

**UNIVERSITATEA DE AUTOMATICA SI CALCULATOARE**

**FUNDAMENTAL PROGRAMMING TECHNIQUES**

**ASSIGNMENT 1**

**DOCUMENTATIE TEMA 1**

**CALCULATOR DE POLINOAME**

**Nume si prenume: Stecko Daiana**

**Grupa 302210 | An 2 semestrul 2**

**Profesor laborator: Mitrea Dan**

**CUPRINS**

**1.Problema si solutia problemei.**

**2.Obiectivul temei.**

**3.Analiza problemei, modelare, scenarii, cazuri de utilazare.**

**4.Proiectare.**

**5.Implementare.**

**6.Rezultate.**

**7.Concluzii.**

**8.Bibliografie.**

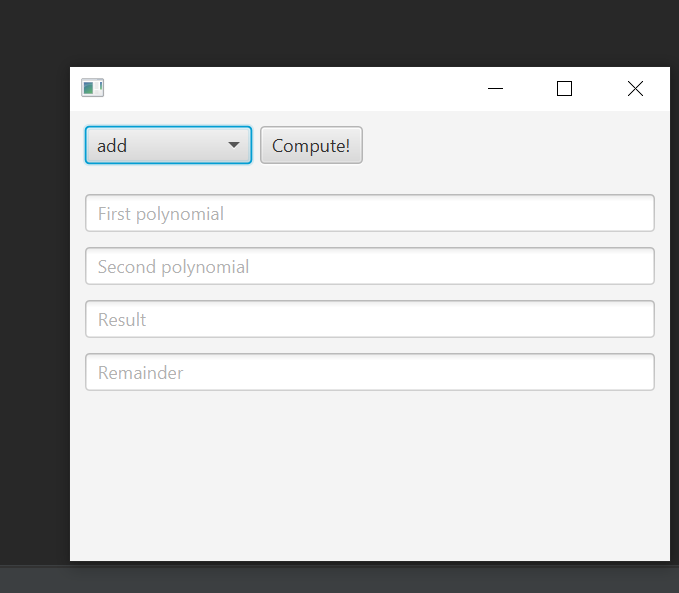
**1.Problema si solutia problemei:**

Problema: In cazul primei teme am fost nevoie sa rezolvam diverse operatii pentru polinomme, fiind destul de dificile pana am gasit un algortim optim si necesita de asemenea mult timp.

Solutia problemei: Pentru a realiza acest lucru ne folosim de un “Calculator de polinoame” care face diverse operatii cu acestea.

Se poate calcula cu usurinta o operatie : (Ex: 4x^4+x^3+x+6x^0) fara a avea probleme la calcule mai mari, si fara a fi necesar dupa sa reluam exercitiului in caz de greseli sau pentru verificari.

Am facut o interfata destul de simpla, care nu a necesitat mult timp si este usor de inteles, pentru calculator de polinoame:



1. Obiective:

Obiectivul acestui laborator este de a implementa si de a realiza un sistem pentru a putea calcula diferite operatii cu polinoame.

Calculatorul de polinoame poate efectua diferite operatii pe polinoame: adunare, scadere, inmultire, derivare si integrare. Datele sunt introduse de utilizator, iar rezultatul va fi generat, dupa efectuarea calculelor, pe ecran in casuta atribuita acestuia.

Pentru interfata, am folosit tehnica “Model-View-Controller”, pentru a crea un aspect cat mai placut si usor de utilizat pentru orcine ar vrea sa isi rezolve corect si in scurt timp exercitiile cu polinoame.

1. Studiul problemei, modelare, scenarii, cazuri de utilizare

Programul „Calculator polinoame” este un program de consolă pur, care este folosit pentru a adăuga, scădea, înmulți și împărți polinoame până la n grade.

Acesta permite utilizatorului sa incarce polinoame de la tastatura, urmand ca sa se rezolve diverse operatii matematice dorite de utilizator.

Ce reprezinta de fapt un polinom ?

In matematica, un polinom este definit ca fiiind o expresie alcatuita din mai multe variabile si constante, folosind doar operatii de adunare, scadere, inmultire, impartire si ridicare la putere.

Ce reprezinta de fapt un monom ?

Monoamele sunt termenii din care este alcatuit un polinom si sunt alcatuite dintr-o constanta si un exponent. Fiecare variabila poate avea sau nu un exponent intreg pozitiv, acesta da gradul variabilei din monom .

Operatiile definite in acest program sunt: adunare, scadere, inmultire, derivare si integrare. Pentru a aduna, scadea, sau inmulti avem nevoie de doua polinoame. Noi va trebui sa introducem un polinom in prima casuta , iar in cea de a doua casuta, alt polinom. Dupa care vom apasa butonul de „compute”, care va realiza operatia selectata de catre noi. In cea de a treia casuta apare rezultatul pentru operatia facuta, practic il afisam.

Am facut o casuta unde am toate tipurile de operatii necesare acestui proiect: add, substract, multiply, divide, differentiale, integrate. Utilizatorul introduce intai polinoamele in cele doua casute „first polynomial” pentru primul polinom, si „second polynomial” pentru cel de al doilea, dupa care selecteaza opratia pe care vrea sa o aplice acestor polinoame.

Pentru implementarea „Calculatorului polinomial”, am folosit sase clase pentru implementare.

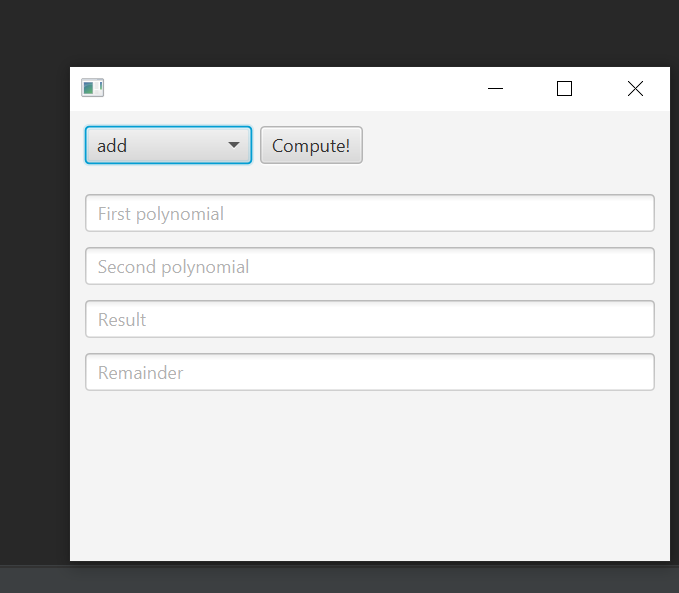
Programul cuprinde clasele: Main, GUI, Controller, PolynomialParser , Monomial si Polynomial.

In clasa Monomial definim conceptul de monom si functiile de adunare, scadere, inmultire si derivare, urmand ca in clasa Polynomial, sa ne generalizam operatiile pentru un polinom. Clasa PolynomialParser ne ajuta sa separam listele si sa extragem doar informatia pe care am vrea sa o folosim. In clasele Main, GUI si Controller mi-am definit partea de interfata necesara proiectului.

De remarcat ca am folosit pachetul javafx pentru implementarea interfetei.

1. Proiectare

O sa incep prin a descrie partea de interfata. Este un program foarte usor de utilizat si in acelasi timp de citit, interfata lui fiind una „User friendly”. Se introduce polinomul de la tastaura, si dupa se alege operatia dorita.

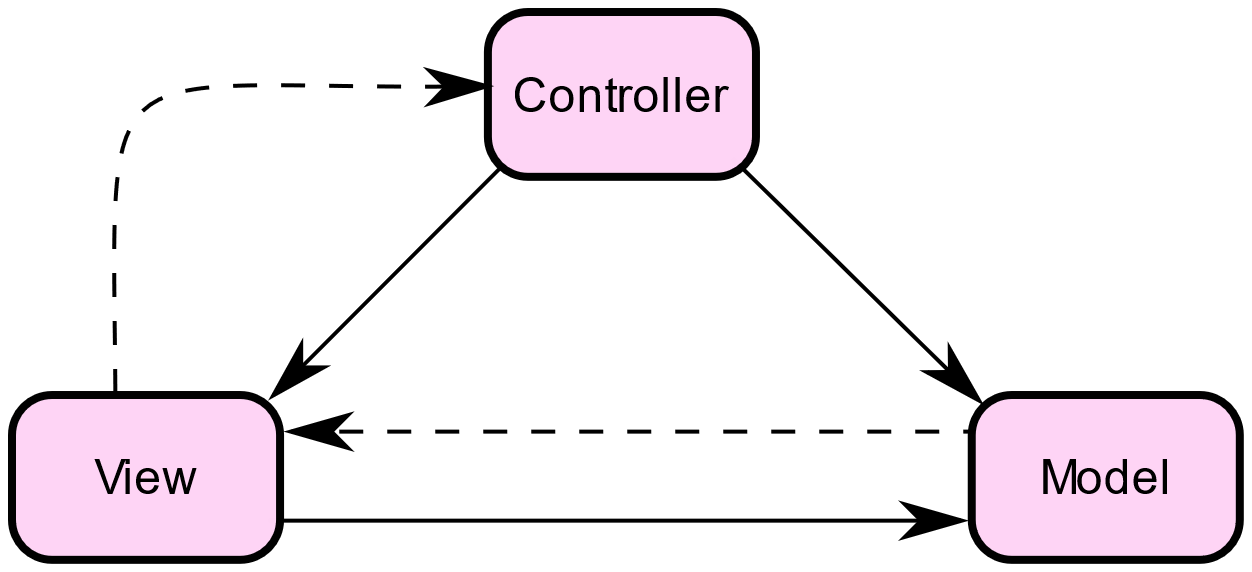


Am folosit MVC care reprezinta un model arhitectural utilizat de catre ingineria software. Succesul modelului se datorează izolării logicii de business față de considerentele interfeței cu utilizatorul si va rezulta o aplicație unde aspectul vizual sau/și nivelele inferioare ale regulilor de business sunt mult mai ușor de modificat, fără a afecta celelalte nivele.

Clasa GUI este folosita pentru reprezentarea grafică, sau chiar mai bine zis, pentru exprimarea formei a datelor finale: interfața grafică este de fapt cea ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la nivelul controlorului.

Cu ajutorul Controlorului, modelului sau a parții de vizualizare putem manipula datele. Depinde doar de noi cum manipulăm și interpretăm aceste "date". Acum realizam că unicele date ale unei adrese web de tip static sunt: obținerea unui fișier de pe disc (adica hard disk) sau din Internet, etc. și de asemenea interpretat sau nu, serverul va răspunde.

Modelul, precum controlorul și vizualizarea (interfața grafică) manipulează toate datele ce se relaționeză cu el. Și numai partea de Gui(view) poate demonstra această informație. În acest fel am demonstrat ierarhia programului nostru: Controlor-Model-View.



Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte si pentru business Process Design etc.

Diagrama UML pentru Controller:

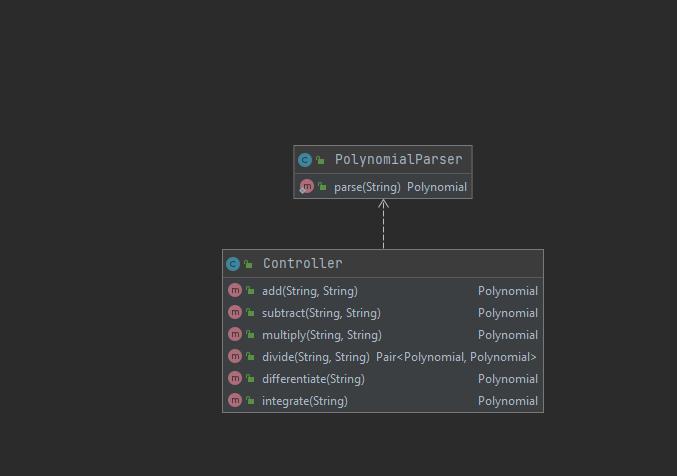


Diagrama UML pentru Model:

O imagine care conține text

Descriere generată automat

Diagrama UML pentru View:

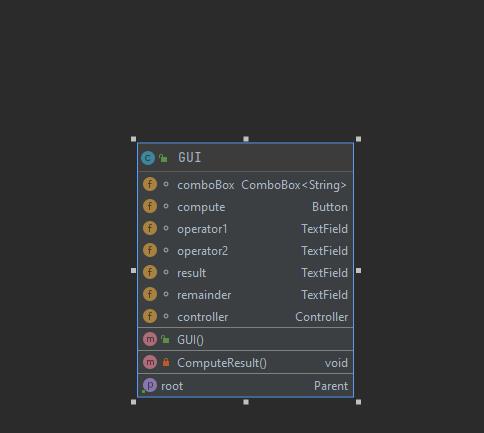
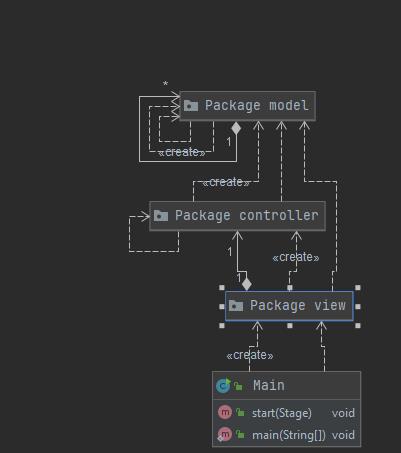


Diagrama UML pentru tot proietul:



1. **Implementare:**

**Clasa GUI:**

In clasa Gui am initializat combobox-urile, dupa care le-am pus in pagina. Mi-am setat textfield-urile, si toate celelalte butoane necesare, etc. In compute result, in functie de operatie este selectata in combobox-uri ce executa operatia din Controller.

**Clasa Controller:**

In clasa controller se defineste add, substract, multiply, divide, differentiate, integrate. Ceea ce face de fapt aceasta clasa este sa ia doua texte care reprezinta un polinom, le parseaza si va returna rezultatul aditie dintre ce doi. Rezultatul aditiei dintre cei doi vine din polynomial.

**Clasa Polynomial:**

In clasa polynomial vom lua de fapt o lista de monomiali. Aici avem un constructor care construieste un ponomial, fiind un degredat. Dupa care vom avem un constructor care defineste printr-o lista de monomiali. De asemenea avem u constructor care defineste printr-o lista de indici. Functia add ia cel mai mare dintre polinoame, il va copia in rezultat si dupa merge prin cel mai mic dintre polinoame si-l va adauga si pe coeficientul lui tot la rezultat. Tot in aceasta clasa avem un toString care returneaza tot polinomul sub forma de string. De asemenea avem un substract care pur si simplu apeleaza functia de add pe un operator 2 inversat, ce face asta de fapt, va inversa monomial al polinomului operandului 2.

In multiply, practic se duce si inmulteste fiecare polinom si va selecta la putearea potrivita rezultatele acelei inmultiri, deci le inmultim fiecare cu fiecare.

Nu am imi functioneaza implementarea la divide, dar am incercat totusi sa o fac. Practic, ar trebui sa faca: daca pol1 mai mic decat pol2, atunci returneaza 0 si op1, daca pol2 este egal cu 0, atunci avem division by zero, dupa cat timp op1 mai mare decat op2, adica gradul mia mare, atunci se iau cei mai mari indici si se divid si rezultatul se adauga acelei diviziuni, se creaza un polinom nou din el, se adauga monomul acelui polinom si se scade din operandul nostru intai inmultirea monomului rezultat din diviziuni si al celui de al doilea polinom. La final se sorteaza, adica vom sorta puterea sa mearga in locul ei. HelpSize ne va da inapoi size-ul ei. Differentiale ne va diferentia prin diferentierea fiecarui monom al polinomului. Acelasi lucru va face si integrate. La final mai avem niste setere si gettere.

**Clasa Monomial:**

Aceasta clasa are un coeficient de tip number, adica poate sa fie si float si int, dupa cum dorim noi, avem un power de tip int. Avem un toString care verifica anumite criterii si ttransforma intr-o reprezentare string. La add, pur si simplu adaugam a si pastram puterea, iar la invert cream un nou polinom doar ca cu „-„ coeficientul celui vechi. Multiply, aici inmultim cei doi coeficientu si adunam puterile. La divide impartim ce doi coeficienti si scadem puterile. La differentiale vom imparti cu puterea coeficientul si va puterea scade. La integrate, coeficientul este impartit cu puterea + 1, si puterea creste. La final mai avem niste setere si gettere pentru power si coefficient.

**Clasa PolinomialParser:**

In clasa PolinomialParser, ne vom lua un regex pattern si ne vom identifica, pattern-ul indentifica de fapt coeficientii, puterea si polinomul. Vom compila pattern-ul, ne vom lua matchr-ul pattern si ne vom stabili un nou polinom. Dupa care ne-am creat doua int-uri si cat timp vom gasi monomi ne vom lua coeficintul si puterea. Daca coeficientul si puterea sunt egale cu Null si va exista x, atunci inseamna ca am match-uit ceva gol. Daca coeficientul si puterea sunt egale cu Null si va exista x, atunci inseamna ca am match-uit ceva , atunci atat coeficntulu cat si puterea sunt 1. In cazul in care n este Null si p este diferit de Null, atunci coeficintul va fi 1 si vom pastra puterea. Daca n este diferit de Null, de asemenea si p este diferit de Null, atunci vom pastra coeficintul si daca am match-uit si un x, atunci puterea va fi egala cu 1, daca nu atunci puterea va fi egala cu 0. Altfel, pur si simplu le vom extrage pe amandoua si la final le construim monomul nostru. In final, vom face o sortare, pentru a putea sorta monomii la locurile lor.

**Clasa Main:**

In clasa Main avem doar gui-ul creat, alaturi de dimensiunile sale.

1. **Rezultate:**

Nu am facut toate cerintele proiectului, mai era necesar pentru a verifica corectitudiea operatiilor, sa mai implementez o clasa de test numita, de exemplu TestPolynomial.

Iar pentru fiecare operatie, sa creez o metoda de testare unde sa ofer un rezultatul final calculat de mine.

Nu am reusit sa fac aceasta parte si nu am reusit nici sa implementez implementez impartirea corect, am facut totusi ceva, am incerccat o implementare, dar nu functioneaza bine, imi da un rezultat in bucla infinita si de asemenea un rezultat gresit.

1. **Concluzii:**

In cadrul primului meu proiect, o concluzie importanta ar fi ca: am realizat inca o data rolul si importanta informaticii in orice domeniu.

De asemenea, acest proiect m-a ajutat sa imi dezvolt mult mai bine cunostiintele in tot ceea ce inseamna limbajul Java, crearea unui program cu ajutorul unei interfete grafice.

Acest proiect a necesitat o documentare foarte amanuntita si lunga deoarece a fost mai complex ca celelalte teme din punctul meu de vedere si a fost ceva cu totul nou pentru mine si a trebuit sa caut si pe surse externe de informatie despre acest concept. Intr-un final am reusit sa inteleg oarecum aceasta idee si am reusit o implementez in proiect meu. Ca si dezvoltari ulterioare, consider ca s-ar putea implementa acest algoritm in viata reala, pentru gestionarea rapida a calculelor, deoarece este foarte usor sa gresim, ele fiind destul de complexe si de multe ori avand coeficientii destul de mari. Este foarte usor sa gresim si exercitiul nu va mai iesi cum ne dorim, de aceai acest calculator de polinoame este util si usor de utilizat de oricine, avand comenzi simple.

**7.Bibliografie:**

Am folosit wikipedia pentru diferite definitii si informatii pentru interfata acestui proiect.

Deasemenea am mai folosit diverse site-uri de programare (geeksforgeeks sau W3schools) pentru a incerca sa implementez cat mai bine programul meu.

Am folosit javaFX din inteliji, pentru aceasta m-am documentat de pe diferite site-uri, cum ar fi: [https://cs108.epfl.ch/archive/17/c/JAFX/JAFX-notes.html,https://www.jetbrains.com/help/idea/javafx.html, unde](https://cs108.epfl.ch/archive/17/c/JAFX/JAFX-notes.html,https://www.jetbrains.com/help/idea/javafx.html,%20unde) sunt explicate lucruri de baza utile in crearea proiectului si a interfetei. Si desemenea m-am folosit de cateva tutoriale vizualizate pe youtube unde explica modul in care putem crea o interfata mai complexa.

De asemnea m-am inspirat pentru definirea polinomului si a monomului de pe diverse site-uri.

Pentru a avea siguranta ca am realizat corect operatiile cu polinoame m-am ajutat de:

<https://pdfslide.tips/documents/operatii-cu-polinoame.html>