DOCUMENTATIE

TEMA *1*

NUME STUDENT: Codrea Andrei Cristian

GRUPA: 30226

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\ytpen\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297891)

1. **Obiectivul temei**

Obiectivul temei este acela de a realiza un calculator de polinoame care să efectueze pe ele următoarele operații: adunarea, scăderea, înmulțirea, împărțirea, derivarea și integrarea. În realizarea temei trebuie respectate anumite bune practici pentru scriere codului: clase și metode mici ca număr de linii, crearea de clase relevante pentru rezolvarea problemei, implementarea unei interfețe grafice, ș.a.m.d.

În final vom avea un calculator în are utilizatorul va putea introduce două polinoame care să efectueze operațiile menționate mai sus.

1. **Analiza problemei**

Pentru a realiza o aplicație funcțională a trebuit să am o interfață cu care utilizatorul să poată interacționa și un set funcțional de operații.

Table

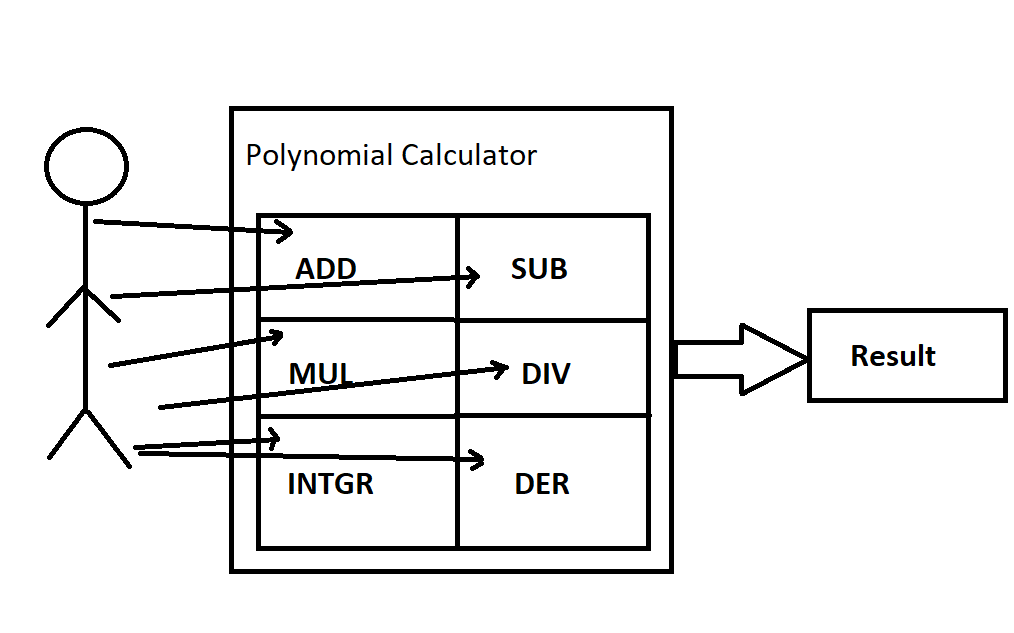
Description automatically generated

Utilizatorul poate astfel introduce două polinoame și alege o operație care să se efectueze.

Pot fi introduse polinoame de forma: .

În cazul operațiilor de integrare și derivare operațiile se efectuează doar pe primul polinom.

Modul de utilizare al calculatorului este descris de următoarea diagramă Use Case:



Add, Sub, Mul, Div, Intgr și Der reprezintă operațiile de adunare, scădere, înmulțire, împărțire, respectiv derivare.

Pentru realizarea operațiilor am folosit mai multe metode din cadrul clasei Polynomial care va fi descrisă mai mult în capitolele 2 și 3. Ca idee generală, toate operațiile sunt implementate în această clasă care are la bază blocul principal din construcția polinoamelor și anume monomul, care este și el la rândul lui o clasă.

1. **Proiectare**

Din punct de vedere al proiectării OOP, programul urmează modelul Model View Controller de proiectare.

Model este pachetul în care este păstrată logica aplicației. Aceasta conține clasele: Monomial, care implementează un monom, Polynomial, care definește și implementează un polinom și operațiile ce se pot efectua pe acesta și Parser care se ocupă de parsarea input-ului dat de utilizator.

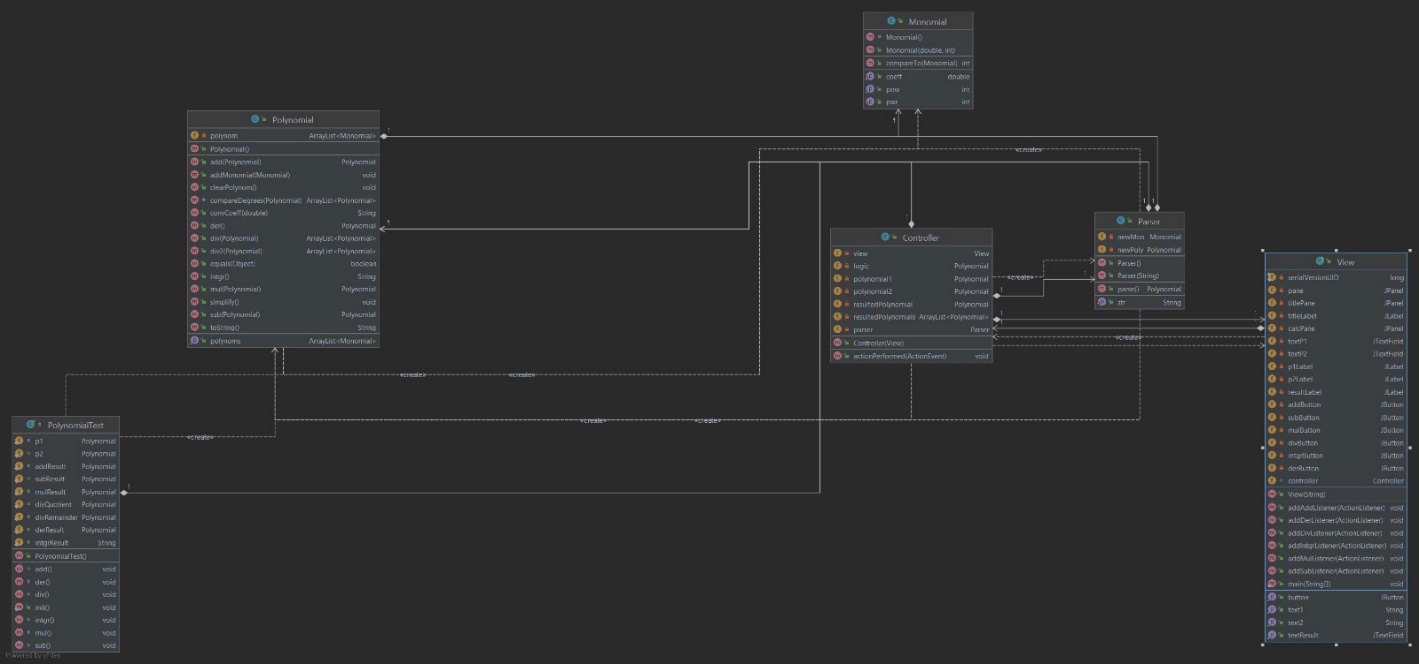
View este pachetul în care se află clasa cu același nume care implementează interfața grafică menită să interacționeze cu utilizatorul.

Controller este pachetul în care este facută legătura dintre interfața grafică din view și logica aplicației din model. Mai exact, va trimite input-ul dat către View în Model ca mai apoi să trimită rezultatul înapoi către View.

Graphical user interface, text, application, chat or text message

Description automatically generated

Fiind un proiect relativ complex legăturile dintre clase și obiecte sunt reprezentate printr-o diagramă UML(Unified Modeling Language). Aceasta servește ca un mod de de a vizualiza proiectul înainte de implementare, în faza de proiectare, sau după ce acesta a fost deja terminat.



1. **Implementare**

În acest capitol voi prezenta pachetele, clasele si metodele din acestea într-o ordine care să aibă sens din punct de vedere al funcționării programului.

Pachetul View:

Clasa View:

Aceasta este clasa care implementează interfața grafică și din care începe execuția programului. Interfața grafică este implementată folosind Java Swing, o librărie integrată din Java.

Clasa conține următoarele variabile, care majoritatea sunt componente swing pe care le vom adăuga ulterior în fereastra pe care o vom crea.

Text

Description automatically generated

-Metoda Main:

Text

Description automatically generated

În interiorul main-ului inițializăm un frame în care să se poată pune elementele grafice care vor fi folosite. Este setat modul de închidere al ferestrei prin metoda setDefaultCloseOperation. Apoi este setată dimensiunea acesteia prin setSize și o setăm ca aceasta să fie vizibilă.

-Metoda Constructor:

Text

Description automatically generated

În această metodă adăugăm componentele grafice în diferite panouri. De exemplu, primul label cu rol de titlu și celelalte label-uri, textbox-uri și butoane care fac parte din calculatorul propriu-zis sunt plasate în două panel-uri diferite pentru a obține aspectul diferit( prin folosirea a diferitor layout-uri). În final aceste două panouri sunt plasate împreună într-un panou principal, panoul pane.

Tot aici avem metodele care adaugă mai multe ActionListener pentru diferitele butoane din interfață.

Pachetul Model:

Clasa Parser:

În această clasă este procesat string-ul care vine ca și intrare de la view. Acesta este parsat, fiecare termen al polinomului fiind transformat într-un monom care este mai apoi adăugat la polinomul respectiv.

În constructor nu facem decât să inițializăm un string care va servi ca și variabila în care vom reține input-ul

-Metoda parse:

Graphical user interface, text, website

Description automatically generated

Ne vom folosi de următorul regex pentru a despărți fiecare termen în trei grupuri: semn, coeficient și grad. În variabila newPoly vom crea noul polinom. Deoarece noi citim două polinoame și deoarece input-ul se poate modifica pe parcursul execuției programului, vom ’’curăța’’ polinomul pentru ca polinomul nou construit să fie cel dorit.

Text

Description automatically generated

În continuare vom construi polinomul pe baza celor trei grupuri extrase în string-uri diferite. Practic vom crea monoame care vor fi adăugate la polinomul final .Variabilele semn, coeff și pow corespund semnului, coeficientului, respectiv gradului. Prima dată verificăm valoarea semnului: plus sau minus. După aceea, în cazul în care există un coeficient, îl vom parsa și îi vom aplica semnul. Apoi, verificam dacă gradul este mai mare de 0, după care îl parsăm. Recalculăm și coeficientul pentru cazul în care acesta este 1 și nu este scris explicit, de exemplu: x^2 sau -x^3. În cele din urmă, dacă coeficientul este diferit de 0 construim noul monom și îl adăugăm la polinomul din care face parte cu ajutorul metodei addMonomial.

Clasa Monomial:

Text

Description automatically generated

Această clasă implementează structura de monom. Are ca atribute coeficientul monomului și gradul acestuia.

Text

Description automatically generated

Avem doi constructori. Unul inițializează atributele pe 0, iar al doilea setează atributele cu valori primite ca parametru.

Text

Description automatically generated

Suprascriu metoda compareTo pentru a putea compara monoamele în funcție de puterea lui x. Această funcție va fi utilă pentru simplificarea expresiilor de tip polinom în care avem mai multe monoame cu aceeași putere.

Clasa Polynomial:

Text

Description automatically generated

Această clasă implementează structura de polinom și metodele aferente acesteia. Ca și atribut la nivel de clasă avem un ArrayList de monoame care reprezinta polinomul în sine, fiecare monom fiind considerat un termen din polinom. Constructorul clasei inițializează un nou obiect de tip ArrayList pentru atributul polynom.

-Metoda add:

Text

Description automatically generated

Această metodă realizează adunarea dintre două polinoame. De exemplu, dacă avem polinomul p1 și p2, ea se va apela în modul p1.add(p2). Deci se va face adunarea dintre p1 identificat prin cuvântul cheie this și p2 care este trimis ca și parametru.

Ca și variabile locale în funcție vom avea un polinom în care salvăm rezultatul și două hashmap-uri care rețin ca și valoare coeficientul unui monom și ca și cheie puterea unui monom, precum și un monom care reprezintă rezultatul adunării dintre două monoame. Primele două for-uri adaugă coeficientul și puterea fiecărui monom a polinoamelor în cele doua hashmap-uri. După ce map1 și map2 sunt populate corespunzător, parcurgem map1 și verificăm dacă există monoame în map2 cu aceeiași putere. Daca există, vom adăuga la rezultat un nou monom care reprezintă suma dintre aceste două monoame și eliminăm monomul respectiv din cel de-al doilea hashmap(nu vrem să-l adaugăm după ce a efectuat suma cu un monom din map1). Dacă nu există, vom adăuga monomul fără a îl schimba. Astfel, ajungem să avem în map2 doar acele monoame care nu au corespondent în map1 din ceea ce privește puterea. Pe acestea le adăugăm fără să le modificăm. În final returnăm polinomul rezultat.

-Metoda sub:

Această metodă realizează scăderea dintre două polinoame. Dacă avem două polinoame p1 și p2, metoda se va apela în modul p1.sub(p2) . Se va face scăderea dintre p1 identificat prin cuvântul cheie this și p2 care este transmis ca și parametru.

Metoda este asemănătoare celei de adunare. Vom avea un polinom care reprezintă rezultatul, două hashmap-uri care rețin ca și valoare coeficientul unui monom și ca și cheie puterea unui monom și un monom care reprezintă rezultatul scăderii dintre două monoame. Primele două for-uri vor popula map1 și map2 corespunzător, astfel încât să avem în primul monoamele care compun primul polinom și în al doilea monoamele care compun al doilea polinom. În următorul for verificăm dacă există monoame din map1 care au corespundent în map2 când vine vorba de putere. Daca există, vom crea un nou monom rezultat care reprezintă scăderea dintre monomul din primul hashmap și cel din al doilea hashmap. Acest monom va fi eliminat din map2. Toate monoamele care nu au corespondent sunt adăugate fără a fi modificate. După acest for, în map2 vor rămâne acele monoame care nu au corespondent în map1. Acestea vor fi adăugate la polinomul rezultat cu semn schimbat deoarece efectuăm operația de scădere. În final returnăm polinomul care rezultă din scădere.

Text

Description automatically generated

-Metoda mul:

Text

Description automatically generated

Această metodă efectuează înmulțirea a două polinoame. Dacă avem două polinoame p1 și p2, metoda se va apela în modul p1.mul(p2) . Se va face înmulțirea dintre p1 identificat prin cuvântul cheie this și p2 care este transmis ca și parametru.

Ca și variabile avem polinomul result care reprezintă rezultatul înmulțtirii și monomul resultedMonomial care reprezintă rezultatul înmulțirii a două monoame. Avem două for-uri pentru că vrem ca fiecare monom din primul polinom să fie înmulțit cu fiecare monom din al doilea polinom. În interiorul for-ului creăm un nou monom prin înmulțirea coeficienților și adunarea puterilor. În final, returnăm polinomul rezultat.

-Metoda der:

Text

Description automatically generated

Această metodă efectuează derivarea primului polinom primit ca input. Are ca variabilă locală un polinom result, care reprezintă rezultatul derivării.

Avem un for, cu care parcurgem fiecare monom al acestui polinom. Pentru fiecare dintre aceste monoame, dacă puterea lui x este mai mare de 0, vom adăuga la polinomul rezultat un nou monom, care are coeficientul egal cu vechiul coeficient înmulțit cu puterea lui x și puterea egala cu vechea putere minus 1. În final returnăm polinomul rezultat în urma derivării.

-Metoda intgr:

Text

Description automatically generated

Această metodă efectuează integrarea primului polinom primit ca input. Are ca variabilă locală un string result, care va fi polinomul rezultat în urma aceste operații. Se va returna un string deoarece doresc să afișez rezultatul sub forma: .

-Metoda div

Text

Description automatically generated

Această metodă efecutează împărțirea dintre două polinoame. Indefernt de ordinea în care sunt introduse polinoamele, se va efectua împărțirea polinomului cu gradul mai mare la cel cu gradul mai mic. Rezultatul acestei operații este salvat într-in ArrayList întrucât vom avea un cât și un rest. Ca și variabile vom folosi un ArrayList de tip Polynomial result, un polyTemp, quotient, s, p și q toate utilizate în calcule cu scopul de a facilita operația de împărțire. Pe lângă acestea mai avem variabilele powQ, powP, powTemp de tip int care reprezintă puterile monoamelor de împărțit, și coeffQ, coeffP și coeffP care reprezintă coeficienții monoamelor de împărțit(coeffTemp este folosit pentru a calcula coeficientul câtului).

Algoritmul de împărțire se bazează pe cel dat în prezentarea temei. Primul pas al împarțirii este realizat înafara buclei do while. Preluăm puterile lui P și Q(cum am menționat mai sus considerăm că unul din polinoame are gradul mai mare, în cazul acestei metode polinomul P va avea întodeauna gradul >= cu cel al polinomului Q). Se creează polinomul polyTemp. În interiorul buclei vom scădea puterile monoamelor(împărțirea puterilor) după care vom împărții și coeficienții acestora. Aceste valori vor fi folosite pentru a crea unul dintre termenii câtului. Pentru a afla unul dintre termenii restului împărțirii, acest monom este adăugat în polinomul s. Polinomul s este mai apoi înmuțit cu câtul. Restul este dat de diferența dintre polyTemp și s. Dacă restul la un moment dat nu există, se termină execuția buclei. La finalul buclei sunt actualizate variabilele powP, coeffP și se curăță variabila s.

La finalul buclei, se verifică dacă gradul polinomului temporar este mai mare decât cel al împărțitorului. Pe urmă adăugăm câtul și restul obținut în listă și o returnăm.

-Metoda simplify:

Această metodă se asigură că avem un singur monom de o anumită putere într-un polinom. Conține două ArrayList-uri care reprezintă monoamele pe care le vom șterge din polinom. Avem nevoie de ele întrucât nu putem șterge elemente când lucrăm cu un foreach. În interiorul celor două for-uri se verifică dacă două monoame au aceeiași putere, iar dacă au, se adaugă suma monoamelor la polinom, iar monoamele care au luat parte la sumă sunt eliminate.

-Metoda compareDegrees:

Această metodă este folosită în cadrul împărțirii. Ea returnează un ArrayList de Polynomial, pe prima poziție aflându-se polinomul cu gradul mai mare, iar pe a doua polinomul cu gradul mai mic.

Clasa controller:

Text

Description automatically generated

Această clasa realizează legătura dintre logica aplicației și interfața grafică a acesteia. Drept urmare, implementează interfața de listener pentru a putea da comanda potrivită pentru fiecare caz de utilizare. Conține un obiect de tip view, adică instanța interfeței grafice.

Tot aici vom stabili operația care urmează să aibă loc în funcție de butonul apăsat. După ce vom obține rezultatul, acesta va fi transmis și afișat în View.

1. **Rezultate**

Pentru testare am folosit framework-ul JUnit.

Text

Description automatically generated

Clasa pentru realizarea testelor se află în pachetu ei separat. Aici avem mai mult polinoame care vor reprezenta două input-uri(p1, p2) și mai multe rezultate a operațiilor ce se pot efectua(adunare, scădere, înmulțire, etc.). În metoda init din cadrul adnotării BeforeAll vom inițializa atât polinomii de intrare cât și cei rezultat cu valorile corespunzătoare.

Text

Description automatically generated

Pentru realizarea testelor propriu-zise, m-am folosit de assertEquals. Pentru că oricum rezultatul unei operații pe polinoame va ajunge să fie un string, voi compara string-ul valorii expected cu string-ul valorii reale. Conversia se face cu ajutorul metodei toString, suprascrisă în clasa Polynomial.

1. **Concluzii**

În concluzie, după finalizarea acestei teme am obținut un calculator de polinoame funcțional, care poate efectua câteva din operațiile de bază. Consider că adevăratul câștig obținut este din procesul de modelare al programului din această temă. Am reușit să implementăm un proiect folosind un design pattern și anume MVC, să ținem cont de bunele practici ale programării în Java și să testăm funcționalitatea acestuia.

1. **Bibliografie**

https://stackoverflow.com/questions/28859919/java-regex-separate-degree-coeff-of-polynomial