# DOCUMENTAȚIE

TEMA 1

Nume : Rîpaș Bianca Ioana

Grupa: 30228

Cuprins

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 4](#_Toc95297886)

[3. Proiectare 6](#_Toc95297887)

[4. Implementare 7](#_Toc95297888)

[5. Rezultate 11](#_Toc95297889)

[6. Concluzii 13](#_Toc95297890)

[7. Bibliografie 14](#_Toc95297891)

1.Obiectivul temei

Obiectivul principal al acestei teme este cel de a realiza un calculator de polinoame(in Java) , care sa implementeze operatiile de adunare si scadere ,inmultire,derivare,impartire si integrare.De asemenea, va fi nevoie si de implementarea unei interfete grafice pentru utilizarea prietenoasa a programului.

Obiectivele secundare :

|  |  |
| --- | --- |
| Analizarea problemei si identificarea nevoilor | Pagina 4 |
| Design ul calculatorului de polinoame | Pagina 9 |
| Implementarea calculatorului de polinoame | Pagina 7 |
| Testarea calculatorului de polinoame | Pagina 11 |

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

Use Case : adunarea , scaderea si inmultirea polinoamelor

Actorul primar : user

Scenariu de success :

1. User-ul insereaza doua polinoame in interfata
2. User-ul selecteaza butonul pentru adunare/scadere/inmultire
3. Calculatorul de polinoame face adunarea/scaderea/inmultirea si afiseaza rezultatul

Alt scenariu:

1. User-ul insereaza doua polinoame in interfata , dar gresite ( pune .. in loc de . sau alta variabila in loc de x)
2. Se arunca exceptie in aplicatia InteliJ

Use Case : derivarea si integrarea polinoamelor

Actorul primar : user

Scenariu de success :

1. User-ul insereaza un polinom in interfata
2. User-ul selecteaza butonul pentru derivare/integrare
3. Calculatorul de polinoame face derivarea/integrarea si afiseaza rezultatul

Alt scenariu:

1. User-ul insereaza un polinom in interfata , dar gresite ( pune .. in loc de . sau alta variabila in loc de x)
2. Se arunca exceptie in aplicatia InteliJ

Cerinte functionale :

1.Calculatorul de polinoame trebuie sa ne permita sa inseram polinoame

2.Calculatorul de polinoame trebuie sa ne permita sa alegem operatia Dorita

3.Calculatorul de polinoame trebuie sa adune , scada si inmulteasca doua polinoame

4.Calculatorul de polinoame trebuie sa deriveze si integreze un polinom

5. Calculatorul de polinoame trebuie sa afiseze constanta c la integrare

6. Calculatorul de polinoame trebuie sa nu afiseze x daca exponentul este 0

7. Calculatorul de polinoame trebuie sa nu afiseze +0 daca scaderea dintre niste coeficienti este 0

8. Calculatorul de polinoame trebuie sa afiseze doar o data un exponent, nu mai multi termeni cu acelasi exponent in cazul inmultirii

Cerinte non-functionale :

1. Calculatorul de polinoame trebuie sa fie usor de folosit pentru user
2. Calculatorul de polinoame trebuie sa afiseze polinomul in ordinea descrescatoare a exponentilor
3. Calculatorul de polinoame trebuie sa afiseze exceptii daca user-ul nu a introdus input-uri bune

3.Proiectare

Proiectul Calculator de Polinoame contine elemente POO, precum :

Contine doi constructori , un lucru des intalnit in POO , deoarece este un mod de a creea obiecte de acelasi tip , dar in moduri diferite.Primul constructor creeaza un TreeMap gol , iar al doilea initializeaza TreeMap-ul.

TreeMap-ul de monomials este declarat private.

Metodele sunt publice.

Se foloseste interfata Map prin Map.Entry<Integer,Double> pentru a itera prin Map-uri si pentru a obtine cheile si valorile pentru fiecare element.

Colectia Map este folosita pentru a stoca monomii polinomului in chei si valori.

Structura de date pe care am folosit-o preponderent este Tree,aceasta este o implementare a Map ,una sortata.

Diagrama :

A picture containing table

Description automatically generated

4.Implementare

Clasa Polynomial

Clasa Polynomial are doi constructori.Primul constructor initializeaza monoamele cu ajutorul unui TreeMap , dar am folosit reverseOrder din clasa Collections pentru a sorta monoamele in ordine descrescatoare a exponentilor .Al doilea constructor (Map<Integer,Double> monomials )(Map este o structura de date) si creeaza un obiect de tipul Polynomial cu monoame.Fiecare cheie reprezinta un exponent , iar valorile reprezinta coeficientii fiecarui termen.

getMonomials() returneaza Map<Integer,Double> care contine monoamele polinomului. Integer este pentru exponent, iar Double pentru coeficient ( am ales sa folosesc Double pentru a reusii operatia de integrare sub forma corecta) si setMonomials(Map<Integer,Double>(monomials)) seteaza monoamele polinomului cu unul dat ca parametru.

Pentru a adauga monoame am folosit addMonomial(int exponent,double coeficient) . Aceasta metoda adauga exponentii si coeficientii , dar in cazul in care exponentul unui monom deja exista , coeficientul se va schimba.

Dupa aceste metode scurte , am implementat metodele pentru realizarea operatiilor.

Prima metoda este add(Polynomial p2) care realizeaza adunarea. Aceasta returneaza un nou obiect de tip Polynomial care contine suma dintre Polynomial p2( acesta este inputul ) si cel curent.Se incepe cu creearea noului obiect de tip Polynomial , iterez prin fiecare monom al primului polinom si apoi pentru fiecare monom din al doilea polinom adaug coeficientul monomului la coeficientul corespunzator din primul polinom sau creez alt monom daca in primul polinom nu a existat exponentul respectiv.Aici folosesc getMonomials despre care am scris mai sus si addMonomials pentru a le adauga in result(polinomul in care va fii stocata adunarea).

Metoda subtract(Polynomial p2) care este asemanatoare cu adunarea se realizeaza la fel , dar diferenta majora este ca se neaga coeficientii polinomului 2 inainte de a fii adaugati la coeficientii primului polinom care se afla in polinomul result.

Metoda multiply(Polynomial p2) inmulteste doua polinoame si returneaza un nou polinom . Am mentinut aceasta strategie la toate operatiile , cea de a creea un nou polinom result.

Se incepe prin iterarea prin fiecare monom al primului polinom si prin fiecare monom din al doilea polinom ,se inmultesc coeficientii monomilor si adun exponentii.Verific daca exponentul curent se gaseste deja in polinomul result folosind ContainsKey ( metoda a Map-ului) .Daca da , exista deja un exponent atunci coeficientul este adaugat la coeficientul monomului deja existent in result , dar in caz contrar , se adauga un nou monom in result cu exponentul respectiv si coeficientul respectiv.Acest lucru il verific pentru ca sa nu existe mai multe monoame cu acelasi exponent , adica daca in urma inmultirii va rezulta 2x^3 + 5x^3 eu in polinomul result voi avea doar 7x^3.

Urmatoarea metoda este differentiate() , care deriveaza un polinom.Se incepe , bineinteles , prin creearea unui nou polinom , dupa aceea se itereaza prin polinomul care trebuie sa fie derivat. Coeficientul trebuie sa fie inmultit cu exponentul , iar exponentul -1. Nu am realizat aceste doua operatii in acelasi loc pentru a nu modifica exponentul , daca am fii modificat exponentul ( exponent -1 ) nu ar fii fost potrivit pentru inmultirea cu coeficientul , de aceea am ales sa modific prima data coeficientul , iar cand adaug in result monoamele , cu ajutorul metodei addMonomial voi modifica exponentul .

Ultima metoda pe care am ales sa o implementez este operatia de integrare ( integrate()).

Aceasta se aseamana cu derivarea , este chiar la fel , diferenta este in operatiile pe care le realizez asupra coeficintului si exponentului. Coeficientul va fii egal cu : coeficientul vechi impartit la( exponent+1). Exponentul va fii si el schimbat , se va adauga plus 1.

Dupa implementarea operatiilor , am implementat o metoda parsePolynomial(String input) pentru a transforma String-ul din interfata ,pe care il introduce utilizatorul intr-un polinom.

Metoda incepe cu creearea unui TreeMap ( Integer pentru exponent si Double pentru coeficient) cu numele de monomials.String-ul regex folosit (?=[-+]) verifica String-uri care sunt urmate ori de + ori de - , el imparte sirul la fiecare + sau – si pastreaza semnul la inceputul substring-ului.Trec peste orice sir gol.Verific daca exista x in sir pentru ca asa inseamna ca este vorba de un polinom.Impart in doua secvente una de dinainte de x care contine coeficientul si una cu exponentul.Transform exponentul intr-un Int daca exista , daca nu , ii dau valoarea 1.Transform coeficientul in 1 sau -1 din prima parte sau cu parseDouble daca nu a indeplinit primele conditii si nu a fost – sau +.Daca nu exista x , se face parseDouble pentru coefficient.La final , adaug in monomials fiecare coefficient la exponentul correspondent, iar dupa aceea intr-un Polynomial result.

Ultimul lucru de care am nevoie pentru a finaliza metodele din clasa Polynomial este de un toString() pentru a afisa rezultatul operatiilor.

Dores sa convertesc un polinom intr-un sir de caractere.

Declar result de tipul StringBuilder pe care il utilizez pentru a construi sirul de caractere.Iterez prin Map si verific mai multe scenarii. Daca coeficientul este 0 , trec la urmatoarea iteratie.Daca coeficientul este pozitiv , adaug semnul “+”, dar numai daca sirul nu e gol, in cazul unui coeficient negativ , adaug “-“ in sirul de caractere, dar modific si coeficientul sa fie pozitiv pentru a afisa corect rezultatul prin Math.abs(coefficient). Un alt scenariu , daca exponentul este 0 adaug coeficientul in sir , daca e 1 adaug coefficient si x , iar daca e mai mare decat 1 adaug coefficient si x si exponent.

La final, daca sirul e gol , afisez 0. Se returneaza String-ul rezultat.

Clasa App

Aceasta clasa deschide o fereastra folosind clasa Window din acelasi pachet , apoi creez o instanta a acestei clase si o stochez in fPrincipala. Metoda main se termina si se deschide o fereastra, pentru ca am creat un obiect de tip Window.

Clasa Window

In aceasta clasa am editat interfata care se afiseaza utilizatorului.Pentru inceput , fereastra are dimensiunile 500,500 si are 7 obiecte de tip JTextField pentru a introduce polinoamele si pentru a afisa rezultatul operatiilor.Folosesc obiecte de tip JLabel , 3 la numar pentru a afisa titlul si pentru a stii utilizatorul unde sa introduca polinomul 1 , respectiv polinomul 2.

Operatiile se realizeaza atunci cand utilizatorul apasa pe butonul corespondent fiecarei operatii. Acest lucru l-am reusit prin “btnD.addActionListener” ( buton pentru derivare) si prin metoda public void actionPerformed(ActionEvent e) , adica atunci cand butonul este apasat trebuie sa se intample ce e scris in interiorul acestei functii , adica am declarat un String p1String , unde se pune textul din TextField-ul polinom 1(TextField) , iar dupa aceea am declarat un obiect de tip Polynomial p1 , pe care il initializez cu ajutorul metodei parsePolynomial cu parametrul p1String. Un nou Polynomial result este declarat care apeleaza metoda differentiate , iar dupa aceea in TextFieldD se pune Polynomial result , dar apeland metoda toString().Acest lucru e identic si pentru integrarea unui polinom , dar cu precizarea ca integrarea se realizeaza pentru polinomul 2.

In cazul operatiilor adunare,scadere si inmultire unde este nevoie de doua polinoame ,procesul se aseamana foarte mult. Am nevoie de doua String-uri p1String si p2String si doua Polynomial. P1String ia textul din TextField\_1 , iar p2String din TextField.

Polynoamele p1 si p2 apeleaza metoda parsePolynomial . Polynomial result este initializat cu = p1.subtract(p2) si dupa aceea se afiseaza rezultatul in textField cu ajutorul metodei toString din clasa Polynomial si setText.

Toate operatiile au nevoie de addActionListener si de metoda public void actionPerformed(ActionEvent e) pe care le -am folosit de fiecare data.

Toate textField-urile si JLabe-urile , butoanele au o dimensiune anume , setata de mine pentru a avea o interfata cat mai lizibila si inteligibila.

Cu ajutorul acestei metode getContentPane().add(**textField\_1**); am adaugat TextField-urile JLabe-urile si butoanele si am setat culoarea calculatorului prin : getContentPane().setBackground(Color.***lightGray***);, iar culoarea pentru rezultatul calculelor prin textFieldI.setForeground(Color.RED); si pentru introducerea polinomului lblNewLabel.setForeground(Color.***RED***);.

In aceasta clasa se folosesc sintaxe specifice unei interfete.

Graphical user interface

Description automatically generated with medium confidence

Dupa utilizarea acelor sintaxe , aceasta este interfata calculatorului de polinoame.

5.Rezultate

Am folosit Junit pentru testarea operatiilor.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

**public void** AdunareTest() {  
 Polynomial polynomial1=**new** Polynomial();  
 Polynomial polynomial2=**new** Polynomial();  
 polynomial1.addMonomial(3,2.0);  
 polynomial1.addMonomial(2,3.0);  
 polynomial2.addMonomial(3,4.0);  
 polynomial2.addMonomial(2,3.0);  
 Polynomial rezultat=**new** Polynomial();

rezultat=polynomial1.add(polynomial2);  
 *assertEquals*(**"6.0x^3 + 6.0x^2"**,rezultat.toString()); }

Acesta este modul in care am implementat testarea adunarii , dar este la fel si pentru celelalte operatii.Am inceput prin creearea metodei AdunareTest , iar dupa aceea am creat 2 obiecte de tip Polynomial si am adaugat in fiecare monoame.Am creat Polynomial rezultat unde va fii stocat rezultatul adunarii si am apelat metoda add.La final, am verificat daca este egal rezultatul adunarii cu rezultatul calculate de mine prin assertEquals.

6.Concluzii

Dupa finalizarea acestui proiect, sintaxele necesare in Java au devenit accesibile pentru mine, modul de a scrie cod in acest limbaj de programare imi este mult mai familiar. De asemenea , datorita faptului ca am folosit o interfata grafica mi-a dezvoltat imaginatia pentru a creea un design si a o pune in functiune a fost foarte util pentru dezvoltarea mea in acest domeniu.

Am invatat sa lucrez cu Map , ceea ce pentru mine a fost nou , tot timpul as fii ales variante ce contin ArrayList , dar nu Map. In schimb , dupa finalizarea acestui proiect pot spune ca voi alege sa folosesc mult mai des Map,TreeMap.

Am dobandit multe cunostinte in scrierea metodelor scurte , care sa functioneze bine sa nu se repete cod degeaba ci totul sa fie succinct , dar sa functioneze pe toate cazurile posibile, un aspect foarte important . Datorita limitarii de clase mai mici sau egale cu 300 de randuri am putut sa imi dezvolt aceasta capabilitate de a scrie un cod succinct si de a utiliza metode daca un anume caz se repeta.

7.Bibliografie

<https://www.baeldung.com/java-treemap>

<https://www.geeksforgeeks.org/treemap-in-java/>

<https://javapointers.com/java/java-se/actionlistener/>

<https://www.w3schools.com/java/java_regex.asp>

<https://www.javatpoint.com/java-regex>

<https://www.tutorialspoint.com/java/number_parseint.htm>

<https://www.geeksforgeeks.org/stringbuilder-append-method-in-java-with-examples/>

<https://www.regular-expressions.info/lookaround.html>