

DOCUMENTATIE

TEMA *1*

NUME STUDENT: Boitor Diana-Elena
GRUPA: 30222

CUPRINS

1.	Obiectivul temei.....	3
2.	Analiza problemei, modelare, scenarii, cazuri de utilizare	3
3.	Proiectare	4
4.	Implementare	5
5.	Rezultate	12
6.	Concluzii.....	14
7.	Bibliografie	15

1. Obiectivul temei

Obiectivul principal este proiectarea și implementarea unui calculator de polinoame, cu o interfață grafică prin care utilizatorul poate adăuga polinoame, poate selecta operația dorită și poate vedea rezultatul operației.

Obiectivele secundare ale temei sunt:

- 1) Analiza problemei și identificarea cerințelor -> capitolul 2
- 2) Proiectarea calculatorului de polinoame -> capitolul 3
- 3) Implementarea operațiilor cu polinoame -> capitolul 4
- 4) Testarea calculatorului -> capitolul 5

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

Cerințele funcționale sunt:

- ➔ Calculatorul permite utilizatorului să insereze polinoame
- ➔ Calculatorul permite utilizatorului să selecteze operația matematică dorită(adunare, scădere, înmulțire, împărțire, derivare și integrare)
- ➔ Calculatorul trebuie să efectueze operația selectată
- ➔ Calculatorul trebuie să afișeze rezultatul obținut în urma efectuării operației alese

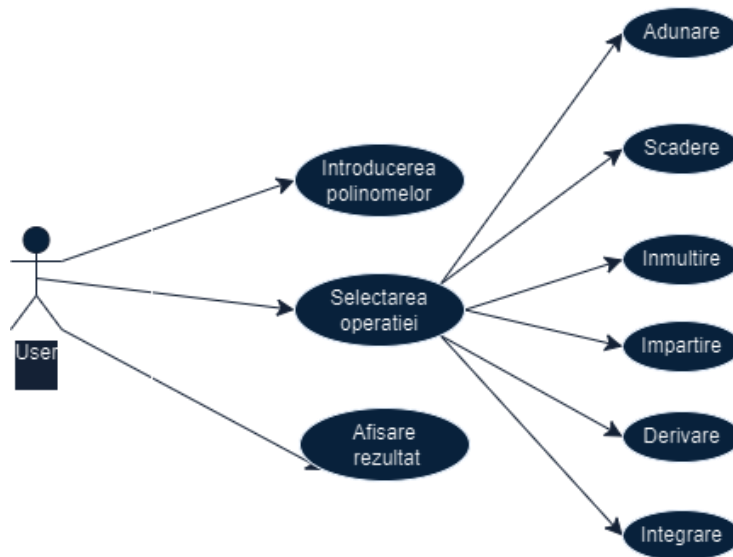
Cerințele nefuncționale sunt:

- ➔ Calculatorul este intuitiv și ușor de folosit de către utilizator
- ➔ Calculatorul are o interfață grafică bine structurată și plăcută vizual

Cazuri de utilizare:

- ➔ Utilizatorul introduce de la tastatură două polinome
- ➔ Utilizatorul selectează apoi una din operațiile ce pot fi efectuate:
 - Adunare
 - Scădere
 - Înmulțire
 - Împărțire
 - Derivare
 - Integrare
- ➔ Calculatorul efectuează operația și afișează rezultatul

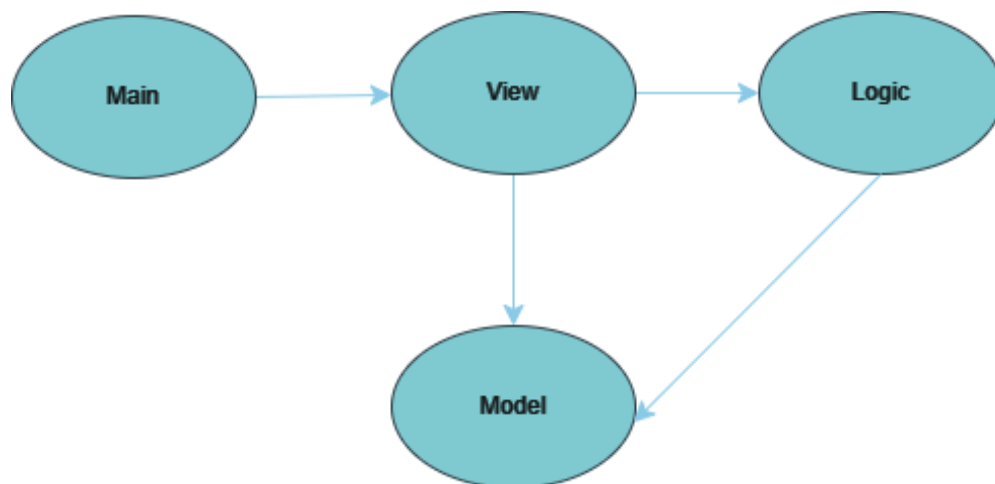
Diagrama de use-case



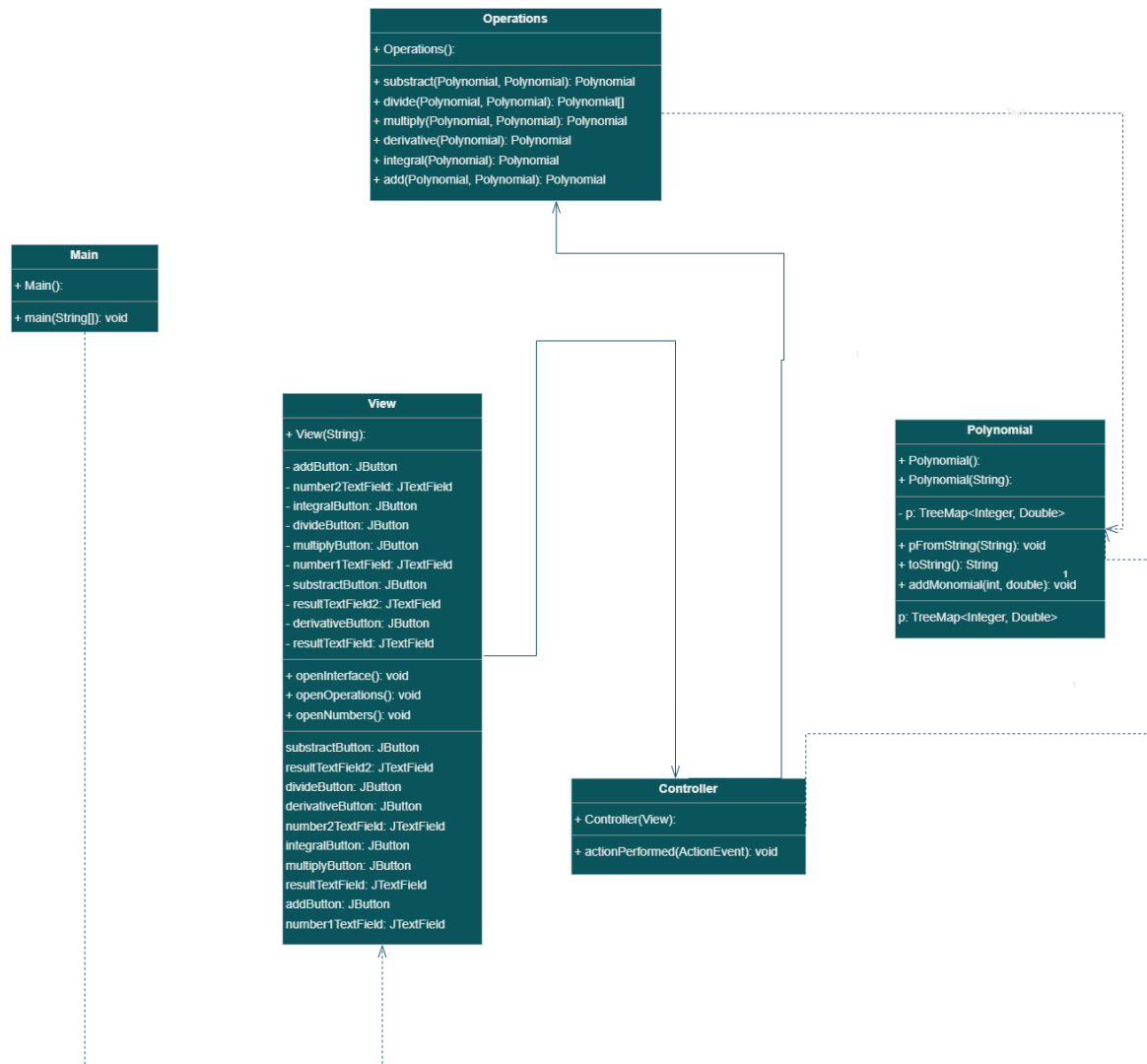
3. Proiectare

Am utilizat structura de date TreeMap, o colecție care stochează perechi cheie-valoare într-o ordine descrescătoare, pentru a avea gradele și coeficienții salvați într-o ordine asemănătoare cu cea a polinoamelor.

➔ Diagrama pachetelor:



➔ Diagrama claselor:



4. Implementare

Calculatorul de polinoame a fost implementat cu ajutorul a cinci clase:

- **Clasa Polynomial**

Această clasă este folosită pentru a reprezenta și manipula polinoame. În clasă este definită o variabilă `p`, ce este un obiect `TreeMap` care stochează gradele și coeficienții polinomului. Acesta este sortat într-o ordine descrescătoare, pentru a fi mai ușor de implementat ulterioarele funcții pentru polinoame.

Clasa are 2 constructori: unul implicit, fără argumente și unul ce primește un șir de caractere și inițializează polinomul folosind metoda `pFromString`.

```

public class Polynomial {

    3 usages
    private TreeMap<Integer,Double> p = new TreeMap<>(Comparator.reverseOrder());
    11 usages
    public Polynomial(){
    }
    24 usages
    public Polynomial(String text){ pFromString(text); }
}

```

➔ Metoda pFromString:

Această metodă primește un șir de caractere ce reprezintă polinomul introdus de către utilizator. Se utilizează apoi regex pentru a extrage gradele și coeficienții fiecărui monom, pe care îl adaugă apoi în polinom folosind metoda addMonomial.

➔ Metoda addMonomial:

Metoda adaugă un nou monom la polinom, gradul fiind cheia și coeficientul valoarea acestuia ce urmează să fie adaugate în TreeMap.

```

public void addMonomial(int degree, double coefficient){ p.put(degree, coefficient); }
1 usage
public void pFromString(String text){
    String pattern = "[+-]?\\d+(x)?\\^?(\\d+)?";
    Pattern regex=Pattern.compile(pattern);
    Matcher matcher=regex.matcher(text);
    while(matcher.find()){
        String coef=matcher.group(1);
        String coefX=matcher.group(2);
        String deg= matcher.group(3);
        double coef2;
        int deg2;
        if(coef==null || coef.isEmpty()){
            if(coefX==null || coefX.isEmpty()) coef2=0;
            else coef2=1;
        }
        else if(coef=="-")    coef2=-1;
        else if(coef=="+")    coef2=1;
        else    coef2=Integer.parseInt(coef);
        if(coefX==null || coefX.isEmpty())    deg2=0;
        else if(deg==null || deg.isEmpty())    deg2=1;
        else    deg2=Integer.parseInt(deg);
        if(coef2!=0 || deg2!=0)    addMonomial(deg2,coef2);
    }
}

```

➔ Metoda toString:

Această metodă convertește polinomul într-un șir de caractere pentru afișare. Se trece pe rând prin fiecare termen din Map și în funcție de semnul coeficientului, termenii sunt concatenați pentru a forma șirul final.

```

@Override
public String toString() {
    String r="";
    for (Map.Entry<Integer,Double>entry: p.entrySet()) {
        if(entry.getValue()>0){
            r=r+" "+String.valueOf(entry.getValue())+ "x^"+ String.valueOf(entry.getKey());
        }
        else if (entry.getValue()<0){
            r=r+" "+String.valueOf(entry.getValue())+ "x^"+ String.valueOf(entry.getKey());
        }
    }
    return r;
}

```

• Clasa Operations

Această clasă conține metode pentru efectuarea diferitelor operații matematice aplicate pe polinoame.

- ➔ Metoda add primește 2 obiecte de tipul Polynomial și returnează un nou obiect Polynomial ce reprezintă suma celor două polinoame. Această funcție parcurge fiecare termen din cele două polinoame (p1 și p2) și adaugă termenii corespunzători într-un nou polinom, calculând suma coeficienților pentru termenii cu aceeași putere.

```

public static Polynomial add(Polynomial p1, Polynomial p2){
    Polynomial newP=new Polynomial();
    for (Map.Entry<Integer,Double>entry1: p1.getP().entrySet()){
        newP.addMonomial(entry1.getKey(),entry1.getValue());
    }
    for (Map.Entry<Integer,Double>entry2: p2.getP().entrySet()){
        if(newP.getP().containsKey(entry2.getKey())){
            double sum = entry2.getValue() + newP.getP().get(entry2.getKey());
            newP.addMonomial(entry2.getKey(),sum);
        }
        else{
            newP.addMonomial(entry2.getKey(),entry2.getValue());
        }
    }
    return newP;
}

```

- ➔ Metoda subtract primește două obiecte de tip Polynomial și returnează un nou obiect Polynomial care reprezintă diferența dintre cele două polinoame. Funcția traversează fiecare termen din cele două polinoame (p1 și p2) și realizează scăderea coeficienților pentru termenii cu aceeași putere, adăugând rezultatele într-un nou polinom.

```

public static Polynomial subtract(Polynomial p1, Polynomial p2){
    Polynomial newP=new Polynomial();
    for (Map.Entry<Integer,Double>entry1: p1.getP().entrySet()){
        newP.addMonomial(entry1.getKey(),entry1.getValue());
    }
    for (Map.Entry<Integer,Double>entry2: p2.getP().entrySet()){
        if(newP.getP().containsKey(entry2.getKey())){
            double dif = newP.getP().get(entry2.getKey())-entry2.getValue();
            if(dif==0){
                newP.getP().remove(entry2.getKey());
            }
            else{
                newP.addMonomial(entry2.getKey(),dif);
            }
        }
        else{
            newP.addMonomial(entry2.getKey(),(-entry2.getValue()));
        }
    }
    return newP;
}

```

- Metoda multiply primește două obiecte de tip Polynomial și returnează un nou obiect Polynomial care reprezintă produsul celor două polinoame. Funcția realizează înmulțirea celor două polinoame (p1 și p2) prin combinarea fiecărui termen din p1 cu fiecare termen din p2, calculând puterile și coeficienții noii expresii rezultate și adăugându-le într-un nou polinom.

```

public static Polynomial multiply(Polynomial p1, Polynomial p2){
    Polynomial newP=new Polynomial();
    for (Map.Entry<Integer,Double>entry1: p1.getP().entrySet()){
        for (Map.Entry<Integer,Double>entry2: p2.getP().entrySet()){
            int degNou=entry1.getKey()+entry2.getKey();
            double coefNou= entry1.getValue()*entry2.getValue();
            if(newP.getP().containsKey(degNou)){
                coefNou += newP.getP().get(degNou);
            }
            newP.addMonomial(degNou,coefNou);
        }
    }
    return newP;
}

```

- Metoda divide primește două obiecte de tip Polynomial și returnează un array de două obiecte Polynomial. Primul element al array-ului este rezultatul împărțirii (câtul), iar al doilea element este restul împărțirii. Funcția realizează împărțirea polinomului cu grad mai mare la polinomul cu grad mai mic.


```

public static Polynomial[] divide(Polynomial p1, Polynomial p2){
    Polynomial[] rezultat = new Polynomial[2];
    rezultat[0]=new Polynomial();
    rezultat[1]=new Polynomial();
    Polynomial remainder = new Polynomial();
    Polynomial quotient = new Polynomial();
    int deg1= p1.getP().firstKey();
    int deg2= p2.getP().firstKey();
    if(deg1<deg2){
        Polynomial auxP=p1;
        p1=p2;
        p2=auxP; }
    int divisorDeg= p2.getP().firstEntry().getKey();
    double divisorCoef= p2.getP().firstEntry().getValue();
    while ((!p1.getP().isEmpty()) && (p1.getP().firstKey()>=divisorDeg)){
        remainder.getP().clear();
        int degNou= p1.getP().firstEntry().getKey()-divisorDeg;
        double coefNou= p1.getP().firstEntry().getValue()/divisorCoef;
        quotient.addMonomial(degNou,coefNou);
        for(Integer deg: p2.getP().keySet()){
            double coef = p2.getP().get(deg);
            remainder.addMonomial( degree: degNou + deg, coefficient: coefNou* coef);
        }
        p1= subtract(p1,remainder);
    }
    rezultat[0]=quotient;
    if(!p1.getP().isEmpty()) rezultat[1]=p1;
    else rezultat[1].addMonomial( degree: 0, coefficient: 0);
    return rezultat;
}

```

- Metoda derivative primește un obiect de tip Polynomial și returnează derivata acestuia, adică polinomul rezultat după aplicarea operației de derivare. Pentru aflarea derivatei polinomului, aplicăm regulile de derivare asupra fiecărui termen al polinomului primit ca parametru.

```

public static Polynomial derivative(Polynomial p1){
    Polynomial newP=new Polynomial();
    for (Map.Entry<Integer,Double>entry1: p1.getP().entrySet()){
        if(entry1.getKey()!=0){
            double coefNou= entry1.getValue()*entry1.getKey();
            int degNou=entry1.getKey()-1;
            newP.addMonomial(degNou,coefNou);
        }
    }
    return newP;
}

```

- Metoda integral primește un obiect de tip Polynomial și returnează integrala acestuia, adică polinomul rezultat după aplicarea operației de integrare. Pentru aflarea integralei polinomului, aplicăm regulile de integrare asupra fiecărui termen al polinomului primit ca parametru.

```

public static Polynomial integral(Polynomial p1){
    Polynomial newP=new Polynomial();
    for (Map.Entry<Integer,Double>entry1: p1.getP().entrySet()){
        int degNou=entry1.getKey()+1;
        double coefNou= entry1.getValue()/degNou;
        newP.addMonomial(degNou,coefNou);
    }
    return newP;
}

```

- ## Clasa Controller

Clasa Controller este responsabilă pentru gestionarea interacțiunii între interfața grafică (clasa View) și logica aplicației (clasa Operations). Ea implementează interfața ActionListener pentru a asculta evenimentele generate de utilizator în interfața grafică.

Constructorul clasei primește o referință către obiectul View și inițializează un obiect Operations pentru a putea efectua operațiile pe polinoame.

Metoda actionPerformed(ActionEvent e) este apelată atunci când un eveniment este declanșat în interfața grafică. În această metodă, se obțin valorile introduse de utilizator din câmpurile de text ale interfeței grafice și se creează obiecte de tip polinom corespunzătoare. Apoi, în funcție de sursa evenimentului (butoanele de adunare, scădere, înmulțire, împărțire, derivare sau integrare), se apelează metodele corespunzătoare din clasa Operations pentru a efectua operația pe polinoamele date. Rezultatele sunt apoi actualizate în câmpurile de text corespunzătoare ale interfeței grafice

```

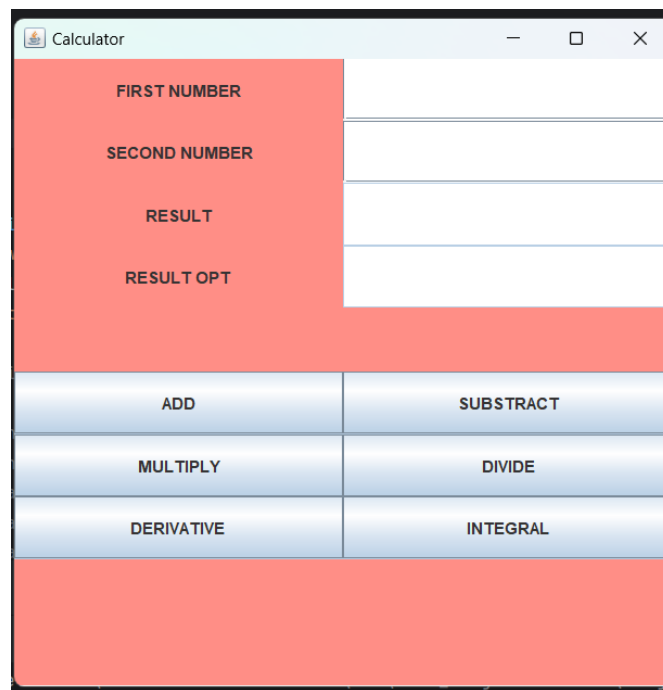
public void actionPerformed(ActionEvent e) {
    String text1= view.getNumber1TextField().getText();
    String text2=view.getNumber2TextField().getText();
    Polynomial p1=new Polynomial(text1);
    Polynomial p2=new Polynomial(text2);
    Polynomial pResult=new Polynomial();
    Polynomial pResult2=new Polynomial();
    view.getResultTextField().setEnabled(TRUE);
    view.getResultTextField2().setEnabled(TRUE);
    String result2="";
    if (e.getSource() == view.getAddButton()) {
        pResult=operations.add(p1, p2); }
    if (e.getSource() == view.getSubtractButton()) {
        pResult=operations.subtract(p1, p2); }
    if (e.getSource() == view.getMultiplyButton()) {
        pResult=operations.multiply(p1, p2); }
    if (e.getSource() == view.getDivideButton()) {
        pResult=operations.divide(p1, p2)[0];
        pResult2=operations.divide(p1, p2)[1];
        result2=pResult2.toString(); }
    if (e.getSource() == view.getDerivativeButton()) {
        pResult=operations.derivative(p1);
        pResult2=operations.derivative(p2);
        result2=pResult2.toString(); }
    if (e.getSource() == view.getIntegralButton()) {
        pResult=operations.integral(p1);
        pResult2=operations.integral(p2);
        result2=pResult2.toString(); }
    String result=pResult.toString();
    view.setResultTextField(result);
    view.setResultTextField2(result2);
}

```

- **Clasa View**

Clasa View este responsabilă pentru gestionarea interfeței grafice a calculatorului de polinoame. Aceasta conține o fereastră principală extinsă din clasa JFrame, în care sunt plasate diferite componente grafice cum ar fi panouri pentru introducerea numerelor și operațiile disponibile, etichete pentru afișarea rezultatelor și butoane pentru efectuarea operațiilor.

Clasa inițializează interfața grafică prin stabilirea dimensiunilor fereastrei și a layout-ului panoului principal. Apoi, creează panourile pentru introducerea numerelor și operațiile disponibile, fiecare având propriul său layout și stil. Panoul pentru introducerea numerelor conține etichete și câmpuri de text pentru introducerea numerelor, precum și câmpuri pentru afișarea rezultatelor. Panoul pentru operații conține butoane pentru adunare, scădere, înmulțire, împărțire, derivare și integrare a polinoamelor. Clasa View furnizează, de asemenea, metode pentru a obține și seta valorile introduse în câmpurile de text și pentru a actualiza afișarea rezultatelor.



- **Clasa Main**

Aceasta clasă conține o metodă statică “main” care reprezintă punctul de intrare al aplicației. În interiorul metodei "main" se creează și deschide fereastra principală pentru aplicația Java.

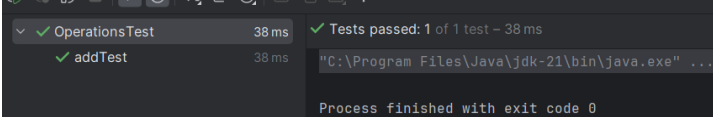
```
public class Main {
    public static void main(String[] args) {
        JFrame frame = new View( name: "Calculator");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

5. Rezultate

Am implementat funcții de testare pentru realizarea testării cu JUnit.

- Implementarea funcției de testare pentru operația de adunare

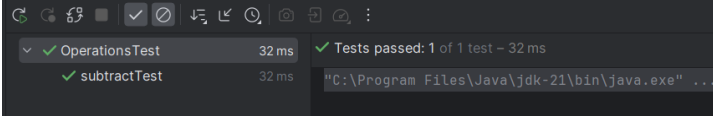
```
@Test
public void addTest(){
    Polynomial p1 = new Polynomial( text: "3x^4+2x^2+3");
    Polynomial p2 = new Polynomial( text: "2x^5+3x^2+6x+7");
    Polynomial p3 = new Polynomial( text: "4x^5-3x^4+1x^2-8x+1");
    Polynomial p4 = new Polynomial( text: "3x^4-1x^3+1x^2+2x-1");
    Polynomial sum = Operations.add(p1,p2);
    Polynomial sum2 = Operations.add(p3,p4);
    String result = sum.toString();
    String result2 = sum2.toString();
    assertEquals (result, actual: "+2.0x^5+3.0x^4+5.0x^2+6.0x^1+10.0x^0");
    assertEquals (result2, actual: "+4.0x^5-1.0x^3+2.0x^2-6.0x^1");
}
```



The screenshot shows the JUnit test results for the `addTest` method. The test passed successfully, taking 38 ms. The output shows the actual results of the polynomial addition tests.

- Implementarea funcției de testare pentru operația de scădere

```
@Test
public void subtractTest(){
    Polynomial p1 = new Polynomial( text: "3x^4+2x^2+3");
    Polynomial p2 = new Polynomial( text: "2x^5+3x^2+6x+7");
    Polynomial p3 = new Polynomial( text: "4x^5-3x^4+1x^2-8x+1");
    Polynomial p4 = new Polynomial( text: "3x^4-1x^3+1x^2+2x-1");
    Polynomial dif = Operations.subtract(p1,p2);
    Polynomial dif2 = Operations.subtract(p3,p4);
    String result = dif.toString();
    String result2 = dif2.toString();
    assertEquals (result, actual: "-2.0x^5+3.0x^4-1.0x^2-6.0x^1-4.0x^0");
    assertEquals (result2, actual: "+4.0x^5-6.0x^4+1.0x^3-10.0x^1+2.0x^0");
}
```



The screenshot shows the JUnit test results for the `subtractTest` method. The test passed successfully, taking 32 ms. The output shows the actual results of the polynomial subtraction tests.

- Implementarea funcției de testare pentru operația de înmulțire

```
@Test
public void multiplyTest(){
    Polynomial p1 = new Polynomial( text: "3x^2-1x+1");
    Polynomial p2 = new Polynomial( text: "x-2");
    Polynomial p3 = new Polynomial( text: "3x^2+2x+1");
    Polynomial p4 = new Polynomial( text: "4x^3-1x^2+2");
    Polynomial produs = Operations.multiply(p1,p2);
    Polynomial produs2 = Operations.multiply(p3,p4);
    String result = produs.toString();
    String result2 = produs2.toString();
    assertEquals (result, actual: "+3.0x^3-7.0x^2+3.0x^1-2.0x^0");
    assertEquals (result2, actual: "+12.0x^5+5.0x^4+2.0x^3+5.0x^2+4.0x^1+2.0x^0");
}

✓ OperationsTest 58 ms ✓ Tests passed: 1 of 1 test - 58 ms
✓ multiplyTest 58 ms "C:\Program Files\Java\jdk-21\bin\java.exe" ...
Process finished with exit code 0
```

- Implementarea funcției de testare pentru operația de împărțire

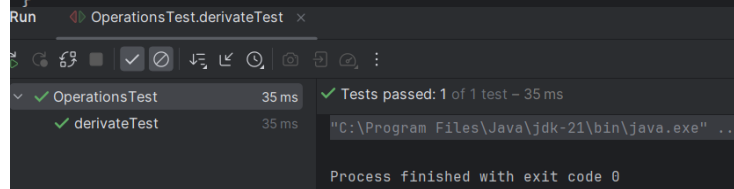
```
@Test
public void divideTest(){
    Polynomial p1 = new Polynomial( text: "x^3-2x^2+6x-5");
    Polynomial p2 = new Polynomial( text: "x^2-1");
    Polynomial p3 = new Polynomial( text: "4x-4");
    Polynomial p4 = new Polynomial( text: "2x-2");
    Polynomial[] div = Operations.divide(p1,p2);
    Polynomial[] div2 = Operations.divide(p3,p4);
    String result = div[0].toString();
    String result2 = div[1].toString();
    String result3 = div2[0].toString();
    String result4 = div2[1].toString();
    assertEquals (result, actual: "+1.0x^1-2.0x^0");
    assertEquals (result2, actual: "+7.0x^1-7.0x^0");
    assertEquals (result3, actual: "+2.0x^0");
    assertEquals (result4, actual: "");
}

✓ OperationsTest 46 ms ✓ Tests passed: 1 of 1 test - 46 ms
✓ divideTest 46 ms "C:\Program Files\Java\jdk-21\bin\java.exe" ...
Process finished with exit code 0
```

- Implementarea funcției de testare pentru operația de derivare

```
@Test
public void derivateTest(){
    Polynomial p1 = new Polynomial( text: "x^3-2x^2+6x-5");
    Polynomial p2 = new Polynomial( text: "3x^4-2x^3+5x^2-1x+7");
    Polynomial deriv = Operations.derivative(p1);
    Polynomial deriv2 = Operations.derivative(p2);
    String result = deriv.toString();
    String result2 = deriv2.toString();
    assertEquals (result, actual: "+3.0x^2-4.0x^1+6.0x^0");
    assertEquals (result2, actual: "+12.0x^3-6.0x^2+10.0x^1-1.0x^0");
}

Run OperationsTest.derivateTest x
```



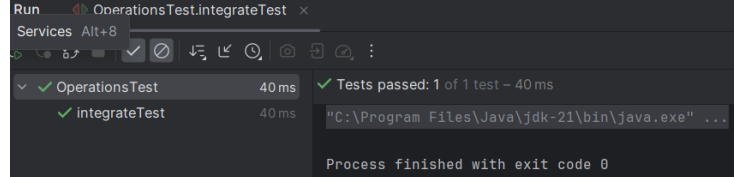
Tests passed: 1 of 1 test – 35 ms

Process finished with exit code 0

- Implementarea funcției de testare pentru operația de integrare

```
@Test
public void integrateTest(){
    Polynomial p1 = new Polynomial( text: "x^3+6x^2+5");
    Polynomial p2 = new Polynomial( text: "2x^3+3x^2-5x+2");
    Polynomial integral = Operations.integral(p1);
    Polynomial integral2 = Operations.integral(p2);
    String result = integral.toString();
    String result2 = integral2.toString();
    assertEquals (result, actual: "+0.25x^4+2.0x^3+5.0x^1");
    assertEquals (result2, actual: "+0.5x^4+1.0x^3-2.5x^2+2.0x^1");
}

Run OperationsTest.integrateTest x
```



Tests passed: 1 of 1 test – 40 ms

Process finished with exit code 0

6. Concluzii

În concluzie, acest proiect m-a ajutat să mi consolidez cunoștințele de Programare Orientată Pe Obiecte , dar și să dobândesc altele noi prin utilizarea unor concept, cum ar fi: folosirea regex-ului, testarea unitară cu JUnit.

Ca și viitoare îmbunătățiri, s-ar putea adăuga mai multe operații cu polinoame (găsirea rădăcinilor unui polinom, ridicarea polinomului la o putere și altele). Interfața ar putea fi și ea îmbunătățită (un exemplu ar fi adăugarea unei tastaturi în fereastra programului prin care să fie introduse astfel polinoamele).

7. Bibliografie

1. https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf
2. https://dsrl.eu/courses/pt/materials/PT_2024_A1_S2.pdf
3. https://dsrl.eu/courses/pt/materials/PT_2024_A1_S3.pdf
4. <https://regexr.com/>
5. <https://www.simplilearn.com/tutorials/java-tutorial/what-is-junit>
6. <https://www.javatpoint.com/java-swing>