**DOCUMENTAȚIE**

**Tema numărul 1**

NUME STUDENT: Bureția Alex-Dian

GRUPA: 30228

**Cuprins**

1. Obiectivul temei..........................3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare.......................................4
3. Proiectare...................................5
4. Implementare.............................7
5. Rezultate....................................11
6. Concluzii.....................................12
7. Bibliografie..................................12
8. **Obiectivul temei**

Obiectivul principal al temei numărul 1 este realizarea unui calculator pentru polinoame, care implementează cu succes funcțiile dintre polinoame (adunare, scădere, înmulțire, împărțire, derivare și integrarea polinoamelor).

Obiectivele secundare sunt citirea polinoamelor de input și prelucrarea acestora astfel încât să obținem date în conformitate cu modul de programare al Programarii Orientate pe Obiecte( transformarea unui șir de caractere citit într-un polinom după ce am implementat clasa polinom folosindu-ne de clasa monom).Citirea si prelucrarea datelor transmise de utilizator sunt detaliate în capitolul 4.

Implementarea operațiilor reprezintă și ea un obiectiv secundar care contribuie la obiectivul principal al lucrării. Astfel, adunarea, scăderea, înmulțirea primesc ca input 2 polinoame obținute prin prelucrarea șirului de caractere transmis de utilizator si returnează un polinom ca rezultat (împărțirea returneaza o listă de polinoame ca rezultat deoarece aceasta genereaza atât un cât, cât și un rest). Alte două operații implementate în cadrul acestui proiect sunt derivarea și integrarea. Acestea primesc ca dată de intrare un singur polinom și returneaza un alt polinom ca rezultat. Deatlierea operațiilor va avea loc în capitolul cadrul 4.

Interfața grafica este un alt obiectiv secundar al acestui proiect. Utilizatorul nu poate interacționa direct cu aceste operații, de aceea am utilizat o interfață grafică prin care acesta poate stăpâni aceste comenzi, poate introduce polinoame și poate obține rezultate ușor. Interfața grafică implementează astfel butoanele corespunzătoare fiecărei operații, JFrame-ul prin care utilizatorul introduce un șir de caractere ca input, mesajele prin care sunt întoarse rezultatele și meniul care le conține pe toate acestea. Interfața va fi detaliată in cadrul capitolului 4.

Un alt obiectiv secundar este implementarea claselor Java corespunzătoare datelor prelucrate de noi, respectiv monom și polinom. Astfel, am reușit să prelucrez un șir de caratere și să îl transform într-o listă de monoame, adică un polinom. Polinomul rezultat este, mai apoi, input la implementarea fiecărei operații în parte. Aceasta obiectiv secundar este descris mai pe larg în cadrul capitolului 4.

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

Interfata grafica este cea care interactioneaza cu utilizatorul. Aceasta a fost implementata conform cel scrise mai jos. In Jframe-uri sunt citite 2 siruri de caractere, in formatul ,,aX^b-cX^d”. Astfel, este necesara scrierea fiecarui coeficient si putere ca algoritmul sa poata rula corect.

public class View extends JFrame*{* private JFrame frame;  
 private JTextField textField,textField1;  
 private JButton Adunare;  
 private JButton Inmultire;  
 private JButton Scadere;  
 private JLabel Title,pol2;  
 private JButton Egal;  
 private JButton Derivare;  
 private JButton Integrare;  
 private JButton Impartire;  
 private JLabel Rezultat;  
 private JLabel Introduceti;  
 public View*() {* this.setBounds*(*100, 100, 1079, 547*)*;  
 this.setDefaultCloseOperation*(*JFrame.*EXIT\_ON\_CLOSE)*;  
 this.getContentPane*()*.setLayout*(*null*)*;  
  
 Title = new JLabel*(*"Calculator polinomial"*)*;  
 Title.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 25*))*;  
 Title.setBounds*(*398, 23, 233, 46*)*;  
 this.getContentPane*()*.add*(*Title*)*;  
  
 textField = new JTextField*()*;  
 textField.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 textField.setBounds*(*36, 199, 255, 63*)*;  
 this.getContentPane*()*.add*(*textField*)*;  
 textField.setColumns*(*10*)*;  
  
 Adunare = new JButton*(*"Adunare"*)*;  
 Adunare.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 16*))*;  
 Adunare.setBounds*(*516, 148, 115, 35*)*;  
 this.getContentPane*()*.add*(*Adunare*)*;  
  
 Inmultire = new JButton*(*"Inmultire"*)*;  
 Inmultire.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 Inmultire.setBounds*(*516, 227, 115, 35*)*;  
 this.getContentPane*()*.add*(*Inmultire*)*;  
  
 Scadere = new JButton*(*"Scadere"*)*;  
 Scadere.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 Scadere.setBounds*(*745, 148, 121, 35*)*;  
 this.getContentPane*()*.add*(*Scadere*)*;  
  
 Impartire = new JButton*(*"Impartire"*)*;  
 Impartire.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 Impartire.setBounds*(*751, 227, 115, 35*)*;  
 this.getContentPane*()*.add*(*Impartire*)*;  
  
 Derivare = new JButton*(*"Derivare"*)*;  
 Derivare.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 Derivare.setBounds*(*516, 313, 115, 36*)*;  
 this.getContentPane*()*.add*(*Derivare*)*;  
  
 Integrare = new JButton*(*"Integrare"*)*;  
 Integrare.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 Integrare.setBounds*(*750, 317, 110, 35*)*;  
 this.getContentPane*()*.add*(*Integrare*)*;  
  
 Introduceti = new JLabel*(*"Introduceti un polinom"*)*;  
 Introduceti.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 14*))*;  
 Introduceti.setBounds*(*104, 126, 156, 36*)*;  
 this.getContentPane*()*.add*(*Introduceti*)*;  
  
 Egal = new JButton*(*"Egal"*)*;  
 Egal.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 18*))*;  
 Egal.setBounds*(*624, 417, 134, 46*)*;  
 this.getContentPane*()*.add*(*Egal*)*;  
  
 textField1 = new JTextField*()*;  
 textField1.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 14*))*;  
 textField1.setBounds*(*46, 417, 245, 62*)*;  
 this.getContentPane*()*.add*(*textField1*)*;  
 textField1.setColumns*(*10*)*;  
  
 pol2 = new JLabel*(*"Introduceti al doilea polinom\r\n"*)*;  
 pol2.setFont*(*new Font*(*"Tahoma", Font.*PLAIN*, 17*))*;  
 pol2.setBounds*(*46, 303, 215, 57*)*;  
 this.getContentPane*()*.add*(*pol2*)*;  
 this.setVisible*(*true*)*;  
 *}*

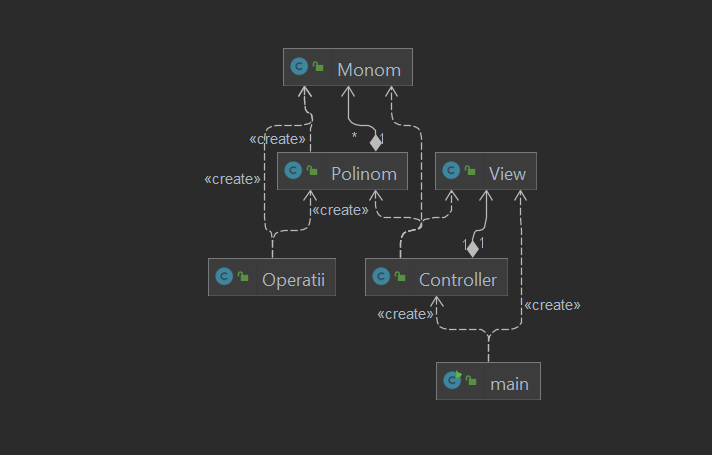
**3.Proiectare**

Principiile programării OOP sunt utilizate cu succes în cadrul acestei aplicații. Încapsularea este utilizată prin ascunderea accesului unei variabile dintr-o anumită clasă de către celelalte clase, iar singura metodă de a prelucra valoarea acesteia rămân metodele de set și get.Astfel, am utilizat cuvântul cheie ,, private” în fața variabilelor din clase.

Compoziția este un alt principiu OOP utilizat in cadrul aplicației. Modul de realizare al clasei View si Controller presupune folosirea compoziției deoarece in clasa View am folosit un Controller și în clasa Controller am folosit un View.

Moștenirea ese utilizată în cadrul proiectului deoarece în cadrul clasei View am utilizat cuvântul cheie ,,extends”. Moștenirea este principiul OOP conform căruia funcționalitățile unei clase sunt preluate și chiar extinse de altă clase.

Diagrama UML specific acestui proiect a fost realizată in IDE-ul IntelliJ și arată în acest mod:



Astfel, putem observa care sunt relațiile dintre clase.

În implementarea acestui proiect am utilizat 2 clase Models (Polinom și Monom), aceste clase construiesc cu succes operațille pe polinoame. În plus, am creat o clasă View pentru realizarea interfeței grafice a proiectului și, in pachetul de controllere, am realizat 2 clase (Controller- utilizată pentru a programa butoanele din proiect, și Operatii- utilizată pentru a definii operatiile propriu zise pe polinoame).

**4.Implementare**

În cadrul acestui capitol voi prezenta fiecare clasă in parte.

Clasa Monom definește un monom, pe baza coeficientului și a puterii sale. Un monom este un termen dintr-un polinom. În plus, pentru a putea accesa câmpurile sale (putere si coeficient), am utilizat metode de get și set.

Clasa Polinom este definită ca o listă de monoame. Constructorul clasei implementează citirea și transformarea șirului de caractere citit din Jframe corespunzător claselor Monom și Polinom. Astfel, pentru citire am luat o lista de caractere in care am pus doar caracterele numere si caracterele ‘-‘, ‘+’. După ce parcurg lista nou creată, transform fiecare caracter numeric în parte in număr (de tip float pentru coeficient și int pentru putere). Elementele aflate pe poziție impară sunt puteri, iar elementele aflate pe poziție pară sunt coeficienții. În plus, dacă înaintea unei cifre este caracterul ‘-‘, atunci coeficientul, căci doar el poate fi negativ, este înmulțit cu -1, fiind astfel negativizat.

Clasa View conține interfața grafică a proiectului, cea cu care interacționează utilizatorul. Am utilizat în cadrul acestei clase câteva butoane pentru operații ( Adunare, Inmultire, Scadere, Derivare , Integrare, Impartire), dar și două JtextField-uri pentru a putea citit textele (polinoamele) introduse de utilizator. Astfel, informația citită din JtextFeild-uri este, mai apoi, transmisa catre constructorul clasei Polinom pentru a fi transformată informația în monoame si apoi, polinoame. Date de intrare valide presupun existența, în fiecare termen, atât a coeficientului(chiar daca e 1), cat și a puterii(chiar dacă e 1 sau 0. Un exemplu de dată de intrare validă este: 3X^22-234X^1-90X^0.

Clasa Controller este strâns legată de clasa View pentru că prima clasă realizează programarea butoanelor prezente in ce de-a doua clasă mai sus menționată. Deci, sunt programate butoanele Adunare, Scadere, Inmultire, Impartire, Derivare și Integrare. Butoanele Adunare Scadere, Inmultire și Impartire citesc, odată ce sunt apăsate, cele 2 texte din JtextFiedl-uri și le transmit ca parametru constructorului clasei Polinom, care, mai departe, le separă dupa metoda descrisă mai sus și reușește cu succes să le transoforme în polinoame. Mai apoi, aceste controllere de butoane transmit ca parametru polinoamele obținute metodelor de realizare a operațiilor de Adunare, Scădere, Inmulțire și Împărțire din clasa Operatii, prin crearea unui obiect de clasa Operatii. În final, polinomul rezultat este afișat cu ajutorului unui mesaj care se deschide în fereastră noua, iar polinoamele introduse de utilizator sunt șterse din JtextField.

Asemănator acestei metode actionează si controllerele de la butoanele de Integrare și Derivare, dar acestea citesc doar primul polinom și îl prelucrează. De observat că în cadrul controller-ului de împărțire sunt returnate 2 valori, și anume atât câtul cât și restul, care sunt afișate mai apoi in mesajul de tip pop-up. Derivarea presupune înmulțirea puterii cu coeficientul fiecarui monom, iar, mai apoi, decrementarea cu o unitate a puterii monomului.Derivarea a fost implementată direct in Controller. La fel și integrarea, care constă în incrementarea cu 1 a puterii monomului, iar mai apoi împărțirea coeficientului monomului cu puterea astfel nou obținută. Motivul pentru care am ales să implementez derivarea și integrarea în controller a fost acela că nu sunt atinse cele 30 de rânduri de cod maxime admise la nivel de metodă, iar astfel nu deranjează în cod.

Clasa Operatii implementează cu succes operațiile pe polinoame. Adunarea realizează o parcurgere pe cele 2 liste de monoame, iar dacă găsim aceleași puter le adunăm coeficienții și puterea este setata pe -1 pentru a ști să nu le mai cautăm in lista a doua, iar mai apoi cea de-a doua listă e parcursă și la rezultat sunt alipite monoamele neparcurge din primul ( care au puterea diferita de -1). Scăderea este realizată prin inmulțirea fiecărui coeficient din a doua listă cu -1 și e implementată ca o adunare. Înmulțirea presupune adunarea puterilor și înmulțirea coeficienților fiecărei perechi de monoame din cele 2 liste diferite de monoame. Împărțirea a fost implementată cu ajutorul metodelor auxiliare getMonomMax, prin care am calculat monomul maxim din polinom. Împărțirea a fost calculată prin împarțirea coeficienților monoamelor maxime din polinom și scăderea puterilor acestora.

In clasa main am creat un obiect de tip View si altul de tip Controller, pentru a programa butoanele existente:

public class main *{* public static void main*(*String*[]* args*) {* View view1 = new View*()*;  
 Controller controller = new Controller*(*view1*)*;  
 *}  
}*

Exemplu de cod din monom:

public class Monom *{* private float coeficient;  
 private int putere;  
  
 public Monom*(*float coeficient, int putere*) {* this.coeficient = coeficient;  
 this.putere = putere;  
 *}* public void setCoeficient*(*float coeficient*) {* this.coeficient = coeficient;  
 *}* public void setPutere*(*int putere*) {* this.putere = putere;  
 *}* public float getCoeficient*() {* return coeficient;  
 *}* public int getPutere*() {* return putere;  
 *}  
}*

Exemplu de cod din înmulțire.

public List*<*Monom*>* inmultire*(*Polinom q, Polinom q2*){* List*<*Monom*>* a3= new ArrayList*<*Monom*>()*;  
 // Polinom rezultat=new Polinom(a3);  
 for*(*Monom w2:q2.getMonom*()){* float coef=1;  
 int putere1=0;  
 for*(*Monom w1: q.getMonom*()){* coef=w1.getCoeficient*()* \*w2.getCoeficient*()*;  
 putere1=w1.getPutere*()*+w2.getPutere*()*;  
 Monom c=new Monom*(*coef,putere1*)*;  
 a3.add*(*c*)*;  
 *}  
 }* List*<*Monom*>* a4= new ArrayList*<*Monom*>()*;  
 for*(*Monom w2:a3*){* float sum=0;  
 sum=w2.getCoeficient*()*;  
 for*(*Monom w1: a3*){* if*(*w1.getPutere*()* ==w2.getPutere*()* && w1.getPutere*()* !=-1 && w1!=w2*){* sum+=w1.getCoeficient*()*;  
 //w1.setCoeficient(0);  
 w1.setPutere*(*-1*)*;  
 *}  
 }* Monom c=new Monom*(*sum,w2.getPutere*())*;  
 a4.add*(*c*)*;  
 *}* return a4;  
*}*

**5. Rezultate**

Rezultatele au fost obtinute in urma testarii proiectului utilizand Junit. Astfel, a trebuit sa configurez fisierul pom.xml pentru a permite Junit-ului sa poata testa proiectul, iar mai apoi am rulat efectiva metodele iar rezultatele acestora le-am comparat cu cele care stiam ca trebuiam sa le obtin.

public class AddOperationTest *{* Polinom u=new Polinom*(*"3X^45+6X^21-5X^0"*)*;  
 Polinom w=new Polinom*(*"1X^45+8X^10-2X^2"*)*;  
 Polinom u1=new Polinom*(*"3X^45"*)*;  
 Polinom w1=new Polinom*(*"8X^10"*)*;  
 Polinom m1=new Polinom*(*"24X^55"*)*;  
 Polinom m=new Polinom*(*"4X^45+6X^21-5X^0+8X^10-2X^2"*)*;  
 @Test  
 public void addTest*(){* Operatii op=new Operatii*()*;  
 List*<*Monom*>* g= op.adunare*(*u,w*)*;  
 Polinom max=new Polinom*(*""*)*;  
 max.setMonom*(*g*)*;  
 for*(*Monom qwer:m.getMonom*()){* qwer.setCoeficient*(*qwer.getCoeficient*()*\**(*-1*))*;  
 *}* List*<*Monom*>* qq=op.scadere*(*max,m*)*;  
 int ok=0, ok1=0;  
 for*(*Monom www:qq*)* if*(*www.getCoeficient*()* !=0*)* ok=1;  
 Assert.*assertEquals(*ok,ok1*)*;  
 *}* public void addTest1*(){* Operatii op=new Operatii*()*;  
 List*<*Monom*>* g= op.inmultire*(*u1,w1*)*;  
 Polinom max=new Polinom*(*""*)*;  
 max.setMonom*(*g*)*;  
 Assert.*assertEquals(*m1,max*)*;  
 *}*

Astfel stim ca daca adun 2 polinoame si apoi scad rezultatul din suma pe care o stiu, rezultatul final trebuie sa fie polinomul nul, iar in acest mod am reusit sa verific atat scaderea cat si adunarea polinoamelor.

**6.Concluzii**

Tema aceasta este una foarte utilă deoarece m-a ajutat sa lucrez cu clasele mai bine. Chiar dacă la început îmi părea greu o abordare orientată pe obiecte cu clase polinom și monom, mai apoi mi-am dat seama că operațiile sunt mai ușor de implementat cu ajutorul acestor clase. În plus, am învățat multe la capitolul lucru cu șiruri de caractere și în prelucrarea acestora. Concluzionând, acest proiect a venit ca sprijin în întărirea bazelor învățate semestrul anterior la POO, printre care și controllerele butoanelor.

Ulterior, dezvoltările care pot fi aduse proiectului sunt de natură grafică în special (îmbunătățirea aspectului, schimbarea fontului și a modului în care sunt afișate rezultatele). La nivel de optimizare, pot fi aduse îmbunătățiri la modul de implementare al operațiilor. Dar și la citirea din șir de caractere, căci pot fi gasite metode sa nu avem nevoie de un alt șir auxiliar de caractere în care să stocăm numerele și semnele acestora.

Consider ca proiectul este unul foarte util deoarece ne face sa lucram practic si sa ne adaptam la piata muncii, pe mai departe.

1. **Bibliografie:**
2. Algoritmul de impartire primit ca model de la laborator
3. Indrumatorul de laborator