DOCUMENTATIE

TEMA 1

NUME STUDENT: MURESAN ALEXANDRU DORIAN

GRUPA: 30229

# 

# CUPRINS

[1. Obiectivul temei 3](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297886)

[3. Proiectare 5](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297887)

[4. Implementare 7](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297888)

[5. Rezultate 11](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297889)

[6. Concluzii 12](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297890)

[7. Bibliografie 13](file:///D:\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297891)

1.Obiectivul temei:

1. Obiectivul principal il reprezinta realizarea unui calculator de polinoame interactiv, care poate efectua operatii matematice, precum adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea.
2. Obiectivele secundare cuprinde :

* Implementarea operatiilor de adunare, scadere, inmultire, impartire, derivare si integrare
* Realizarea unei interfete pentru utilizator, unde isi poate introduce propriile polinoame si care ii va afisa rezultatele pentru operatiile dorite
* Validarea datelor de intrare astfel incat sa se respecte formatul unui polinom, iar aplicatia sa functioneze cum trebuie
* Documentatie ce ajuta utilizatorul sa inteleaga cum se utilizeaza aplicatia

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

Prin analiza problemei ne referim la procesul de identificare, definire și înțelegere a unei probleme pentru a găsi o soluție sau un set de soluții. Avem clar un avantaj pentru că folosim un limbaj orientat pe obiecte. Acest lucru ne permite să lucrăm la un nivel conceptual superior, fără a fi constrânși de munca tehnică.

Această strategie este cunoscută și ca proiectare bottom-up. Are un mare avantaj, deoarece putem găsi cu ușurință structuri care sunt conectate într-un fel cu lumea reală (obiecte). Clar, acest concept nu are doar avantaje, ci vine și cu un nivel ridicat de complexitate.

Aplicația va fi utilizată de diferite tipuri de utilizatori, ceea ce ne spune că punctul de pornire ar trebui să fie implementarea unei interfețe care este ușor de utilizat și convenabilă.

Știm că Calculatorul Polinomial trebuie să implementeze operații algebrice de bază, ceea ce determină că sunt necesare cel puțin două câmpuri diferite pentru fiecare polinom. Cel care utilizează calculatorul poate introduce șiruri simple de la tastatură.

Utilizatorul trebuie să folosească primele două etichete pentru a introduce polinoamele în forma: (coeficient x^ putere). Este foarte important ca monoamele să nu fie separate de semnul lor printr-un spațiu. Un exemplu poate fi: 3x^2+2x^1-2.Graphical user interface, application

Description automatically generated

De asemenea, utilizatorul trebuie să știe că programul are o problemă atunci când coeficientul este egal cu 1 și este scris în forma: "x" în loc de "1x".

Ordinea în care sunt introduse polinoamele nu contează, deoarece aplicația va rearanja polinoamele în ordine descrescătoare după putere.

3) Proiectare:



1. Diagrama UML



b) Structuri de date

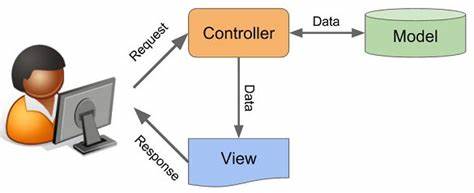
Structurile de date folosite de mine pentru crearea acestei aplicatii sunt tipuri privitive, adica de tip int si de tip double, dar si de tip HashMap pentru a stoca atat tipuri primitive, cat si obiecte, dar eu am ales sa stochez doar tipuri primitive.

Folosind HashMap am reusit sa accesez rapid si eficient datele, precum cheile ce semnificau puterea si valoarea ce semnifica coeficientul. Accesarea in functie de cheie este una foarte rapida. Am putut sa adaug si sa sterg cu usurinta date. Le am putut adauga folosind metoda put() si sa le sterg folosind metoda remove().



c) Interfetele definite

Pentru acest proiect eu am ales sa folosesc Model-View-Controller:



Model-View-Controller, prescurtat si MVC este un pattern design ce interactioneaza efficient in legatura dintre interfata utilizatorului si logica din spate. Este foarte util deoarece reutilizeaza codul obiectului ajuta la reducerea timpului necesar in dezvoltarea unui aplicatii dorite.

Arhitectura MVC presupune împărțirea unei aplicații vizuale în trei părți distincte:

* Un model care reprezintă datele interne ale aplicației si care modeleaza problema care este in curs de solutionare cu ajutorul programului. Acesta este independent, dar poate sa ofere servicii, atat controlerului cat si vizualizarii(view)
* Vizualizare (view) - reprezentarea vizuală a acestor date si locul unde utilizatorul poate vedea ce face programul. Aceasta arata ce face modelul
* Un controlor (controller) care preia intrarea de la utilizator și o transformă în schimbări în model si care decide ce urmeaza sa faca modelul si in ce fel sa fie facut. Acesta depinde de model

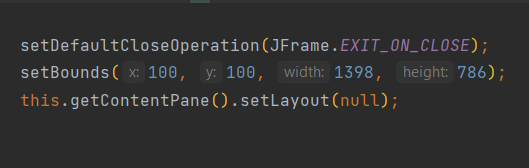
Uneori Controlorul şi Vizualizarea sunt combinate, mai ales în programe de mici dimensiuni. Combinarea Controlorului şi a Vizualizării este potrivită dacă cele două sunt foarte interdependente, iar modelul trebuie sa ramana complet independenta.

În general, arhitectura MVC este considerată un pattern de design robust și scalabil care ajută la crearea de aplicații mai bine organizate, mai ușor de extins și de întreținut.

4)Implementarea:

4.1 Clasa Calculator View

Furnizează un JFrame care construiește întreaga interfață a aplicației.



Toate butoanele, zonele de text, etichetele etc.... sunt private și tot codul care construiește interfața este implementat în constructorul clasei Calculator View. Constructorul este apoi apelat în clasa principală pentru a oferi vizibilitate și funcționalitate. De asemenea, am creat setteri și getteri pentru fiecare variabilă pentru a accesa ușor în alte clase atunci când trebuie să oferim ieșiri sau să obținem informații din intrări.

Text

Description automatically generated

Pentru a face interfața funcțională, avem nevoie să implementăm ascultători pentru butoane:

Text

Description automatically generated

4.2 Clasa Calculator Controller

Este responsabilă de funcționalitatea butoanelor. Când se întâmplă un eveniment în aplicație (utilizatorul interacționează), această clasă va fi responsabilă de a face acțiunea așteptată. Constructorul acestei acțiuni este conectat la clasa Calculator View.

Text

Description automatically generated

De asemenea, furnizează metoda de citire -> preia intrarea introdusă în aplicație de către utilizator și o împarte în șiruri care reprezintă monoamele.

Text

Description automatically generated

4.3 Clasa Monom

Această clasă este utilizată doar pentru metoda de transformare care preia intrarea citită din Calculator Controller, convertește coeficientul în Double și puterea în Integer și apoi pune rezultatele convertite într-un Hash Map furnizat de clasa Polynomials.

4.4 Clasa Polinoame

Această clasă este reprezentarea unui polinom, reprezentat într-un Hash Map, cu cheia de tip Integer care corespunde puterii și valoarea de tip Double care este coeficientul.

În primul rând, furnizează metoda: addinHashMap care ia coeficienții și puterile ca parametri și îi pune în HashMap. Dacă polinomul are mai mulți coeficienți ai aceleiași puteri, metoda îi adaugă automat.

Text

Description automatically generated

În al doilea rând, furnizează metoda care afișează ieșirea. Parametrul numit tip este necesar să fie specificat deoarece va determina în care zonă de text va fi afișată ieșirea. Fiecare operație are un câmp de ieșire diferit. Metoda folosește, de asemenea, transformToString, care convertește toate elementele din Hash Map-uri în șiruri și le concatenează cu semnele care corespund coeficienților lor.

Text

Description automatically generated

4.5 Clasa Operație

Această clasă este utilizată în mod specific pentru a stoca metodele operațiilor algebrice. În acest caz, metodele utilizate sunt: adunare, scădere, înmulțire, împărțire, derivare și integrare.

Vom începe cu metoda de adunare:

Text

Description automatically generated

În primul rând, am utilizat o instrucțiune "for each" pentru a itera prin toate elementele din Hash Map-ul primului polinom. Dacă, în timpul acestei iterații, găsesc că puterea unei intrări este egală cu puterea altei intrări din al doilea polinom, adun cele două valori cheie și salvez rezultatul într-o variabilă numită "resultPolynomial". Dacă nu există o putere potrivită, celelalte elemente sunt plasate în variabila fără a fi adăugate. De asemenea, am creat o iterație separată pentru al doilea polinom și am pus toate intrările care nu au fost adăugate în prima iterație.

Metoda de scădere este aproape identică cu cea de adunare. Singura diferență constă în faptul că în loc de plus, utilizăm minus atunci când sunt găsite două elemente cu aceeași putere.

Text

Description automatically generated

Metoda de înmulțire este foarte simplă. Iterăm primul polinom și înmulțim intrarea curentă cu toate intrările din al doilea polinom. Răspunsul este apoi salvat într-o variabilă numită "resultPolynomial".

Text

Description automatically generated

Metoda de integrare este utilizată pentru fiecare polinom în mod individual. Iterăm polinomul, iar dacă puterea este mai mare sau egală cu 0, se efectuează integrarea intrării. Când integrarea este finalizată, puterea crește cu 1, iar coeficientul este împărțit cu puterea + 1.

Text

Description automatically generated

Metoda de derivare este destul de similară cu integrarea. Diferența constă în faptul că, atunci când se efectuează derivarea, puterea este redusă cu 1, iar coeficientul este înmulțit cu puterea.

Text

Description automatically generated

Ultima din listă și cea mai complexă este metoda de împărțire. Implementarea folosită pentru acest algoritm este Diviziunea Lungă. Codul efectuează următorii pași: găsește termenul cu cea mai mare putere a primului polinom (dividend) și al celui de-al doilea polinom (divizor), calculează coeficientul și puterea următoarei intrări din Hash Map-ul catului, adaugă următoarea intrare în cat, înmulțește divizorul cu următoarea intrare din cat, scade rezultatul din pasul anterior din primul polinom pentru a obține un nou rest și, în cele din urmă, elimină orice termeni cu coeficienți zero din Hash Map-ul restului.

Text

Description automatically generated

5)Rezultate:

Testarea a fost realizată prin intermediul interfeței implementate în clasa principală și prin intermediul testării Junit.

Graphical user interface

Description automatically generated

Acesta este un exemplu de toate ieșirile care pot fi furnizate prin aceste intrări.

Testarea Junit a fost utilizată doar pentru cazurile de adunare și scădere, care sunt testate într-un mod aproape identic. Declarăm o hartă așteptată și creăm două variabile de tip Polynomial care stochează câteva valori. Utilizarea metodei assert Equals și metoda de adunare furnizată de clasa Operation vor confirma dacă testul a fost reușit.

Text

Description automatically generated

Este important să se observe că unele intrări nu pot fi cele mai bune pentru aplicații. Unele criterii pe care le-am găsit utile pentru buna funcționare a aplicației prin testare sunt:

"3x ^ 2 - 2x", nu "3x ^ 2 - 2x". Spațiul dintre elemente poate afecta programul.

utilizați "1x" în cazul coeficientului 1, nu doar "x".

6) Concluzii:

Partea care m-a provocat cel mai mult în procesul de creare a aplicației a fost conversia de la intrarea String -> împărțirea String-ului prin semnul specific și apoi memorizarea rezultatelor convertite într-un Hash Map. În acest fel, am învățat câteva trucuri în utilizarea metodei Split. La fel se întâmplă și cu conversia înapoi în String (pentru afișare), unde am descoperit ce este StringBuilder și cum să folosesc DecimalFormat. De asemenea, am învățat algoritmul de împărțire lungă și cum să-l implementez în IntelliJ.

În dezvoltările viitoare aș putea îmbunătăți interfața utilizatorului a aplicației și sistemul de intrare. De asemenea, un buton de comutare pentru a modifica ordinea de intrare a polinoamelor sună interesant.

7) Bibliografie:

* <https://stackoverflow.com/>
* <https://www.geeksforgeeks.org/>
* <https://www.youtube.com/watch?v=H62Jfv1DJlU>
* https://www.youtube.com/watch?v=\_Ph9WsCm4Uk&t=1710s