**DOCUMENTAȚIE**

**TEMA 1.**

**CALCULATOR DE POLINOAME**

**NUME STUDENT: MARIȚA-THURI ANNA**

**GRUPA: 30221**

**CUPRINS**

1. Obiectivul temei …....................................................................3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare …...4
3. Proiectare și Implementare …....................................................5
4. Testare ….................................................................................11
5. Concluzii …..............................................................................13
6. Bibliografie …..........................................................................14
7. **Obiectivul temei**

Despre obiectivul temei, este important să subliniem că acest proiect are scopul de a crea un calculator de polinoame utilizând limbajul de programare Java. Calculatorul va permite utilizatorilor să efectueze operații matematice cu polinoame, cum ar fi adunarea, scăderea, înmulțirea și derivarea, prin intermediul unei interfețe grafice.

Interfața grafică va permite utilizatorilor să introducă polinoame prin intermediul unor câmpuri de introducere și să selecteze tipul de operație pe care doresc să o efectueze prin intermediul unor butoane. Rezultatele operațiilor vor fi afișate într-o altă zonă a interfeței grafice.

Scopul final al acestui proiect este de a crea un instrument util pentru studenți, profesori și cercetători din domeniile matematicii, fizicii și informaticii, care au nevoie să efectueze operații cu polinoame în mod frecvent.

**2. Analiza problemei, modelare, scenarii,**

**cazuri de utilizare**

### **2.1. Analiza problemei:**

Polinoamele sunt expresii matematice care conțin variabile ridicate la puteri și coeficienți. Ele sunt utile în multe domenii, inclusiv în matematică, fizică și informatică.

Un calculator de polinoame poate fi folositor pentru a efectua operații matematice cu acestea. Utilizatorii pot introduce polinoamele prin intermediul unei interfețe grafice și pot efectua diverse operații, precum adunarea, scăderea, înmulțirea și derivarea.

### **2.2. Modelare:**

Pentru a construi acest calculator, vom crea o clasă Polynom care va conține următoarele metode:

* public Polynom add(Polynom P2) - metoda care aduna doua polinoame
* public Polynom substract(Polynom P2) - metoda care scade doua polinoame
* public Polynom multiply(Polynom P2) - metoda care inmulteste doua polinoame
* public String toString() - metoda care converteste rezultatul intrun string pentru afisare

De asemenea, vom crea o interfață grafică care va permite utilizatorilor să introducă polinoamele și să efectueze operații cu acestea. Aici avem urmatoarele metode:

Butoane:

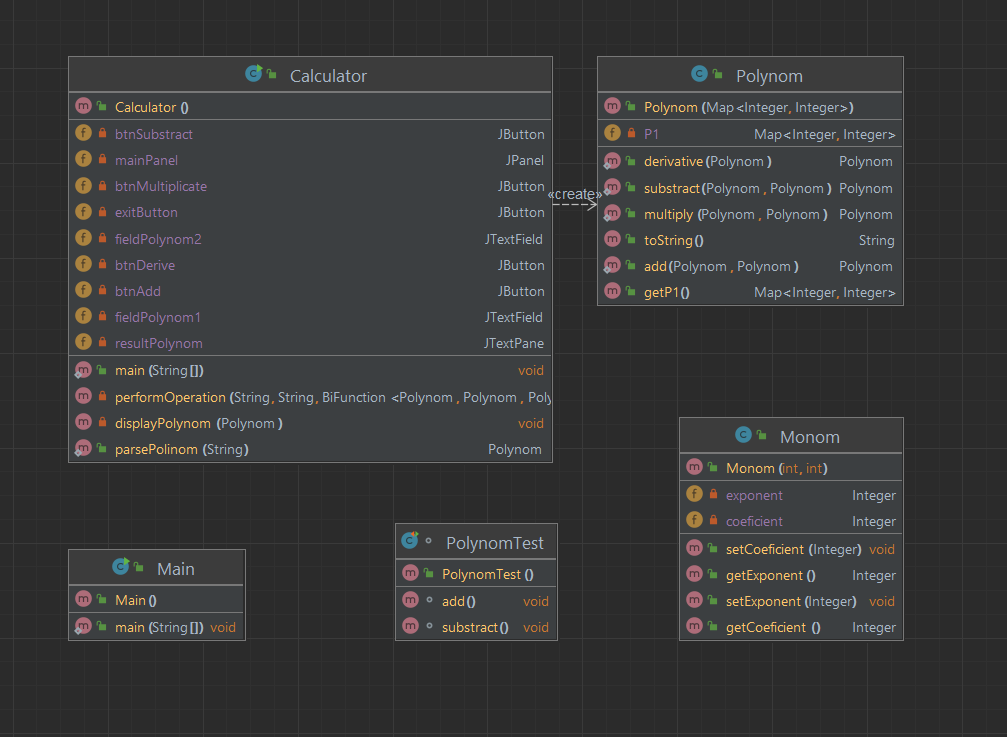
* BtnAdd.addActionListener
* BtnSubstract.addActionListener
* BtnMultiplicate.addActionListener
* exitButton.addActionListener

Metode:

* private Polynom parsePolinom(String polytext)
* private void displayPolynom(Polynom polynom)

1. **Proiectare și Implementare**
   1. **Diagrame UML**

Limba de modelare unificată, cunoscută sub numele de UML, este un limbaj standard pentru descrierea modelelor și specificațiilor software. Inițial, UML a fost dezvoltat pentru a reprezenta complexitatea programelor orientate pe obiect, care se bazează pe structurare programelor în clase și instanțele acestora. Cu toate acestea, datorită eficienței și clarității sale în reprezentarea unor elemente abstracte, UML este utilizat și în afara domeniului IT



* 1. **Clase**
* **Clasa Polynom**

Clasa Polynom reprezintă un polinom matematic cu coeficienți și exponenți, unde fiecare termen este memorat într-un obiect Map cu cheia reprezentând exponentul și valoarea reprezentând coeficientul. Această clasă oferă metode pentru adunare, scădere, înmulțire și derivare a polinoamelor, precum și o reprezentare sub formă de șir de caractere a polinomului.

* **Clasa Monom**

Clasa Monom reprezintă un termen al unui polinom și stochează coeficientul și exponentul acestuia.

* **Clasa Calculator**

Clasa Calculator este o clasa Java care implementează o interfață grafică de utilizator (GUI) pentru a efectua operații aritmetice cu polinoame. Această clasă include o serie de butoane și câmpuri de text, care permit utilizatorului să introducă două polinoame, iar apoi să efectueze operațiile de adunare, scădere, înmulțire și derivare a acestora. Clasa utilizează, de asemenea, o clasă numită Polynom pentru a reprezenta polinoamele și pentru a efectua operațiile corespunzătoare.

* **Clasa Main**

Clasa Main reprezintă punctul de intrare în program și conține metoda main care execută operațiile de testare pentru clasa Polynom și Calculator, cum ar fi parsarea, adunarea, scăderea, înmulțirea și derivatele a două polinoame date.

* **Clasa PolynomTest**

Clasa PolynomTest este o clasă de testare care conține două metode de testare: add() și substract(), care testează funcțiile de adunare și scădere din clasa Polynom. Testele folosesc câteva cazuri de testare pentru a verifica dacă metodele produc rezultate corecte pentru diferite polinoame.

* 1. **Metode**
* **Metode utilizate in clasa Polynom**
  + - 1. **Adunare – metoda add(Polynom P1, Polynom P2) :**

Este o metodă statică ce primește două polinoame și returnează suma lor. Pentru a face asta, metoda creează o copie explicită a primului polinom și îi adaugă termenii celui de-al doilea polinom, dacă aceștia au același exponent. Dacă termenul există deja în primul polinom, atunci coeficientul este adunat cu cel din cel de-al doilea polinom. Dacă termenul există doar în cel de-al doilea polinom, atunci este adăugat la rezultat.

* + - 1. **Scadere – metoda substract(Polynom P1, Polynom P2) :**

Este o metodă statică ce primește două polinoame și returnează diferența lor. Metoda face o copie a primului polinom și îi scade termenii celui de-al doilea polinom, dacă aceștia au același exponent. Dacă termenul există deja în primul polinom, atunci coeficientul este scăzut cu cel din cel de-al doilea polinom. Dacă termenul există doar în cel de-al doilea polinom, atunci este adăugat cu coeficientul schimbat de semn la rezultat.

* + - 1. **Inmultire - multiply(Polynom P1, Polynom P2) :**

Este o metodă statică ce primește două polinoame și returnează produsul lor. Metoda creează un nou polinom gol și îl umple cu toate termenii posibili ce se obțin prin înmulțirea termenilor celor două polinoame. Dacă doi termeni obținuți au același exponent, atunci coeficienții lor sunt adunați. Metoda returnează noul polinom obținut.

* + - 1. **Derivare – derivative(Polynom P1) :**

Este o metodă statică ce primește un polinom și returnează derivata sa. Metoda crează un nou polinom gol și adaugă termenii obținuți prin derivare a fiecărui termen al polinomului dat. Pentru un termen de forma ax^n, derivata sa este nax^(n-1). Metoda returnează noul polinom obținut.

* + - 1. **toString() :**

Este o metodă suprascrisă ce returnează reprezentarea sub formă de șir de caractere a polinomului. Metoda parcurge toți termenii polinomului și îi concatenează la șirul de caractere de ieșire într-un mod coerent. Dacă șirul de ieșire este gol, atunci metoda returnează șirul "0".

* **Metode utilizate in clasa Calculator**

Această clasă folosește o interfață grafică de utilizator (GUI) pentru a permite utilizatorilor să introducă polinoame și să vadă rezultatele operațiilor.

**Metoda performOperation() :**

Această metodă primește un obiect de tip "Polynom" și afișează reprezentarea sub formă de text a acestuia în câmpul de text "resultPolynom". Această metodă este apelată de celelalte metode care realizează operatiile.

**Metode pentru implementarea butoanelor: actionPerformed(ActionEvent e)**

* **btnAdd:** primește două texte reprezentând polinoame din câmpurile de text "fieldPolynom1" și "fieldPolynom2". Folosind metoda "parsePolinom" se creează obiectele "polinom1" și "polinom2" de tip "Polynom". Se apelează metoda "performOperation" cu argumentele "poly1txt", "poly2txt" și "Polynom::add", care va aplica metoda "add" asupra obiectelor "polinom1" și "polinom2" și va afișa rezultatul apelând metoda "displayPolynom".
* **btnSubstract:** primește două texte reprezentând polinoame din câmpurile de text "fieldPolynom1" și "fieldPolynom2". Folosind metoda "parsePolinom" se creează obiectele "polinom1" și "polinom2" de tip "Polynom". Se apelează metoda "performOperation" cu argumentele "poly1txt", "poly2txt" și "Polynom::substract", care va aplica metoda "substract" asupra obiectelor "polinom1" și "polinom2" și va afișa rezultatul apelând metoda "displayPolynom".
* **btnMultiplicate:** primește două texte reprezentând polinoame din câmpurile de text "fieldPolynom1" și "fieldPolynom2". Folosind metoda "parsePolinom" se creează obiectele "polinom1" și "polinom2" de tip "Polynom". Se apelează metoda "performOperation" cu argumentele "poly1txt", "poly2txt" și "Polynom::multiply", care va aplica metoda "multiply" asupra obiectelor "polinom1" și "polinom2" și va afișa rezultatul apelând metoda
* **btnDerive:** primește un text reprezentând un polinom din câmpul de text "fieldPolynom1". Folosind metoda "parsePolinom" se creează obiectul "polinom1" de tip "Polynom". Apoi se aplică metoda "derivative" asupra obiectului "polinom1" pentru a obține derivata și se afișează rezultatul apelând metoda "displayPolynom".

**Metoda displayPolynom() :**

Această metodă primește un obiect de tip "Polynom" și afișează reprezentarea sub formă de text a acestuia în câmpul de text "resultPolynom". Această metodă este apelată de celelalte metode care realizează opera ții cu polinoame pentru a afișa rezultatul final al acestora.

**Metoda parsePolynom() :**

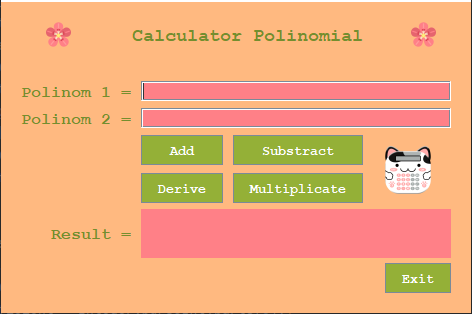
Această metodă ia un string ‘politext’ si creează un obiect Polynom nou. Compilează un regex ca să găsească termeni individuali ai polinomului. Pentru fiecare rezultat găsit din input, metoda extrage coeficientul și exponentul său. Dacă nu conține x, presupune că exponentul este 0.

* 1. **GUI**

Interfața grafică, sau GUI (Graphical User Interface), este o modalitate de interacțiune între utilizator și sistemul de calcul, ce utilizează elemente grafice pentru a afișa informații și a permite utilizatorului să efectueze acțiuni. Scopul unei interfețe grafice este de a face sistemul de calcul mai ușor de folosit pentru utilizatorii non-specializați. În cazul calculatorului de polinoame prezentat, am implementat o interfață grafică user-friendly, pentru a facilita utilizarea acestuia.

Pentru realizarea interfeței mele, am folosit Jawa Swing. Swing este un set de componente și biblioteci de programare în limbajul Java, care permit dezvoltatorilor să creeze interfețe grafice de utilizator (GUI) în mod facil și eficient. Swing oferă o gamă largă de elemente de interfață, cum ar fi butoane, casete de text, panouri, ferestre, etichete și multe altele, pe care dezvoltatorii le pot utiliza pentru a crea aplicații cu o interfață grafică modernă și atractivă. Swing este parte a Java Foundation Classes (JFC) și este inclus în kitul de dezvoltare Java (JDK) de la Oracle.

Calculatorul este o aplicație cu interfață grafică (GUI) pentru a efectua operații matematice pe polinoame. Interfața utilizatorului constă în două câmpuri text, câte un buton pentru fiecare operație, un câmp text pentru afișarea rezultatului și un buton pentru ieșirea din aplicație. În funcție de butonul selectat, se apelează metoda corespunzătoare pentru a efectua operația pe polinoame. Rezultatul este afișat în câmpul de text "resultPolynom".



**Mod de utilizare:**

Ca să putem utiliza interfața trebuie urmați urmatoriii pași:

* + - 1. Introduceți două polinoame dorite în formă explicită.



* + - 1. Selectați operația pe care doriți săo executați.



Ex:

* + - 1. Rezultatul va apărea în spațiul designat, marcat cu ’Result =’.



* + - 1. Dacă nu mai doriți să calculați nimic, putem închide calculatorul cu butonul de Exit.

1. **Testare**

Funcționalitatea și corectitudinea calculatorului se poate testa in mai multe modalități.

Inițial operațiile au fost testate în main creeând doi polinomi la alegere și afișând sugestiv și lizibil rezultatele operațiilor:

*Polynom p1 = Calculator.parsePolinom("-3x^3+2x^2+1x^1");*

*Polynom p2 = Calculator.parsePolinom("1x^0+x^2");*

*System.out.println("p1: " + p1);*

*System.out.println("p2: " + p2);*

*Polynom sum = Polynom.add(p1, p2);*

*System.out.println("p1 + p2: " + sum);*

*Polynom sub = Polynom.substract(p1, p2);*

*System.out.println("p1 - p2 = " + sub);*

*Polynom mul = Polynom.multiply(p1, p2);*

*System.out.println("p1 \* p2 = " + mul);*

*Polynom derivat = Polynom.derivative(p1);*

*System.out.println("derivata lui " + p1 + " este " + derivat);*

Afișând pentru acest exemplu:

*p1: x + 2x^2 - 3x^3*

*p2: 1 + x^2*

*p1 + p2: 1 + x + 3x^2 - 3x^3*

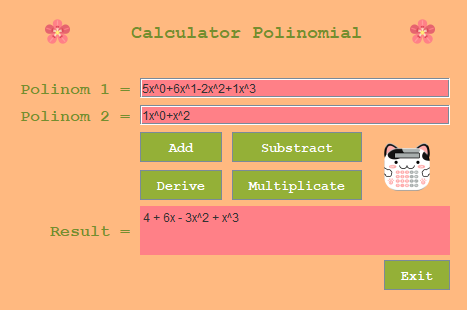
*p1 - p2 = -1 + x + x^2 - 3x^3*

*p1 \* p2 = x + 2x^2 - 2x^3 + 2x^4 - 3x^5*

*derivata lui x + 2x^2 - 3x^3 este 1 + 4x - 9x^2*

O altă modalitatea este testare din interfață:

Ex. Testând scăderea: dacă rezultatul este corect și afișat la fel ca în main știm că este corect.



Ultima modalitate de testare pe care am folosito este cea cu JUnit Testing. Acest cod folosește JUnit pentru a testa clasa Polynom, care conține operații pentru adunare și scădere a polinoamelor. Metodele add și subtract sunt testate prin intermediul metodei assertAll, care verifică multiple afirmații cu ajutorul mai multor aserțiuni de testare. Fiecare afirmație este verificată prin comparația rezultatului așteptat cu valoarea returnată de metoda testată pentru diferite inputuri.

Exemplu pentru înmulțire :

() -> Assertions.assertEquals("0", Polynom.multiply(Calculator.parsePolinom("0"), Calculator.parsePolinom("0")).toString()),

Sau :

() -> Assertions.assertEquals("-x^2 - 4x^3 + 2x^4 + 12x^5 - 9x^6", Polynom.multiply(Calculator.parsePolinom("-3x^3+2x^2+1x^1"), Calculator.parsePolinom("3x^3-2x^2-1x^1")).toString())

Exemplu pentru derivare :

() -> Assertions.assertEquals("0", Polynom.derivative(Calculator.parsePolinom("1")).toString()),

Sau : () -> Assertions.assertEquals("1 + 4x + 9x^2", Polynom.derivative(Calculator.parsePolinom("1x^1+2x^2+3x^3")).toString()),

1. **Concluzii**

În urma dezvoltării acestui calculator și a procesului de învățare a tehnologiilor Java și JUnit Testing, se pot trage câteva concluzii importante.

În primul rând, dezvoltarea unui calculator complex în Java necesită o înțelegere solidă a conceptelor de programare orientată obiect și a modului în care acestea se aplică în Java. Folosirea framework-ului Jawa Swing a fost o alegere bună pentru realizarea interfeței grafice a calculatorului.

În al doilea rând, JUnit Testing este o unealtă extrem de utilă pentru a asigura că aplicația noastră funcționează corect și pentru a identifica eventualele erori sau probleme în timpul procesului de dezvoltare. Testarea unitară poate fi foarte utilă în etapa de dezvoltare și este important să se acorde atenție testelor pentru a asigura că aplicația noastră funcționează corect și este robustă.

În ceea ce privește dezvoltarea ulterioară a acestui calculator, o posibilă îmbunătățire ar fi adăugarea de funcționalități noi, cum ar fi posibilitatea de a realiza operații cu fracții sau de a lucra cu matrice. De asemenea, se poate îmbunătăți interfața grafică a aplicației, adăugând opțiuni de personalizare și un aspect mai modern.

În concluzie, dezvoltarea acestui calculator în Java și utilizarea JUnit Testing au fost o experiență utilă și educativă, care a permis îmbunătățirea abilităților de programare și a cunoștințelor tehnice, și care poate fi folosită pentru dezvoltarea ulterioară a aplicațiilor Java.

1. **Bibliografie**
2. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.*
3. *What are Java classes? -* [*www.tutorialspoint.com*](http://www.tutorialspoint.com)
4. *Regex -* [*https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression*](https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression)
5. *Swing -* [*https://www.javatpoint.com/java-swing*](https://www.javatpoint.com/java-swing)
6. *Testing - https://www.tutorialspoint.com/junit/junit\_test\_framework.htm*