DOCUMENTATIE

TEMA 1

NUME STUDENT: Ivan Alexandru - Mihai
GRUPA: 30222

CUPRINS

1. Obiectivul temei

Obiectivul principal

Realizarea unui calculator care opereaza pe polinoame si ofera functionalitati pentru adunare, scadere, inmultire, impartire, derivare si integrare a polinoamelor, utilizatorul interactionand cu calculatorul printr-o interfata grafica dedicata prin care acesta poate adauga doua polinoame, apoi selecteaza operatia dorita.

Obiectivele secundare

- Analizarea problemei si identificarea necesitatilor.
- Crearea unui design pentru calculator
- Implementarea calculatorului care lucreaza pe polinoame
- Testarea calculatorului

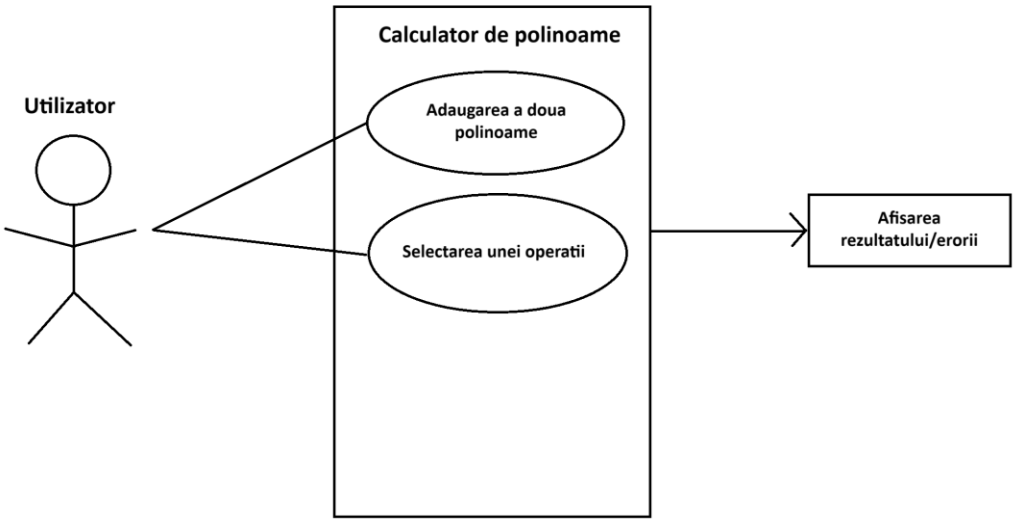
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerintele functionale ale temei sunt:

- Crearea unei interfete grafice prin care utilizatorul poate interactiona cu calculatorul
- Posibilitatea introducerii polinoamelor de catre utilizator
- Posibilitatea selectii operatilor de catre utilizator
- Posibilitatea vizualizarii rezultatului dupa ce operatia s-a efectuat sau aparitia unui mesaj de eroare in caz de esec

Cerintele non-functionale ale temei sunt:

- Calculatorul de polinoame trebuie sa fie intuitiv si usor de folosit
- Performanta trebuie sa fie una ridicata, calculatorul oferind raspunsuri rapide
- Gestionarea eficienta a operatiilor matematice
- Folosirea unor structuri de date eficiente pentru stocarea si efectuarea operatiilor pe polinoame



Cazuri de utilizare

- Odata cu rularea aplicatiei, utilizatorului i se va deschide interfata calculatorului de polinoame
- Utilizatorul va putea introduce 2 polinoame
- De preferat polinoamele sa fie de forma (+-)a*x^b, insa daca se prefera o varianta mai rapida se pot introduce si in felul urmator (+-)axb, operatorii ‘*’ si ‘^’ fiind optionali
- Dupa introducerea a doua polinoame, utilizatorul va avea posibilitatea de a selecta operatia dorita asupra celor doua polinoame
- Operatiile de derivare si intrare vor opera doar pe primul polinom
- Operatia de impartire se va efectua doar in cazul in care exista doua polinoame valide si cel de-al doilea polinom este diferit de 0
- In cazul in care operatiile s-au efectuat cu succes se va afisa in dreptul textului “Result polynomial” rezultatul operatiilor
- In plus la operatia de impartire se va afisa un rezultat si in dreptul textului “Division remainder”
- In cazul in care operatiile nu se pot efectua pe cele 2 polinoame se va afisa un mesaj de eroare in dreptul textului “Error message”

3. Proiectare

Diagrama Package

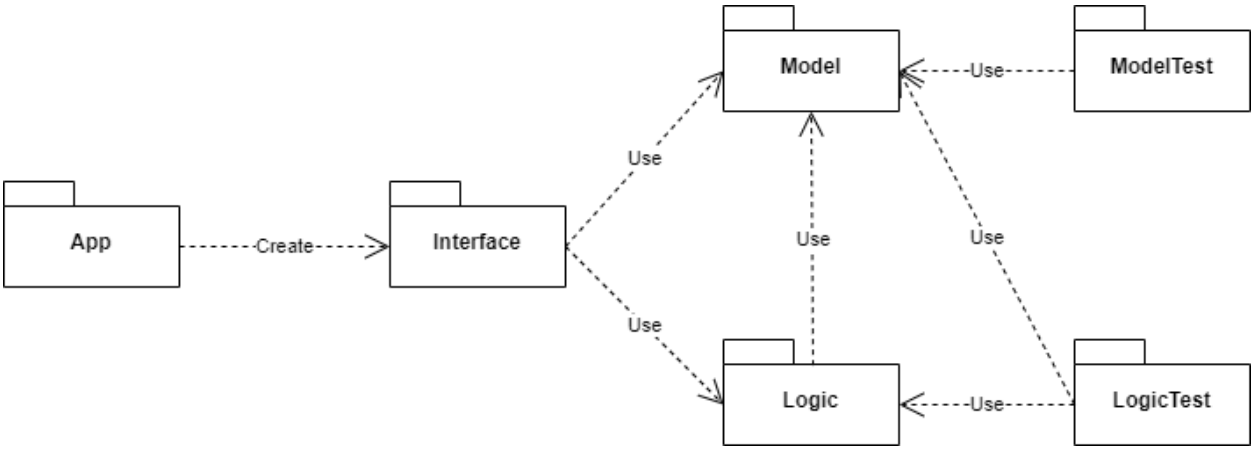
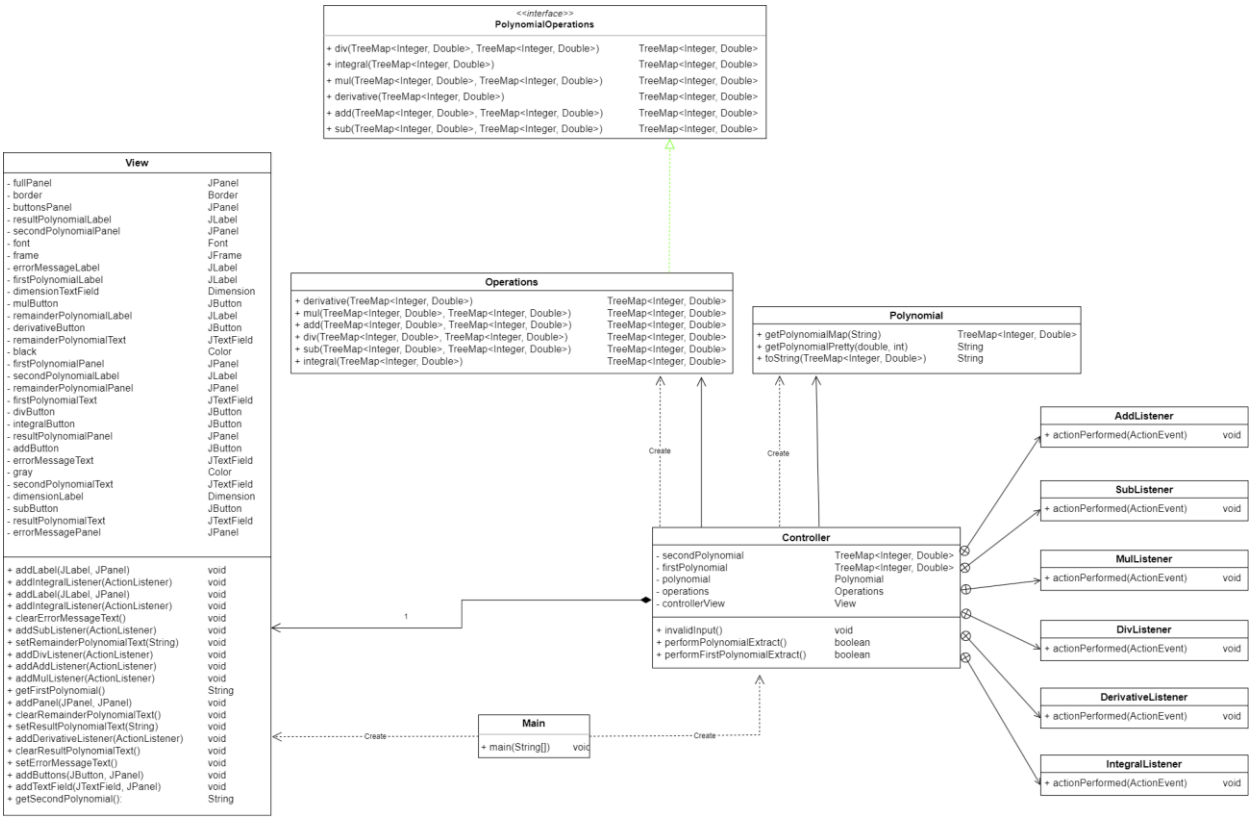


Diagrama UML



4. Implementare

1. Clasa Polynomial

Clasa Polynomial transforma polinomul intr-un TreeMap, respectiv transforma TreeMap-ul intr-un String pentru a putea fi afisat

- Constructori: Constructorul este cel implicit

- Metode:

  - **public TreeMap<Integer, Double> getPolynomialMap(String input)**: Primeste ca input sirul introdus de catre utilizator si il converteste intr-un TreeMap care este returnat la finalul metodei
  - **public String toString(TreeMap<Integer, Double> polynomial)**: Dupa ce s-a efectuat o operatie, TreeMap-ul cu rezultatul ajunge aici, iar aceasta metoda converteste polinomul stocat in TreeMap intr-un String pentru a putea fi trimis catre TextBox-ul care afiseaza rezultatul operatiei.

2. Interfata PolynomialOperations

Aceasta interfata contine toate metodele de operatii care se vor executa pe polinoame in cadrul acestui calculator de polinoame

- Metode:

  - **TreeMap<Integer, Double> add(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial);**

  - **TreeMap<Integer, Double> sub(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial);**

  - **TreeMap<Integer, Double> mul(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial);**

  - **TreeMap<Integer, Double> div(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial);**

  - **TreeMap<Integer, Double> derivative(TreeMap<Integer, Double> polynomial);**

  - **TreeMap<Integer, Double> integral(TreeMap<Integer, Double> polynomial);**

3. Clasa Operations

Clasa Operations implementeaza interfata PolynomialOperations

- Constructori: Constructorul este cel implicit
- Metode
  - **public TreeMap<Integer, Double> add(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial)**: Aceasta metoda face adunarea a doua polinoame, adica aduna numai valorile care au aceeasi cheie si returneaza un polinom rezultat in care s-a facut adunarea
  - **TreeMap<Integer, Double> sub(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial)**: Aceasta metoda face scaderea a doua polinoame, scade din primul polinom valorile celui de al doilea polinom care au aceeasi cheie si returneaza un polinom rezultat in care s-a facut scaderea
  - **TreeMap<Integer, Double> mul(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial)**: Aceasta metoda face inmultirea a doua polinoame, se parcurg ambele polinoame si se inmultesc toate valorile, iar cheile se aduna, in final se returneaza polinomul rezultat.
  - **TreeMap<Integer, Double> div(TreeMap<Integer, Double> firstPolynomial, TreeMap<Integer, Double> secondPolynomial)**: Aceasta metoda face impartirea primului polinom la cel de-al doilea polinom, in caz ca un TreeMap este gol, nu exista sau al doilea polinom este egal cu 0, se va returna “Invalid Input”, altfel se va face impartirea, catul va fi stocat si returnat intr-un polinom rezultat, iar restul va fi stocat in firstPolynomial.
  - **TreeMap<Integer, Double> derivative(TreeMap<Integer, Double> polynomial)**: Folosind formula de derivare $(x^a)' = a*x^{a-1}$, asa se vor deriva toate valorile din TreeMap, iar in final se va returna un polinom rezultat
  - **TreeMap<Integer, Double> integral(TreeMap<Integer, Double> polynomial)**: Folosind formula de integrare $\int x^n dx = x^{n+1}/(n+1)$, asa se vor integra toate valorile din TreeMap, iar in final se va returna un polinom rezultat

4. Clasa View

In clasa View se afla interfata grafica prin care utilizatorul interactioneaza in mod direct cu calculatorul, aceasta clasa contine atribute si metode care lucreaza implicit doar cu interfata grafica

5. Clasa Controller

In clasa Controller se intampla intreg programul

- Atribute

- **private Polynomial polynomial = new Polynomial()**: Cu acest atribut se vor extrage/afisa polinoamele

- **private Operations operations = new Operations()**: Cu acest atribut se vor face operatiile pe polinoame

- **private View controllerView**: Cu acest atribut se va afisa interfata grafica

- **private TreeMap<Integer, Double> firstPolynomial = new TreeMap<>(Collections.reverseOrder())**: Reprezinta primul polinom stocat intr-un Treemap

- **private TreeMap<Integer, Double> secondPolynomial = new TreeMap<>(Collections.reverseOrder())**: Reprezinta al doilea polinom stocat intr-un Treemap

- Constructor

- **public Controller(View view)**: Se face legatura cu interfata si se creeaza subclasele pentru ActionListener-uri

- Metode

- **public boolean performPolynomialExtract()**: In aceasta metoda se vor extrage cele doua polinoame si se va returna adevarat in caz ca acestea au putut fi extrase, fals in caz contrar

- **public boolean performFirstPolynomialExtract()**: In acesta metoda se va extrage doar primul polinom si se va returna adevarat in caz ca acesta a putut fi extras, fals in caz contrar (metoda folosita la derivare si integrare)

- **public void invalidInput()**: In aceasta metoda se va afisa un mesaj de eroare daca polinoamele nu au putut fi extrase, sau o operatie a esuat

- Subclase

- **class AddListener implements ActionListener**: Face adunarea celor 2 polinoame

- **class SubListener implements ActionListener**: Face scaderea celor 2 polinoame

- **class MulListener implements ActionListener**: Face inmultirea celor 2 polinoame

- **class DivListener implements ActionListener**: Face impartirea celor 2 polinoame

- **class DerivativeListener implements ActionListener**: Face derivarea celor 2 polinoame

- **class IntegralListener implements ActionListener**: Face integrarea celor 2 polinoame

6. Clasa Main

In clasa main se creeaza interfata apoi se apeleaza Controllerul pentru a se porni calculatorul si a se putea efectua operatii pe acesta

5. Rezultate

Rezultatele testarii pentru extragerea polinoamelor intr-un TreeMap, extragerea polinoamelor dintr-un TreeMap intr-un String, operatiile de adunare, scadere, inmultire, impartire, derivare, integrare.

Fiecare metoda a fost testata cu cate de 2 exemple

http://localhost:63342/ProiectPolinoame/target/site/surefire-report.html#LogicTest.OperationsTest

Valorile folosite

Polynomial Test

```
public class PolynomialTest {
    4 usages
    Polynomial polynomial = new Polynomial();

    @Test
    public void getPolynomialMapTest() {
        TreeMap<Integer, Double> result;
        TreeMap<Integer, Double> expected = new TreeMap<>(Collections.reverseOrder());

        result = polynomial.getPolynomialMap( input: "x^2+2*x-1");
        expected.put(2, 1.0);
        expected.put(1, 2.0);
        expected.put(0, -1.0);
        assertEquals(expected, result);

        expected.clear();
        result = polynomial.getPolynomialMap( input: "-3*x^4-2*x^2+x^2+35");
        expected.put(4, -3.0);
        expected.put(2, -1.0);
        expected.put(0, 35.0);
        assertEquals(expected, result);
    }
```

```java
    @Test
    public void toStringTest(){
        String result;
        String expected;
        TreeMap<Integer, Double> polynomialMap = new TreeMap<>(Collections.reverseOrder());

        polynomialMap.put(3, -1.0);
        polynomialMap.put(1, -5.0);
        polynomialMap.put(0, 3.0);
        result = polynomial.toString(polynomialMap);
        expected = "- x^3 - 5.0*x + 3.0";
        assertEquals(expected, result);

        polynomialMap.put(5, 2.0);
        polynomialMap.put(1, -3.0);
        polynomialMap.put(0, -2.0);
        polynomialMap.put(3, -5.0);
        result = polynomial.toString(polynomialMap);
        expected = "2.0*x^5 - 5.0*x^3 - 3.0*x - 2.0";
        assertEquals(expected, result);
    }
}
```

Operations Test

```java
public class OperationsTest {
    22 usages
    private TreeMap<Integer, Double> firstPolynomial = new TreeMap<>(Collections.reverseOrder());
    15 usages
    private TreeMap<Integer, Double> secondPolynomial = new TreeMap<>(Collections.reverseOrder());
    24 usages
    private TreeMap<Integer, Double> resultPolynomial = new TreeMap<>(Collections.reverseOrder());
    62 usages
    private TreeMap<Integer, Double> expectedPolynomial = new TreeMap<>(Collections.reverseOrder());
    7 usages
    private TreeMap<Integer, Double> expectedRemainderPolynomial = new TreeMap<>(Collections.reverseOrder());
    12 usages
    Operations operations = new Operations();

    6 usages
    public void firstTestSet() {
        firstPolynomial.clear();
        secondPolynomial.clear();
        firstPolynomial.put(2, 1.0);
        firstPolynomial.put(1, 2.0);
        firstPolynomial.put(0, -3.0);

        secondPolynomial.put(1, 1.0);
        secondPolynomial.put(0, 2.0);
    }
```

```
    public void secondTestSet() {
        firstPolynomial.clear();
        secondPolynomial.clear();
        firstPolynomial.put(4, 1.0);
        firstPolynomial.put(2, -4.0);
        firstPolynomial.put(0, -2.0);

        secondPolynomial.put(2, 1.0);
        secondPolynomial.put(1, 1.0);
        secondPolynomial.put(0, -2.0);
    }

    @Test
    public void addTest() {
        firstTestSet();
        resultPolynomial = operations.add(firstPolynomial, secondPolynomial);
        expectedPolynomial.clear();
        expectedPolynomial.put(2, 1.0);
        expectedPolynomial.put(1, 3.0);
        expectedPolynomial.put(0, -1.0);
        assertEquals(expectedPolynomial, resultPolynomial);

        secondTestSet();
        expectedPolynomial.clear();
        resultPolynomial = operations.add(firstPolynomial, secondPolynomial);
        expectedPolynomial.put(4, 1.0);
        expectedPolynomial.put(2, -3.0);
        expectedPolynomial.put(1, 1.0);
        expectedPolynomial.put(0, -4.0);
        assertEquals(expectedPolynomial, resultPolynomial);
    }
    @Test
    public void subTest() {
        firstTestSet();
        resultPolynomial = operations.sub(firstPolynomial, secondPolynomial);
        expectedPolynomial.clear();
        expectedPolynomial.put(2, 1.0);
        expectedPolynomial.put(1, 1.0);
        expectedPolynomial.put(0, -5.0);
        assertEquals(expectedPolynomial, resultPolynomial);

        secondTestSet();
        resultPolynomial = operations.sub(firstPolynomial, secondPolynomial);
        expectedPolynomial.clear();
        expectedPolynomial.put(4, 1.0);
        expectedPolynomial.put(2, -5.0);
        expectedPolynomial.put(1, -1.0);
        assertEquals(expectedPolynomial, resultPolynomial);
    }
```

```java
@Test
public void mulTest() {
    firstTestSet();
    resultPolynomial = operations.mul(firstPolynomial, secondPolynomial);
    expectedPolynomial.clear();
    expectedPolynomial.put(3, 1.0);
    expectedPolynomial.put(2, 4.0);
    expectedPolynomial.put(1, 1.0);
    expectedPolynomial.put(0, -6.0);
    assertEquals(expectedPolynomial, resultPolynomial);

    secondTestSet();
    resultPolynomial = operations.mul(firstPolynomial, secondPolynomial);
    expectedPolynomial.clear();
    expectedPolynomial.put(6, 1.0);
    expectedPolynomial.put(5, 1.0);
    expectedPolynomial.put(4, -6.0);
    expectedPolynomial.put(3, -4.0);
    expectedPolynomial.put(2, 6.0);
    expectedPolynomial.put(1, -2.0);
    expectedPolynomial.put(0, 4.0);
    assertEquals(expectedPolynomial, resultPolynomial);
}

@Test
public void divTest() {
    firstTestSet();
    resultPolynomial = operations.div(firstPolynomial, secondPolynomial);
    expectedPolynomial.clear();
    expectedRemainderPolynomial.clear();
    expectedPolynomial.put(1, 1.0);
    expectedRemainderPolynomial.put(0, -3.0);
    assertEquals(expectedPolynomial, resultPolynomial);
    assertEquals(expectedRemainderPolynomial, firstPolynomial);

    secondTestSet();
    resultPolynomial = operations.div(firstPolynomial, secondPolynomial);
    expectedPolynomial.clear();
    expectedRemainderPolynomial.clear();
    expectedPolynomial.put(2, 1.0);
    expectedPolynomial.put(1, -1.0);
    expectedPolynomial.put(0, -1.0);
    expectedRemainderPolynomial.put(1, -1.0);
    expectedRemainderPolynomial.put(0, -4.0);
    assertEquals(expectedPolynomial, resultPolynomial);
    assertEquals(expectedRemainderPolynomial, firstPolynomial);
}

@Test
public void derivativeTest() {
    firstTestSet();
    resultPolynomial = operations.derivative(firstPolynomial);
    expectedPolynomial.clear();
    expectedPolynomial.put(1, 2.0);
    expectedPolynomial.put(0, 2.0);
    assertEquals(expectedPolynomial, resultPolynomial);

    secondTestSet();
    resultPolynomial = operations.derivative(firstPolynomial);
    expectedPolynomial.clear();
    expectedPolynomial.put(3, 4.0);
    expectedPolynomial.put(1, -8.0);
    assertEquals(expectedPolynomial, resultPolynomial);
}
```

```java
    @Test
    public void integralTest() {
        firstTestSet();
        resultPolynomial = operations.integral(firstPolynomial);
        expectedPolynomial.clear();
        expectedPolynomial.put(3, 0.33);
        expectedPolynomial.put(2, 1.0);
        expectedPolynomial.put(1, -3.0);
        assertEquals(expectedPolynomial, resultPolynomial);

        secondTestSet();
        resultPolynomial = operations.integral(firstPolynomial);
        expectedPolynomial.clear();
        expectedPolynomial.put(5, 0.2);
        expectedPolynomial.put(3, -1.33);
        expectedPolynomial.put(1, -2.0);
        assertEquals(expectedPolynomial, resultPolynomial);
    }
}
```

6. Concluzii

In aceasta tema dobandit urmatoarele skill-uri:

- Intelegerea conceptelor OOP
- Organizarea si structurarea codului pentru un proiect MVC
- Reutilizarea codului
- Gestiunea complexiatii
- Testarea si depanarea
- Folosirea unui Regex

Posibile dezvoltari

- Pentru operatiile de integrare si derivare sa se faca un nou panel cu un singur polinom ca si valoare de intrare

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://app.diagrams.net/
6. https://maven.apache.org/surefire/maven-surefire-report-plugin/usage.html