DOCUMENTATIE

TEMA *NUMARUL_1*

NUME STUDENT: Jarda Adina-Ionela
GRUPA: 30222

CUPRINS

1. Obiectivul temei

Obiectivul principal al acestei prime teme este crearea unui calculator de polinoame cu o interfata grafica dedicata utilizatorului. Utilizatorul poate introduce de la tastatura polinoamele dorite, iar mai apoi poate selecta operatia de efectuat; adunarea, scaderea, inmultirea, impartirea, derivarea, dar si integrarea polinoamelor; si poate vizualiza rezultatul, care este tot afisat pe ecran, in campul “Rezultat”.

Obiectivele secundare ale temei sunt:

- Implementarea unei structuri de date pentru reprezentarea polinoamelor si a operatiilor cu acestea

-Utilizarea expresiilor regulate si potrivirii de modele pentru extragerea coeficientilor polinomului

-Implementarea testelor unitare folosind Junit

Toate aceste lucruri sunt detaliate in capitolul **Implementare** unde se va prezenta fiecare clasa in parte si metodele importante folosite.

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

In prima parte vom discuta despre cerintele functionale si non-functionale implementate in cadrul acestui proiect. Principalele cerinte functionale ale temei sunt reprezentate de:

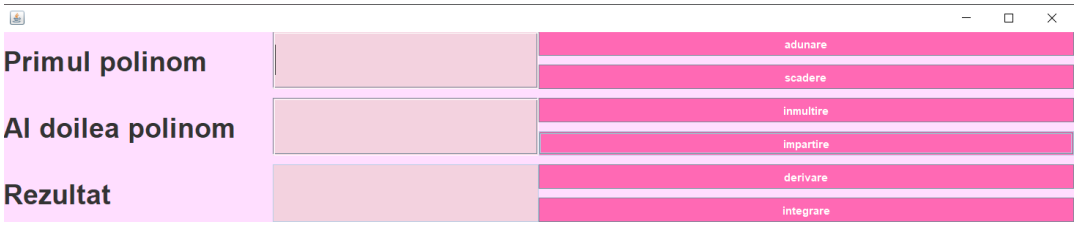
-Crearea unui calculator de polinoame cu interfata grafica(GUI);
-Posibilitatea introducerii polinoamelor de catre utilizator;
-Selectarea unei operatii matematice dintre cele implementate(adunare, scadere, inmultire, impartire, derivare si integrare);
-Vizualizarea rezultatului operatiei selectate anterior;

Iar principalele cerinte non-functionale ale proiectului sunt:

-Utilizarea unei structuri de date eficiente pentru reprezentare polinoamelor (TreeMap, HashMap);
-Interfata garfica sa fie intuitiva si usor de utilizat;
-Eficienta in efectuarea operatiilor matematice

Manual de utilizare

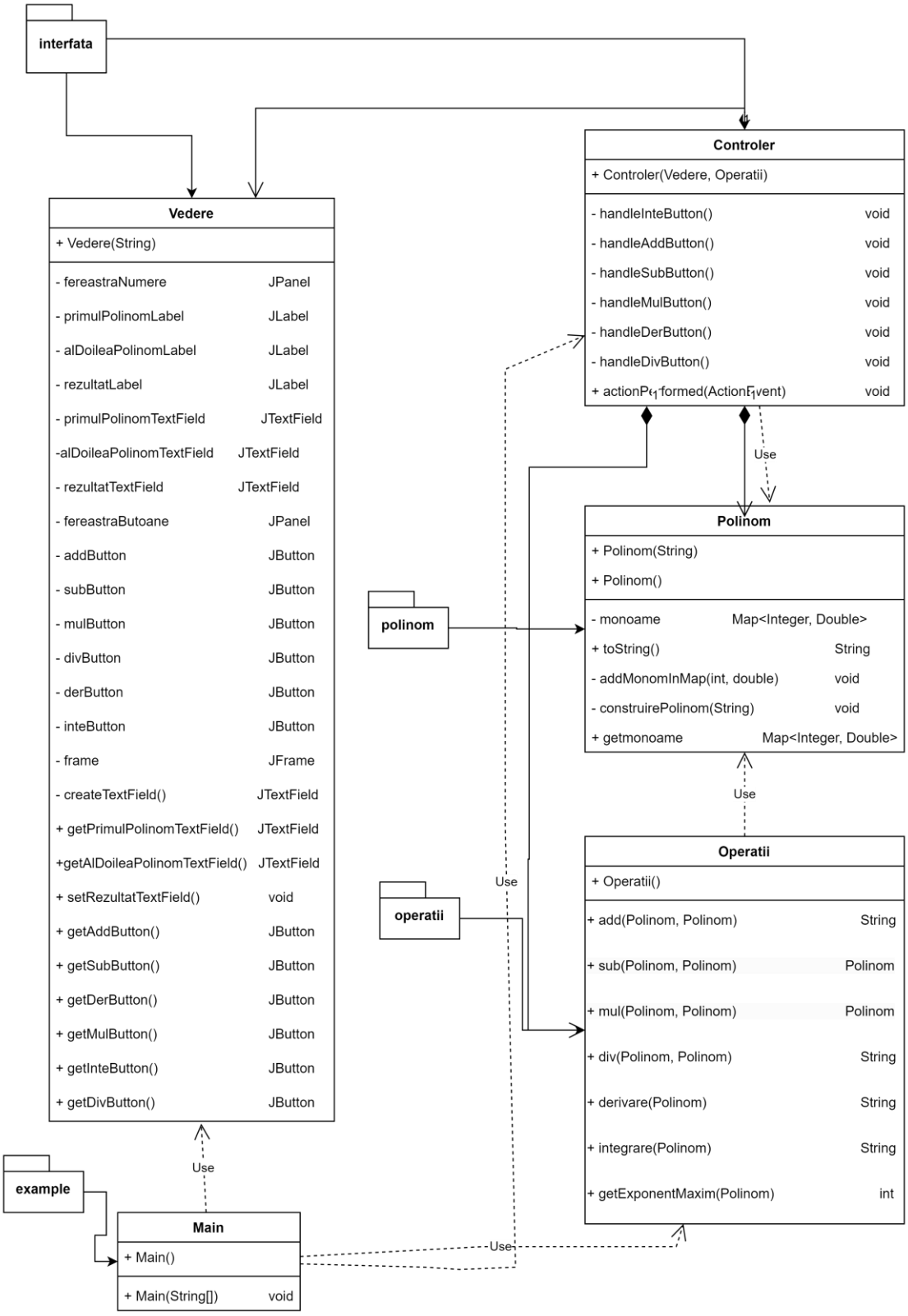
- Utilizatorul ruleaza programul ce va deschide calculatorul de polinoame.
- Odata deschisa, fereastra calculatorului arata astfel:



- Utilizatorul poate sa incerce sa introduca polinoame.
- Coeficientii pot fi introdusi sub 2 forme; prima de tipul c*X si a doua de tipul cX (ex: 2*x sau 2x), iar exponentul poate fi scris ca x^e sau xe (ex: x^2 sau x2).
- Utilizatorul alege ulterior operatia pe care doreste sa o aplice asupra polinoamelor introduse apasand unul dintre butoanele denumite sugestiv operatiilor pe care acestea le implementeaza.
- OPERATIA DE INTEGRARE SI DERIVARE SE FACE DOAR PE POLINOMUL 1.

- Odata ce butonul operatiei de efectuat este apasat rezultatul va fi afisat in chenarul roz din dreptul textului “Rezultat”. Fiecare buton apasat implementeaza in spate (in calsa de operatii) operatia dorita.
- Apasand butonul “adunare” se va efectua adunarea primului polinom cu al doilea, iar suma este tiparita in “Rezultat”, acelasi lucru este implementat si pentru scadere si inmultire.
- Cand vine vorba de impartire, daca primul polinom are gradul maxim mai mic decat al doilea polinom in “Rezultat” va aparea textul “p1 mai mic decat p2”, iar impartirea nu va avea loc. Daca polinomul al doilea este 0, la fel, impartirea nu va avea loc.
- Derivarea si Integrarea au loc doar asupra primul polinom iar daca acesta este valid, in “Rezultat” va aparea polinomul derivat sau integrat dupa caz.

3. Proiectare



4.Implementare

1.Polinom

Campuri: ***-monoame;*** Un **Map<Integer, Double>** care stocheaza monoamele polinomului, unde cheia este exponentul si valoarea este coeficientul.

Constructori: ***-Public Polinom();*** Constructor fara parametrii care initializeaza ***monoame*** ca un **HashMap**

- Public Polinom(String PolinomString); Constructor ce primeste un sir de caractere reprezentand un polinom si construieste polinomul utilizand metoda ***construirePolinom();***

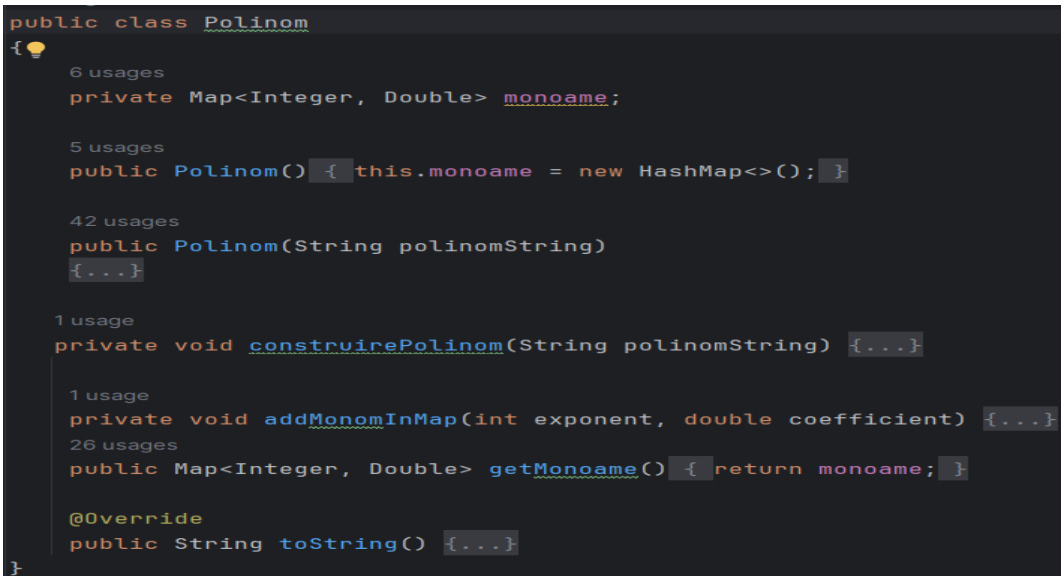
Metode: ***-private void construirePolinom(String polinomString);*** O metoda private care analizeaza sirul de caractere dat drept input si construieste monoame pe baza acestuia.

-private void addMonomInMap(int exponent, double coefficient); O metoda private care adauga un monom (coeficientul si exponentul) in ***monoame***, gestionand cazurile speciale.

-public Map<Integer, Double>getMonoame(); O metoda publica care returneaza map-ul ***monoame***.

-@Override public String toString(); O metoda suprascrisa care returneaza o reprezentare sub forma de sir de caractere a polinomului, folosind coeficientii si exponentii, si aplicand unele reguli de formatere pentru a-l face mai usor de citit si de inteles.

Aceasta este o prezentare generala a clasei Polinom si a metodelor sale. Aceasta clasa reprezinta o parte esentiala a implementarii calculatorului de polinoame si este responsabila pentru manipularea si reprezentarea polinoamelor in cadrul aplicatiei.



```
public class Polinom
{
    6 usages
    private Map<Integer, Double> monoame;

    5 usages
    public Polinom() { this.monoame = new HashMap<>(); }

    42 usages
    public Polinom(String polinomString)
    {...}

    1 usage
    private void construirePolinom(String polinomString) {...}

    1 usage
    private void addMonomInMap(int exponent, double coefficient) {...}
    26 usages
    public Map<Integer, Double> getMonoame() { return monoame; }

    @Override
    public String toString() {...}
}
```

2.Operatii

Metode: ***-public String add(Polinom p1, Polinom p2);*** Aceasta metoda aduna doua polinoame p1 si p2 . Parcurge fiecare monom din p2, verifica daca exista deja in p1, si adauga coeficientul corespunzator. Returnează un sir de caractere reprezentand polinomul rezultat.

-public Polinom sub(Polinom p1, Polinom p2); Aceasta metoda scade polinomul p2 din polinomul p1. Similar cu metoda **add()**, parcurge fiecare monom din p2 si actualizeaza coeficientii din p1. Returneaza polinomul rezultat.

-public Polinom mul(Polinom p1, Polinom p2); Aceasta metoda inmulteste doua polinoame p1 si p2. Parcurge fiecare monom din ambele polinoame, efectueaza inmultirea si adauga monoamele intr-un nou polinom rezultat. Returneaza polinomul rezultat.

-public int getExponentMaxim(Polinom p); Aceata metoda determina exponentul maxim dintr-un polinom p. Parcurge toate monoamele din polinom si returneaza cel mai mare exponent.

-public String div(Polinom p1, Polinom p2); Aceasta metoda imparte polinomul p1 la polinomul p2. Implementeaza algoritmul de impartire a polinoamelor. Returneaza un sir de caractere reprezentand rezultatul impartirii si restul

-public String derivare(Polinom p1, Polinom p2); Aceasta metoda calculeaza prima derivata a polinomului p1. Parcurge fiecare monom si aplica regula derivarii. Returneaza un sir de caractere reprezentand polinomul rezultat.

-public String integrare(Polinom p1, Polinom p2); Aceasta metoda calculeaza integrala polinomului p1. Parcurge fiecare monom si aplica regula integrarii. Returneaza un sir de caractere reprezentand polinomul rezultat.

Această clasa contine operatiile matematice principale pentru manipularea polinoamelor si este esentiala pentru functionarea corecta a calculatorului de polinoame.

```
public class Operatii
{
    public String add(Polinom p1, Polinom p2) {...}

    4 usages
    public Polinom sub(Polinom p1, Polinom p2) {...}

    4 usages
    public Polinom mul(Polinom p1, Polinom p2) {...}

    7 usages
    public int getExponentMaxim(Polinom p) {...}

    3 usages
    public String div(Polinom p1, Polinom p2) {...}

    3 usages
    public String derivare(Polinom p1) {...}

    3 usages
    public String integrare(Polinom p1) {...}
}
```

3.Controler

Campuri: ***-p1;*** Un obiect de tip **Polinom** care reprezinta primul polinom cu care se lucreaza.

-p2; Un obiect de tip **Polinom** care reprezinta al doilea polinom cu care se lucreaza.

-view; O referinta catre obiectul de tip **Vedere**, care reprezinta interfata grafica.

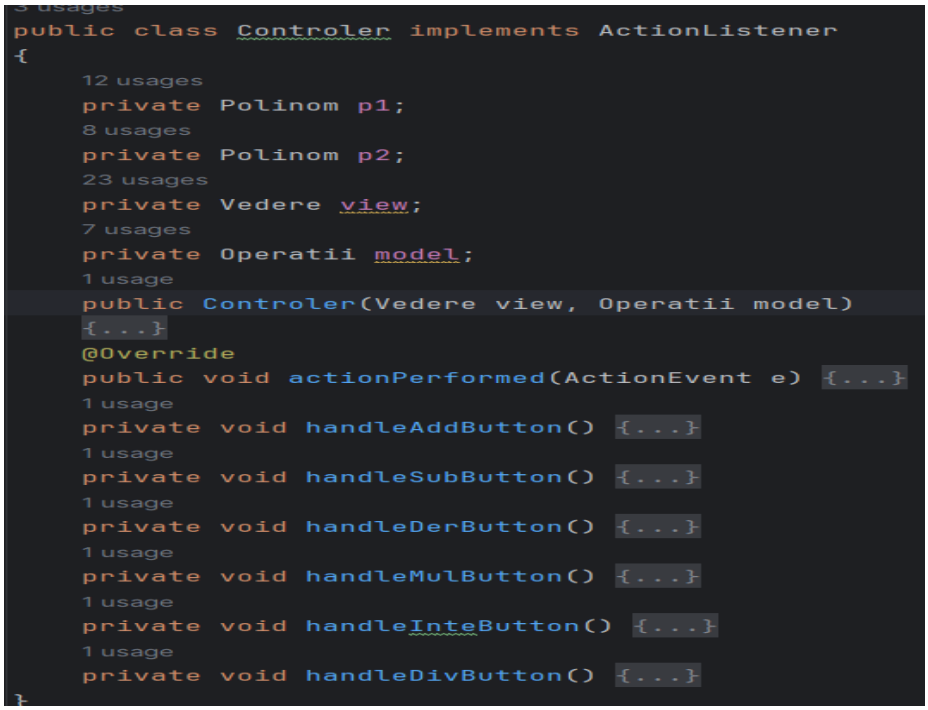
-model; O referinta catre obiectul de tip Operatii, care contine operatiile matematice

Constructor: ***-public Controler(Vedre view, Operatii model);*** Constructorul clasei **Controler,** care primeste un obiect de tip **Vedere** si unul de tip **Operatii**, si le asociaza campurilor corespunzatoare. De asemenea, adauga ascultatori pentru butoanele din interfata grafica.

Metode: ***-public void actionPerformed(ActionEvent e);*** O metoda suprascrisa din interfata `ActionListener` care gestioneaza evenimentele declansate de butoanele din interfata grafica si apeleaza metodele corespunzatoare in functie de butonul apasat.

-metode private ***handleAddButton(); handleSubButton(); handleMulButton(); handleDivButton(); handleDerButton(); handleInteButton();*** Aceste metode sunt responsabile pentru preluarea polinoamelor introduse de utilizator din interfata grafica, apelarea metodelor corespunzatoare din clasa `Operatii` pentru a efectua operatiile matematice si actualizarea interfetei grafice cu rezultatul obtinut.

Aceasta clasa serveste ca intermediar intre interfața grafica si operatiile matematice, gestionand interactiunea dintre acestea doua si asigurand functionarea corecta a aplicatiei.



4.Vedere

Campuri: ***-fereastraNumere;*** Un panou care contine etichetele si campurile text pentru intorducerea polinoamelor si afisarea rezultatului.

-primulPolinomLabel; alDoileaPolinomLabel; rezultatLabel; Etichete care indica utilizatorului scopul fiecarui camp text.

-primulPolinomTextField; alDoileaPolinomTextField; rezultatTextField; Campuri text in care utilizatorul poate introduce polinoame si in care se afiseaza rezultatul.

-fereastraButoane; Un panou care contine butoanele pentru operatiile matematice.

-addButon; subButton; mulButton; divButton; derButton; inteButton; Butoane care permit utilizatorului sa selecteze operatia dorita.

-frame; Obiectul de tip **JFrame** care reprezinta fereastra principala a aplicatiei.

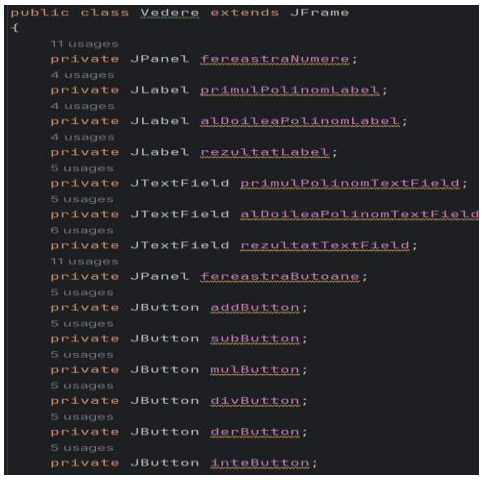
Costructor: ***-public Vedere(String name);*** Constructorul clasei **Vedere**, care initializeaza componentele interfetei grafice, cum ar fi etichetele, campurile text si butoanele si le adauga in fereastra principala.

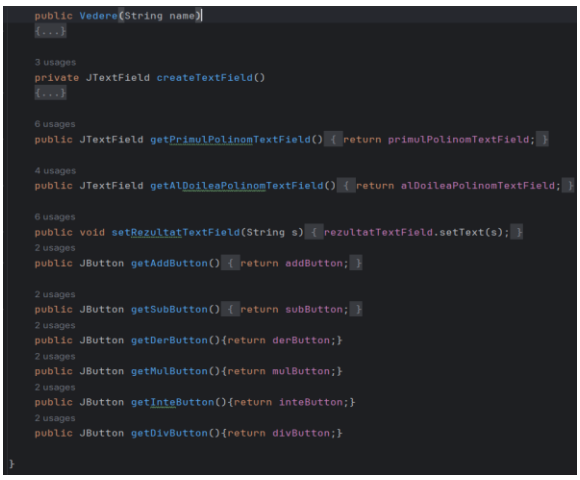
Metode: ***-private JTextField createTextField();*** O metoda private care creeaza un camp text cu o anumita dimensiune si ii setaza fundalul.

-private void setRezultatTextField(String s); O metoda care setaza textul afisat in campul pentru rezultat.

-Metode de accesoriu ***getPrimulPolinomTextField(); getAlDoileaPolinomTextFied(); getAddButton(); getSubButton(); getDerButton(); getMulButton(); getDivButton(); getInteButton();*** Aceste metode permit accesul la campurile si butoanele interfetei grafice din alte clase, cum ar fi **Controler.**

Aceasta clasa reprezinta interfata grafica a aplicatiei, oferind utilizatorului un mod intuitiv de a interactiona cu calculatorul polinomial.





5. Main

Metode: ***-main();***

-public static void main(String[] args); Aceasta metoda reprezinta punctul de intrare in aplicatie. In cadrul acestei metode se initializeaza toate componentele necesare pentru functionarea aplicatiei, cum ar fi interfata grafica, operatiile matematice si controlerul.

- se creeaza un obiect de tip **Vedere** pentru a afisa interfata grafica a aplicatiei.

-se creeaza un obiect de tip **Operatii** pentru a efectua operatiile matematice asupra polinoamelor.

-se creeaza un obiect de tip **Controler**, care leaga interfata grafica de operatiile matematice, astfel incat sa se poata reactiona la actiunile utilizatorului.

Aceasta clasa reprezinta punctul de intrare in aplicatie si initializeaza toate componentele necesare pentru functionarea corecta a calculatorului polinomial.

```
public class Main
{
    public static void main( String[] args )
    {
        Vedere view=new Vedere( name: "proiect");
        Operatii model=new Operatii();
        Controler controller = new Controler(view, model);

    }
}
```

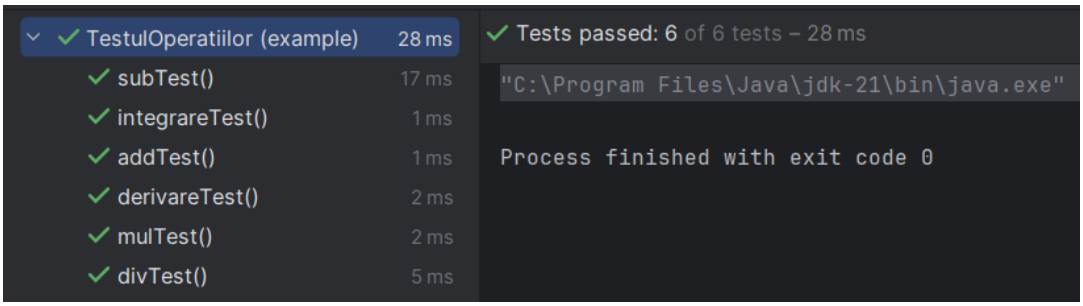
5. Rezultate

Pentru a prezenta scenariile de testare, putem enumera câteva situații de testare pentru fiecare operație a calculatorului polinomial. Acestea pot include: adunare, scadere, inmultire, impartire, derivare, integrare.

```java
@Test
public void addTest()
{
    Polinom p1=new Polinom( polinomString: "x^3+x^2");
    Polinom p2=new Polinom( polinomString: "-x");

    Operatii operatie=new Operatii();
    String suma=operatie.add(p1, p2);

    Polinom test=new Polinom( polinomString: "x^3+x^2-x");

    assertEquals(test.toString(), suma);

    Polinom p3=new Polinom( polinomString: "4*x^5-3*x^4+x^2-8*x+1");
    Polinom p4=new Polinom( polinomString: "3*x^4-x^3+x^2+2*x-1");

    String suma2=operatie.add(p3, p4);

    Polinom test2=new Polinom( polinomString: "4x^5-x^3+2*x^2-6*x");

    assertEquals(test2.toString(), suma2);
}
@Test
public void addTest()
{
    Polinom p1=new Polinom( polinomString: "x^3+x^2");
    Polinom p2=new Polinom( polinomString: "-x");

    Operatii operatie=new Operatii();
    String suma=operatie.add(p1, p2);

    Polinom test=new Polinom( polinomString: "x^3+x^2-x");

    assertEquals(test.toString(), suma);

    Polinom p3=new Polinom( polinomString: "4*x^5-3*x^4+x^2-8*x+1");
    Polinom p4=new Polinom( polinomString: "3*x^4-x^3+x^2+2*x-1");

    String suma2=operatie.add(p3, p4);

    Polinom test2=new Polinom( polinomString: "4x^5-x^3+2*x^2-6*x");

    assertEquals(test2.toString(), suma2);
}
@Test
public void mulTest()
{
    Polinom p1=new Polinom( polinomString: "x^3+x^2");
    Polinom p2=new Polinom( polinomString: "-x");

    Operatii operatie=new Operatii();
    Polinom produs=operatie.mul(p1, p2);

    Polinom test=new Polinom( polinomString: "-x^4-x^3");

    assertEquals(test.toString(), produs.toString());

    Polinom p3=new Polinom( polinomString: "3*x^2-x+1");
    Polinom p4=new Polinom( polinomString: "x-2");

    Polinom produs2=operatie.mul(p3, p4);

    Polinom test2=new Polinom( polinomString: "3*x^3-7*x^2+3*x-2");

    assertEquals(test2.toString(), produs2.toString());
}
```

```java
@Test
public void divTest()
{
    Polinom p1=new Polinom( polinomString: "x^3+x^2");
    Polinom p2=new Polinom( polinomString: "-x");

    Operatii operatie=new Operatii();
    String cat=operatie.div(p1, p2);

    String test="-x^2-x ,Rest: ";

    assertEquals(test, cat);

    Polinom p3=new Polinom( polinomString: "x^3-2*x^2+6*x-5");
    Polinom p4=new Polinom( polinomString: "x^2-1");

    String cat2=operatie.div(p3, p4);

    String test2="x-2.0 ,Rest: 7.0*x-7.0";

    assertEquals(test2, cat2);
}
 @Test
 public void integrareTest()
 {
     Polinom p1=new Polinom( polinomString: "4*x^3-3*x^2+6*x-5");

     Operatii operatie=new Operatii();
     String integrare=operatie.integrare(p1);

     Polinom test=new Polinom( polinomString: "x^4-x^3+3*x^2-5*x");

     assertEquals(test.toString(), integrare);

     Polinom p2=new Polinom( polinomString: "3x^2+2*x+1");

     String integrare2=operatie.integrare(p2);

     Polinom test2=new Polinom( polinomString: "x^3+x^2+x");

     assertEquals(test2.toString(), integrare2);
 }
@Test
public void subTest()
{
    Polinom p1=new Polinom( polinomString: "x^3+x^2");
    Polinom p2=new Polinom( polinomString: "-x");

    Operatii operatie=new Operatii();
    Polinom diferenta=operatie.sub(p1, p2);

    Polinom test=new Polinom( polinomString: "x^3+x^2+x");

    assertEquals(test.toString(), diferenta.toString());

    Polinom p3=new Polinom( polinomString: "4*x^5-3*x^4+x^2-8*x+1");
    Polinom p4=new Polinom( polinomString: "3*x^4-x^3+x^2+2*x-1");

    Polinom diferenta2=operatie.sub(p3, p4);

    Polinom test2=new Polinom( polinomString: "4x^5-6*x^4+x^3-10*x+2");

    assertEquals(test2.toString(), diferenta2.toString());
}
```

Toate operatiile de test au fost trecute cu success.



6. Concluzii

Prin explorarea acestei teme, am avut ocazia sa ma familiarizez cu regex-urile pentru prima dată, cat si sa inteleg cum functioneaza in detaliu si cum pot fi aplicate in diverse contexte. De asemenea, am descoperit puterea si utilitatea HashMap-urilor pentru stocarea eficienta a datelor, permitandu-mi sa gestionez informatiile intr-un mod optim.

Pentru a continua dezvoltarea proiectului, m-am gandit la doua directii interesante. In primul rand, consider ca o etapa importanta ar fi sa construiesc o interfata mai placuta din punct de vedere vizual. O astfel de interfata ar putea imbunatati experienta utilizatorului si ar putea face aplicatia mai atragatoare si mai usor de utilizat.

In al doilea rand, mi-am dat seama ca ar fi util sa adaug o verificare suplimentara pentru a asigura ca polinomul introdus de utilizator este intr-o formă corecta si valida. Acest lucru ar putea implica implementarea unei conditii de validare care sa verifice daca forma polinomului este conforma cu cerintele specifice ale sistemului. Astfel, utilizatorii ar fi ghidati sa introduca date corecte si ar reduce posibilitatea de erori sau confuzii in utilizare.

Aceste doua imbunatatiri ar putea consolida si mai mult functionalitatea si utilitatea proiectului, oferind o experienta imbunatatita si mai placuta pentru utilizatori.

7. Bibliografie

https://dsrl.eu/courses/pt/materials/PT2024_A1.pdf
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S2.pdf
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S3.pdf