**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

***Tehnici de programare fundamentale***

Documentatie Tema 1

Calculator de Polinoame

Student : Matioc Bogdan Darius

Profesor: Sera David

Cuprins

[1. Obiectivul temei 3](#_Toc67008533)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc67008534)

[2.1. Analiza problemei 3](#_Toc67008535)

[2.2. Modelare 3](#_Toc67008536)

[2.3. Scenarii și cazuri de utilizare 4](#_Toc67008537)

[3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator) 4](#_Toc67008538)

[Decizii de Proiectare: 4](#_Toc67008539)

[Structurarerea pe pachete 5](#_Toc67008540)

[Pachetul Model 5](#_Toc67008541)

[Pachetul Controller 6](#_Toc67008542)

[Pachetul View 6](#_Toc67008543)

[Diagrama UML 7](#_Toc67008544)

[4. Implementare 8](#_Toc67008545)

[ Clasa OperationModel 8](#_Toc67008546)

[ eliminatesRedundantTerms 8](#_Toc67008547)

[ groupMonomialsWithSamePower 8](#_Toc67008548)

[ addsPolynomials 8](#_Toc67008549)

[ decreasesPolynomials 8](#_Toc67008550)

[ multipliesPolynomials 8](#_Toc67008551)

[ dividesPolynomials 9](#_Toc67008552)

[ derivesPolynomial 9](#_Toc67008553)

[ integratesPolynomial 9](#_Toc67008554)

[5. Rezultate 10](#_Toc67008555)

[6. Concluzii 11](#_Toc67008556)

# Obiectivul temei

Obiectivul principal al acestei teme contă în realizarea unui calculator de polinoame, de o singura variabila (x) cu coeficienti **Intregi**, dar mai presus de asta, odată cu realizarea calculatorului în cauză, se dorește a se ilustra cunoștințele de programare orientată pe obiect. Obiectivele secundare care fac posibilă realizarea obiectivului principal abia menționat sunt următoarele:

* Definirea claselor corespunzătoarea
* Implementarea operațiilor pe polinoame
* Realizarea interfeței grafice
* Preluarea polinomului introdus de utilizator
* Testarea și verificarea corectitudinii implementării realizate

Operatiile pe care le face aplicatia sunt:

* Citirea unui polinom de la tastatura sub forma de String
* Adunarea a doua polinoame
* Scaderea a doua polinoame
* Inmultirea a doua plinoame
* Impartirea a doua polinoame
* Derivarea unui polinom
* Integrarea unui polinom
* Reducere termini asemenea
* Reducerea termenilor redundanti
* Sortare dupa coeficienti

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## 2.1. Analiza problemei

În primul rând, o să începem prin a specifica ce este acela un polinom. Așadar, un polinom este o expresie construită dintr-una sau mai multe variabile și constante. În cazul calculatorului de polinoame realizat la această temă, polinoamele folosite sunt de o singură variabilă, mai precis x. Polinoamele sunt de forma următoare: .

Polinoamele sunt construite din termeni numiți monoame, care sunt alcătuite dintr-o constantă și una sau mai multe variabile. În ceea ce privește calculatorul realizat la această temă, monoamele sunt de forma: .

## 2.2. Modelare

Din punct de vedere al modelării, se pot distinge două părți principale. Prima parte este reprezentată de interfața grafică. Utilizatorul interacționează în mod direct cu aceasta. Interfața este ușor de folosit și sugestivă, din punct de vedere al funcționalității pe care o are. Așadar, utilizatorul poate să efectueze prin intermediul acesteia operații pe polinoame precum: adunare, scădere, înmulțire, împărțire, derivare și integrare. Rezultatul o să fie afișat imediat corespunzător. Cea de-a doua parte este reprezentată de partea de cod aflată în spatele interfeței, care face posibilă întreaga funcționalitate a programului și înglobează algoritmii necesari pentru toate operațiile.

## Scenarii și cazuri de utilizare

Descriere caz de utilizare:

Utilizator

Aplicatie: Calculator de polinoame

* Utilizatorul introduce două polinoame.
* Utilizatorul alege operația care dorește să se efectueze.
* Calculatorul afișează rezultatul corespunzător.

Odată ajunși la pasul 4, utilizatorul are în continuare două posibilități, prima dintre ele presupune de a apăsa butonul de resetare, astfel revenind la pasul 1 prezentat anterior, fie să aleagă să efectueze o altă operație pe polinoamele deja introduse, astfel revenind la pasul 2 prezentat anterior.

# Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

## Decizii de Proiectare:

Utilizarea aplicatiei presupune introducerea a 2 polinoame in format String in cele 2 casute(TextField) si prin apasarea a unor butoane (Button ) se vor realiza diferitele operatii, iar rezultatul va fi afisat in alta casuta (TextField- needitabil pentru utilizatorul aplicatiei). Deorece este posibil ca utilizatorii sa introduca un input gresit cum ar fii sa insereze un spatiu sau sa introduca monoamele intr –o alta ordine decat cea normala de exemplu (x+x^2+1) sau daca utilizatorul nu introduce coeficientul 1 sau gradul 1 sau 0 la x, sau chiar x, atunci programul o sa le rearanjeze si programul o sa introduca de la sine gradul si coeficientul indicat. Programul va elimina si eventualele spatiii introduce.

In cazul introducere unui input gresit (precum introducerea de caracatere necomforme sau incorrect pozitionate, prezenta simbolului ^ fara a fi urmat de un intreg , ex: x^, 3x^+, etc.) se va arunca exceptia InvalidInputDataException.

In cazul impartirii in care cele 2 polinoame nu se pot impartiti (dempartitul e 0 ) se va arunca DivideByZeroPolynomialException.

Aceste exceptii se vor trata de Clasa controller care va seta textul casutei pentru rezultatat cu mesajul corespunzator exceptiei.

## Structurarerea pe pachete

Respectind arhitectura Model-View-Controler programul va avea cele 3 pachete caracteristici: Model, View, Controller. Pe langa aceste aceste pachete mai exista si pachetul Exceptions care grupeaza clasele ce extend clasa Exceptions. Clasele care contin testele unitare vor fi grupate inntr-un alt pachet special pentru teste.

### Pachetul Model

Pachetul grupeaza clasele, pachetele care contin logica aplicatiei. Subpachetul Monomial grupeaza clasele ce implementeaza un monom, iar Polynomial continue momentan doar clasa Polynomial insa este gandit astefel incat la upgrade al aplicatiai sa poata gazdui si alte clase care sa ofere multiple implementari ale unui plynom.

#### Pachetul Monomial

##### Clasa Monomial

Clasa Monomial este definite pentru a usura lucrul cu polinoamele insusi fiecare monom avand un coefficient si un grad. Aceasta clasa are ca variabila instant o variabila de tip private int care reprezinta puterea monomului si totodata o variabila de tip private double ce reprezinta coeficientul polinomului. Cele 2 variabila instant au fost declarate de tip private pentru ca mai apoi sa nu poata fii modificata din alte clase.

Clasa contine doar un constructor care da unui nou obiect instantiate un double drept coeficient si un int drept grad si totodata implementeaza si metodele getters pentru cele 2 variabile instanta.

Deasemenea aceasta clasa va implementa interfata Compararable, implementand metoda compareTo pentru a putea sorta descrescator in funtctie de puterea monomului.

##### Clasa IntegerMonomial

Extinde clasa Monomial implementeaza metoda toString astfel incat coeficientii sa fie afisati ca numere intregi. Introducerea acestei clase a fost pentru a putea usura o dezvoltare ulterioara a aplicatiei.

##### Clasa DoubleMonomial

Extinde clasa Monomial implementeaza metoda toString astfel incat coeficientii sa fie afisati ca numere intregi. Introducerea acestei clase a fost pentru a putea usura o dezvoltare ulterioara a aplicatiei.

#### Clasa Monomial

##### Clasa Plynomial

Clasa polinom reprezinta un polinom. Aceasta clasa are ca variabile instanta o colectie de monoame care reprezinta polinomul propriu zis dar si gradul care reprezinta gradul monomului de pe pozitia 0 ( cel cu puterea cea mai mare).

#### Clasa OperationModel

Prin intermediul acestei clase se realizează operații pe polinoame precum: adunarea a două polinoame, scăderea a două polinoame, înmulțirea a două polinoame, împărțirea a două polinoame, derivarea unui polinom și integrarea unui polinom. Fiecare dintre aceste operații reprezintă o metodă în cadrul acestei clase. De asemenea, pentru o mai bună modularizare a codului și pentru a ușura implementarea unora din operațiile menționare anterior, clasa în cauză, înglobează și operații pe monoame, cum ar fi: înmulțirea a două monoame, împărțirea a două monoame, derivarea unui monom sau integrarea unui monom. O metodă importantă din cadrul clasei OperationModel, este metoda groupMonomialsWithSamePower, care are rolul de a grupa elementele care au aceeași putere. O alta metoda utila este metodda eliminatesRedundantTerms care are rolul de a elimina termini redundanti (cei cu coeficientul 0) din polinom.

#### Clasa ParsingDataModel

Această clasa este folosită pentru a extrage datele introduse de utilizator. Mai precis, prin intermediul acestei clase extragem coeficientul și puterea și creăm un monom, care apoi va fi introdus în lista de monoame a polinomului care e în curs de construire. Așadar, prin această clasă preluăm datele necesare despre polinomul nostru. Metoda din această clasă care se ocupă cu acest lucru se numește „ parseStingToPolynomial”.

### Pachetul Controller

#### Clasa Controller

Clasa Controller cuprinde metodele prin intermediul cărora se realizează de fapt legătura dintre partea de model și de vizualizare. Prin intermediul acestor metode putem controla accesul la aplicația noastră.

Cuprinde metode care se ocupă cu preluarea informațiilor introduse de utilizator in textField, și metode care implementează acțiunile butoanelor din interfață, care se ocupă fie cu calcularea anumitor operații, fie cu resetarea informațiilor.

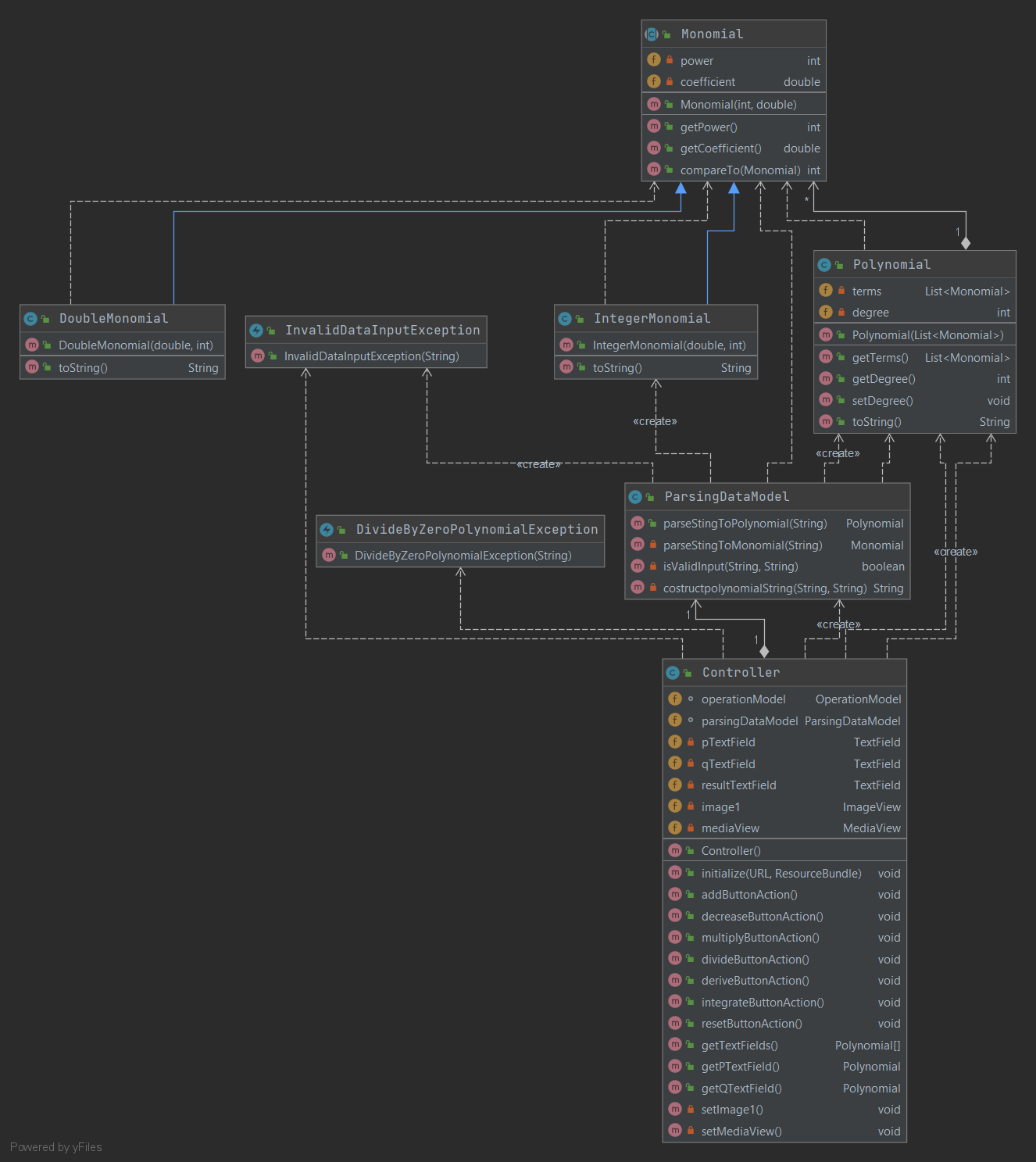
Tot acesta clasa se ocupa si de initializarea aplicatiei (fundal video,imagini).

### Pachetul View

Pachetul *ViewPackage* cuprinde partea de cod care realizează interfața grafică. Prin intermediul claselor din acest pachet i-a naștere partea vizuală a proiectului, aceea care interacționează cu utilizatorul.

Fiind o aplicatie JavaFx pachetul nu va contine nicio clasa. In schimb aici vom avea fiserul fxml care va continue structura interfetei UI. Tot in acest pachet se regasete si un fisier CSS care va contine partea de cod care implementeaza diferite stiluri pentru interfata. Resursele necesare pentru interfata UI, imaginile si vidioclipul penru fundal, vor fi stocate tot in acest fisier.

## Diagrama UML



# Implementare

##### **Metode importante implementate:**

## Clasa OperationModel

### eliminatesRedundantTerms

Aceasta metoda peimeste o lista de monoame si elimina toti termini cu coeficientul egal cu 0. Daca lista ramane va elemrnte se va introduce monomul cu coeficientul 0 si puterea 0.

### groupMonomialsWithSamePower

Aceasta metoda primeste o lista de monoame si grupeaza prin insumare toti termini cu puterea egala. Deoarece se poate intalni cazul in care sua moanoamelor de aceasi puteri din lista primita ca parametru sa fie 0 vom apela *eliminatesRedundantTerms* care va elimina acesti noi termini redundanti.

### addsPolynomials

Aceasta metoda relizeaza adunarea polinoamelor primite ca parametri (polynomial1 – polynomial2). Am declarant o noua colectie in care am adaugat toate monoamele din cele polinoame date ca parametru . Am creat un nou polinom care are a carui colectie de polinoame este chiar acest arraylist si am apelat pe Polinomul nou creat metoda *groupMonomialsWithSamePower*, pentru reducere termeni asemenea in cazul in care cele 2 polinoame au polinoame cu acelasi grad si pentru a elimina termenii redundanti .

### decreasesPolynomials

Aceasta metoda relizeaza scaderea polinoamelor primite ca parametri (polynomial1 – polynomial2). Am declarant o noua colectie in care am adaugat toate monoamele din primul polinom nemodificate, iar apoi am adaugat toate monoamele din cel de al doilea polinom insa acestea au avut coeficientul cu -1. Am creat un nou polinom care are a carui colectie de polinoame este chiar acest arraylist si am apelat pe Polinomul nou creat metoda *groupMonomialsWithSamePower*, pentru reducere termeni asemenea in cazul in care cele 2 polinoame au polinoame cu acelasi grad si pentru a elimina termenii redundanti .

### multipliesPolynomials

Metoda are ca parametri 2 polinoame si returneaza un obiect de tip polinom si consta in crearea unei noi colectii rezultat iar mai apoi parcurgerea listei de monoame din polinomul dat ca prim parametru. Pentru fiecare monom se parcurge lista de monoame din cel de al doilea polinomul dat ca parametru si introducerea in aceasta colectie un nou monom care are coeficientul produsul dintre coeficientii celorlalte 2 si ca grad suma gradelor monoamelor I si j . Dupa introducerea monoamelor se instantiaza un nou obiect de tip polinom care are ca lista de monoame colectia create anterior. Si in continuare se apeleaza metoda de adunarea a plinomilui intermediar, rezultat prin inmultirea monului current din prima primul plynom si toate moanoamele din cel de al doilea, cu polinomul resultat,.

### dividesPolynomials

Metoda are ca parametri 2 polinoame si returneaza o colectie de polinoame care reprezinta catul si restul imparitiri polinomului 1 la polinomul 2. Aceasta e cea mai complexa metoda din intreaga clasa pentru ca prezinta mai multe exceptii care trebuie tratate .

Cazul 1: polinomul 2 este 0. Pentru acest caz se va arunca exceptia *DivideByZeroPolynomialException.*

Cazul 2: polinomul 1 are gradul mai mic decat polinoml 2 sau polinomul 1 este 0. Pentru acest caz se va polinomul cu catul 0 si restul egal cu polinomul 1.

Cazul 3: este cazul general. Catul va fi egal cu raportul dintre cei mai semnificativi termini ale celor 2 polinoame adunat cu catul impartirii dintre diferenta dintre polinumolu1 si produsul termenul semnificativ al celui de al doilea polinom si raportul calculat anterior la polinomul 2;

### derivesPolynomial

Aceasta metoda realizeaza derivarea termen cu termen (monom cu monom ) a polinomului primit ca parametru si returneaza un polinom rezultat in care fiecare monom este transformat dupa formula c\*x^n=n\*x^n-1 oricare ar fii n un numar natural nenul. In cazul n = 0 rezultatul va fi 0.

### integratesPolynomial

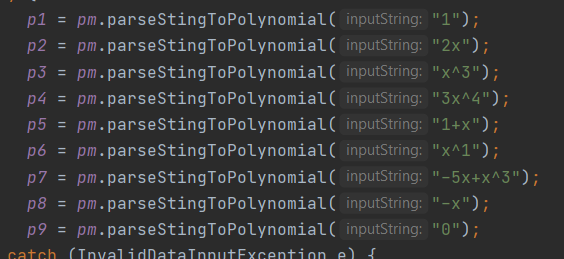
Aceasta metoda realizeaza integrarea termen cu termen (monom cu monom ) polinomului primit ca parametru si returneaza un polinom rezultat in care fiecare monom este transformat dupa formula c\*x^n=(1/(n+1))x^(n+1) oricare ar fii n un numar natural si c numar natural nenul . Pentru n = 0 rezultatul va fi 0.

##### **Interfata:**

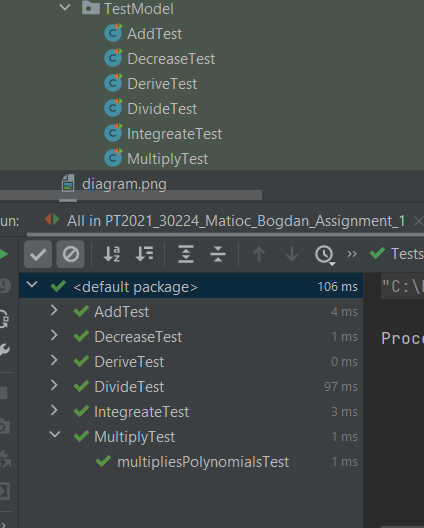
După cum se poate observa interfața are, în primul rând, un titlu sugestiv care anunță întru-totul așteptările pe care utilizatorul trebuie să le aibă vizavi de acest proiect. Urmează apoi două casete text însoțite de două etichete, casete în care utilizatorul trebuie să introducă câte un polinom. După ce acesta a introdus polinoamele, respectând formatul specificat, are posibilitatea de a apăsa pe oricare dintre cele 6 butoane, care au asignate pe ele semnul corespunzător operației care se va realiza odată cu apăsarea lui. După alegerea operației dorite, în caseta text următoare o să apară rezultatul corespunzător. Dacă apoi se apasă un alt buton de operație, atunci rezultatul din caseta text finală o să se schimbe. Se poate observa în dreapta jos și un buton de resetare. Utilizatorul are posibilitatea de a șterge conținutul tuturor casetelor text, prin simpla apăsare a acestui buton de resetare. Astfel, interfața va ajunge să arate exact așa cum se vede și în imaginea de mai sus.

# Rezultate

In ceea ce privește proiectul realizat, am făcut și teste unitare pentru validarea funcționării corespunzătoare a tuturor operațiilor pe polinoame. Pentru fiecare operatie: adunare, scadere, inmultire, impartire, derivare si integrare am create cate o clasa de test. In fiecare clasa am instantiate 9 polinoame dinsticte prin intermediul cararora reusesc sa cuprind un set larg de posibile input-uri.



Prin cobinarea cate 2 am reusit sa acopar toate cazurile ca ear putea genera erori. (Sugerez citirea codului din pachetul TestModel pentru a putea viziona toate testele deoarece acestea sunt intr-un numar ridicat).



# Concluzii

Prin prisma acestei teme mi-am consolidat cunoștințele de programare orientată pe obiect. De asemenea, am reușit să înțeleg mai bine modul în care trebuie să îmi structurez codul și logica pe care trebuie să o aplic pentru a respecta formatul Model-View-Controller. Totodată, prin intermediul acestei teme am reușit să înțeleg ce este și cum se folosește regex pentru a reuși să extrag coeficientul și puterea monoamelor din polinoamele introduse de utilizator.

Fiind primul proiect implementat folosiind JavaFx pot spune ca invatarea punerea in practica a cunostintilor dobindite din tutoriale online si imbinaea cu arhitectura MVC m-a ajutat sa invat aceasta biblioteca .

Din punct de vedere al dezvoltării ulterioare a acestui calculator polinomial, există vaste posibilități de îmbunătățire. În primul rând, acest calculator polinomial ar putea fi modificat așa încât să permită utilizatorului să introducă coeficienți reali monoamelor din polinom. Această îmbunătățire s-ar putea realiza fără a fi nevoie de prea multe modificări pe codul deja existent, în principiu ar trebui să fie puțin modificat regex-ul așa încât acesta să ia și coeficienți reali. Mai mult decât atât, acest calculator ar putea să fie dezvoltat așa încât să poată realiza calcule pe polinoame cu mai multe variabile. Pentru a putea adauga aceasta optiune ar trebui modificata atat clasa Monom prin adaugarea unei variabila instante care sa retina variabila monomului cat si metodele care implementeaza operatiile pe monoame. Deasemenea vor fi necesar si mici modificari la metodele care efectueaza parsarea.