addPolynomial” spre exemplu. Rezultatul operației îl convertim in string înapoi si îl testam cu „rezultatul corect” care e de asemenea de tip string. Mici diferențe sunt la derivare si integrare in care avem doar un polinom.



Exemplu de test :

```
@Test
public void additionTest() {

    String test1,test2,rezString;
```

```
        test1 = "4x^5+2x^3+6x^2+x^1";
        test2 = "2x^4+3x^3+2x^2+3x^1";

        Polynomial poly1 = new Polynomial();
        poly1 = poly1.StringToPolynomial(test1);
        System.out.println("Primul polinom :");
        poly1.printPolynomial();

        Polynomial poly2 = new Polynomial();
        poly2 = poly2.StringToPolynomial(test2);
        System.out.println("Al doilea polinom :");
        poly2.printPolynomial();

        Polynomial rez;
        rez = poly1.addPolynomial(poly2);
        rezString = rez.polinomToString();
        System.out.println("Rezultat : ");
        System.out.println(rezString);

        assertEquals(rezString,"4.0X^5+2.0X^4+5.0X^3+8.0X^2+4.0X^1");
    }
```

Asemanator am implemetat si restul testelor pentru celelalte operatii.

6. Concluzii

In concluzie,in urma implementarii acestei teme se pot trage urmatoarele

concluzii :

Am invatat si modul de lucru cu regex pentru prelucrarea string-urilor.

Am reaprofundat matematica din spatele operatiilor cu polinoame.

Este mai usor si mai eficient sa lucram cu colectii decat cu tablouri unidimensionale ,creste viteza si eficienta programului.Codul este mult mai usor „citibil".

Am exersat implementarea unui „proiect" care are un de deadline si ne-a format capacitatea de a ne incadra intr-un anumit timp pentru implementarea temei.

Am reusit implementarea unui aplicatii de tip calculator, mvc, pentru polinoame.Pe viitor se pot implementa si alte operatii cum ar fi radicalul ridicarea la patrat, aplicarea functiei exponentiale sau logaritmice, aplicarea unor functii trigonometrice sau chiar aflarea radacinilor polinomului.

7. Bibliografie

https://users.utcluj.ro/~igiosan/teaching_poo.html - cursuri OOP Ionel Giosan

https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf - prezentarea proiectului

https://www.w3schools.com/java/java_arraylist.asp - informatii despre ArrayList<>

https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_1.pdf - matematica din spatele operatiilor

https://stackoverflow.com/questions/34946528/decode-polynomial-from-string-with-pattern-and-matcher - codul pentru regex

https://regex101.com/r/nI5oA5/1 -regex demo