DOCUMENTATIE

TEMA 1

NUME STUDENT: BULEANDRA RUBEN

GRUPA: 30221

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

*Se vor prezenta: (i) obiectivul principal al temei printr-o fraza, si (ii) obiectivele secundare sub forma tabelara sau ca lista. Obiectivele secundare reprezinta pasii care trebuie urmati pentru indeplinirea obiectivului principal. Fiecare obiectiv secundar va fi descris si se va indica in care capitol al documentatiei va fi detaliat.*

Obiectivul temei este realizarea unui calculator cu interfata grafica in care utilizatorul introduce polinoame, alege operatia si primeste rezultatul.

Calculatorul polinomial este implementat folosind un limbaj orientat pe obiecte (Java), are o interfata grafica si foloseste modelul arhitectural MVC (Model View Controller).

# Analiza problemei, modelare, scenarii, cazuri de utilizare

*Se va prezenta cadrul de cerinte functionale formalizat si cazurile de utilizare ca si diagrame si descrieri de use-case. Descrierile use-case-urilor se vor face sub forma unui flow-chart ori sub forma unei liste continand pasii executiei fiecarui use-case.*

Calculatorul trebuie sa implementeze urmatoarele operatii:

-adunare

-scadere

-inmultire

-impartire

-derivare

-integrare

1. In oricare caz, utilizatorul trebuie mai intai sa introduca doua polinoame sub forma de text.

2. Utilizatorul apasa apoi pe butonul aferent operatiei.

3. Rezultatul dorit va fi vizibil utilizatorului.

Daca utilizatorul nu introduce un text care poate fi interpretat drept polinom de catre calculator, atunci trecem din nou la pasul 1 (utilizatorul introduce polinoamele).

Calculatorul va face posibila si introducerea polinoamelor prin intermediul unor butoane din interfata grafica, fara a fi nevoie de a utiliza tastatura.

# Proiectare

*Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite, interfetele definite si algoritmii folositi (*daca e cazul)

Aplicatia foloseste arhitectura MVC. M - model; in model se vor face toate calculele si modelul va tine minte toate datele folosite de program. V -View; reprezinta interfata grafica si tot ce tine de aceasta. C – controller. Controller-ul va face legatura dintre model si view si dintre utilizator si aplicatia in sine. Model-ul nu trebuie sa stie de existenta vreunui view, ci doar va executa comenzile primite de la controller. La fel si view-ul; acesta nu trebuie sa stie de existenta vreunui model, ci doar afiseaza ce primeste de la controller.

Am folosit nume sugestive pentru clase: Model, GUI si Controller.

In Model se vor tine minte doua polinoame, cele folosite la operatii, si va mai contine un polinom pentru rezultat. Deci in model se introduc doua polinoame P si Q, se alege operatia dorita, iar rezultatul va fi pus in polinomul result.

GUI reprezinta interfata grafica. Contine un grid de butoane pentru eliminarea nevoii de tastatura, 3 casute de text pentru polinoame, si cateva butoane pentru controlul operatiei de catre utilizator. Pentru simplitate si pentru ca calculatorul sa isi pastreze o oarecare forma la redimensionare am folosit splitPane-uri. Un splitPane se poate imparti in doua si se poate redimensiona dupa preferinte.

Singurul mod prin care controller-ul si GUI-ul interactioneaza sunt listener-ii. In controller sunt creati listener-ii pentru butoanele de control, care sunt apoi trimisi la GUI.

La inceput polinomul va fi reprezentat printr-un sir de caractere, dar asta nu ne ajuta foarte mult la calculele matematice ce urmeaza, asa ca un polinom va fi in continuare reprezentat de o instanta a clasei Polynomial.

Clasa Polynomial contine o lista de monoame.

Clasa Monomial este de fapt doar o structura folosita pentru a tine minte un coeficient real si o putere intreaga.

Pentru listele de monoame din polinoame, am folosit liste inlantuite. Deoarece va fi nevoie mai ales sa introducem monoame in lista la anumite pozitii, sunt mai eficiente listele inlantuite decat array-urile spre exemplu.

Algoritmii folositi pentru operatiile matematice sunt eficienti din punct de vedere al timpului de executie.

In unele locuri am folosit iteratoare, tocmai pentru eficienta, iar in alte locuri am folosit bucle for each.

Adunarea se va face folosind iteratoare, nu bucle for each, deoarece complexitatea va fi data de gradul celui mai mare polinom.

Scaderea este la fel ca adunarea.

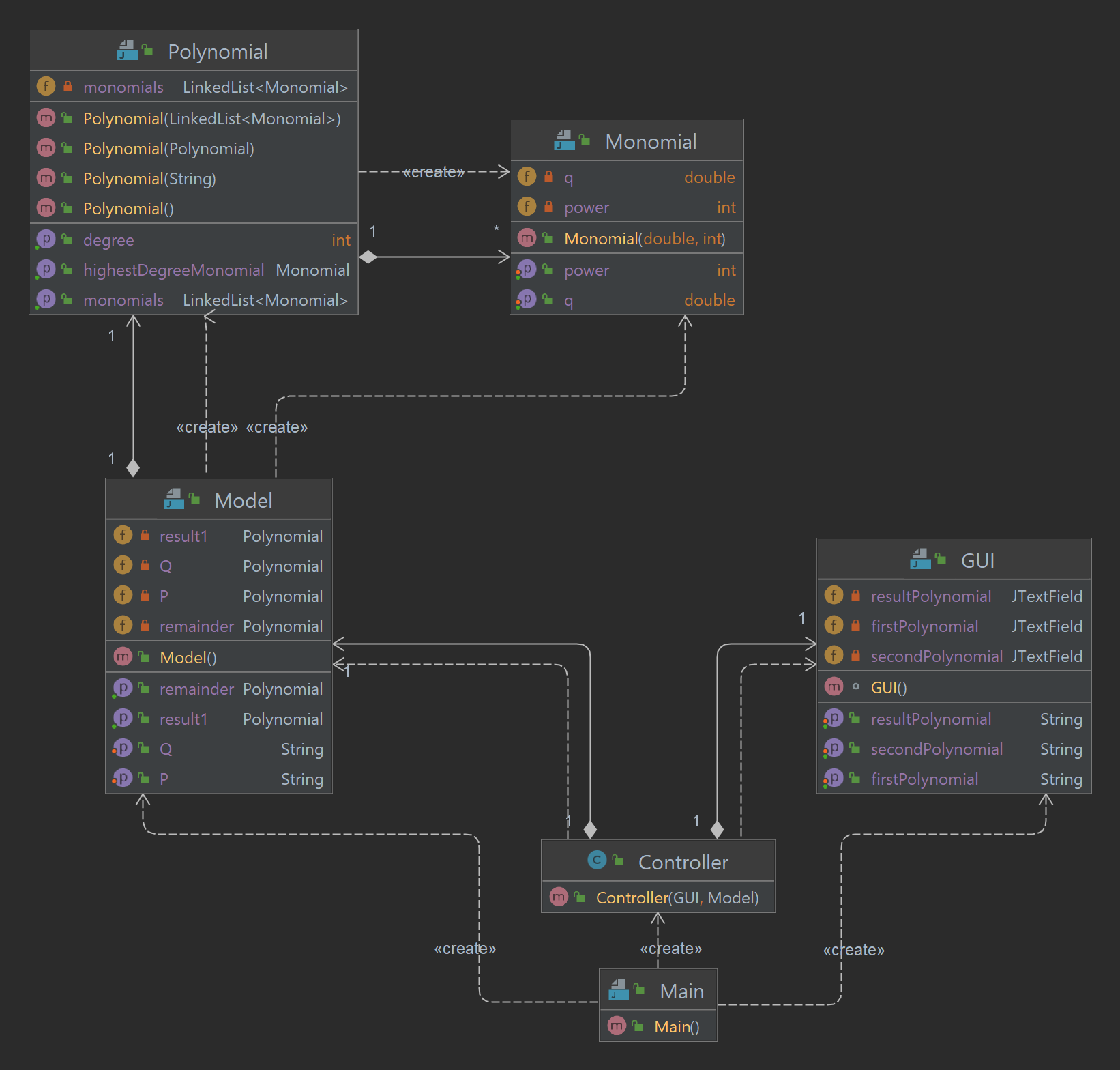
Inmultirea se va face folosind bucle for each, rezultat o complexitate O(n\*m) unde n si m sunt gradele fiecarui polinom.

Impartirea se va face folosind algoritmul „long division”, si va avea o complexitate asemanatoare inmultirii. Algoritmul presupune ca o modalitate de inmultire si de scadere a polinoamelor deja exista.

Aplicatia foloseste conceptul de incapsulare. Variabilele sunt private, si pot fi accesate doar prin intermediul getter-lor si setter-ilor.

Aplicatia foloseste si conceptul de supraincarcare a metodelor. De exemplu, pentru a instantia clasa Polynomial, se vor putea da ca argument constructorului atat o lista de monoame, cat si un string care mai apoi sa fie transformat intr-o lista de monoame, si va avea si un constructor de copiere.

Diagrama UML:



# Implementare

*Se va descrie fiecare clasa cu campuri si metodele importante. Se va descrie implemantarea interfetei utilizator.*

Monomial:

Clasa Monomial reprezinta matematic un monom. Un monom in matematica este caracterizat de un coeficient si o putere. „2.5\*x^3” – ‚2.5’ este coeficientul, iar ‚3’ este puterea. Puterile sunt intregi, iar coeficientii sunt intregi reprezentati sub forma double. Contine getter-i si setter-i si un constructor simplu.

Polynomial:

Clasa Polynomial reprezinta matematic un polinom. O instanta a clasei Polynomial va contine o lista de monoame. Un astfel de polinom poate fi creat in doua feluri.

Furnizarea unei liste de monoame: in acest caz lista va fi automat sortata descrescator, se vor adauga monoamele lipsa (ex: 0\*x^3), si se vor verifica dubluri (1\*x^2 si 2\*x^2 devin 3\*x^2).

Printr-un sir de caractere. In acest caz, sirul respectiv va fi mai intai modificat (metoda stringTransform) astfel incat sa poata fi despartit in monoame de metoda split. Metoda split desparte stringul in monoame, luandu-se dupa semne (+,-).

Exista de asemenea si un constructor pentru copiere.

Metoda stringTransform primeste ca argument un String care trebuie sa aiba forma unui polinom si ii adauga la inceput un ‚+’ daca lipseste, adauga ‚\*’ intre coeficienti si x si adauga ‚1\*’ in cazul monoamelor cu coeficient 1.

Aceasta metoda este folositoare pentru urmatorii pasi.

Metoda split primeste ca argument un String si foloseste functia split pentru a desparti un sir de caractere conform unui model. In acest caz trebuie sa despartim sirul de caractere in monoame, deci despartim dupa + sau -. De asemenea este important sa pastram semnul inaintea monoamelor.

Apoi, pentru fiecare monom, se cauta ‚x’.

Daca x nu apare in monom atunci consideram x^0.

Daca x apare, atunci puterea este numarul de dupa ‚^’.

Daca un numar nu apare inainte de x, atunci coeficientul este 1.

Daca un numar apare inainte de x atunci trebuie cautat ‚/’.

Dupa ce am stabilit ce forma are monomul se poate transforma din string intr-o instanta a clasei Monomial, cu coeficient real si putere intreaga.

Metoda mergeMonomials itereaza lista de monoame si cauta cate doua monoame cu aceeasi putere. Daca gaseste, le uneste, adunandu-le coeficientii. Pentru asta am folosit doua monoame, unul plasat dupa celalalt, si metoda remove a iteratorului.

Metoda addMissingPowers itereaza lista de monoame, asemanator metodei mergeMonomials si adauga in lista de monoame toate puterile care lipsesc. De exemplu, daca polinomul este x^3+x^1, atunci lista de monoame va contine de fapt x^3, 0x^2, x^1, 0x^0. Aceasta operatie ajuta la realizarea tuturor operatiilor matematice.

Metoda deleteFirstZeroQ sterge din lista monomul de la inceput, daca acesta are coeficientul 0. Acest lucru este necesar mai ales dupa operatiile de inmultire si impartire deoarece dupa ele poate sa ramana la inceput un monom inutil si care incurca.

Metoda toString returneaza un String care reprezinta polinomul intr-o forma placuta la privit.

Metoda getDegree returneaza puterea celui mai mare monom.

Model:

Clasa Model contine datele folosite (doua polinoame de intrare, doua polinoame de iesire) si logica din spatele operatiilor de adunare, scadere, inmultire, etc., fiecare cu metoda aferenta operatiei.

Operatia de adunare:

Tinand cont ca lista de monoame este ordonata descrescator (puterea cea mai mare prima), pentru adunare trebuie deci sa luam cele doua liste de monoame de la cele mai mici puteri, in mod simultan si le adunam coeficientii. Continuam pana cand se termina monoamele ambelor liste.

Operatia de scadere:

La fel ca la adunare, doar ca scadem coeficientii.

Operatia de inmultire:

Doua bucle for each, una in interiorul celeilalte, pentru a lua fiecare termen din prima lista si a-l inmulti cu fiecare termen din a doua lista. Pentru inmultire se va folosi metoda multiplyMonomials, care primeste doi monomi si returneaza un monom cu coeficientii inmultiti si puterile adunate;

Operatia de impartire:

Respecta algoritmul „long division” pentru impartirea polinoamelor. Vor rezulta doua polinoame: ce se poate imparti si ce nu se poate imparti. Ce se poate imparti se pune in variabila result, iar polinomul ce nu se poate imparti este considerat rest si este pus in variabila remainder.

Operatia de derivare:

Se ia fiecare monom, puterea i se scoate in fata, se inmulteste cu coeficientul, iar puterea scade cu 1. Monoamele raman aceleasi si se ia in considerare cazul in care primul monom va avea coeficientul 0, caz in care se sterge.

Operatia de integrare:

Puterea fiecarui monom creste cu 1, iar coeficientul se imparte la puterea initiala.

Polinoamele de intrare se pot seta folosind metodele setP, setQ, metode care accepta atat un polinom, cat si un String.

Rezultatele se preiau prin metodele getResult1 si getRemainder (in cazul impartitii).

Clasa GUI:

Reprezinta interfata grafica. Extinde JFrame.

Panoul principal este impartit folosind JSplitPane-uri.

Un JSplitPane permite impartirea unei zone in doua parti cu un anumit raport, sau un anumit numar de pixeli.

Am folosit JSplitPane pentru a putea pastra proportiile zonelor la redimensionarea ferestrei.

Incepand de jos am impartit in felul urmator:

Mai intai partea de butoane 0-9, X, +, - etc., apoi partea de butoane de control Add, Subtract, etc., apoi partea de input, output, cu cele doua polinoame introduse si cel de iesire.

Butoanele 0-9 este partea de jos a unui splitPane, iar butoanele de control este partea de sus. Un alt splitPane contine splitPane-ul anterior si label-urile si textField-urile. La redimensionare, impartirea isi pastreaza raportul initial.

GUI-ul contine metode pentru obtinerea polinoamelor sub forma de String si pentru setarea polinomului de iesire. De asemenea, contine si o metoda care primeste care argumente listeneri, pe care mai apoi le atribuie butoanelor de control.

Controller:

Clasa Controller implementeaza interactiunea dintre Model si GUI. Contine si implementarea listener-ilor.

Primeste ca argumente un model si un GUI;

Apeleaza metoda addListeners din GUI pentru a adauga fiecarui buton cate un listener si in functie de caz sa efectueze operatia dorita.

# Rezultate

*Se vor prezenta scenariile pentru testare cu Junit sau alt framework de testare.*

Am folosit JUnit pentru a testa functia de split a clasei Polynomial si pentru a testa fiecare operatie matematica din Model.

Pentru split:

6.2x^4-2/10\*x^2+3x+1;

6x^4-2.6x^2

Am cuprins cazul in care string-ul contine sau nu \*, coeficientul este fractie sau direct un numar cu virgula.

Pentru operatiile matematice am folosit teste parametrizate:

P = 4x^6+x^5-2x^2+1

Q = 3x^5+2x^3-2x

P = 2x^4-3x^3-15x^2+32x-12

Q = x^2-4x-12

# Concluzii

*Se vor prezenta concluziile, ce s-a invatat din tema, dezvoltari ulterioare.*

Proiectul a fost distractiv, am invatat sa caut pe internet, sa caut parerile altor programatori cu mai multa experienta.

Am invatat ca operatiile matematice se pot realiza destul de usor in java, am exersat operatiile cu siruri de caractere si dupa parerea mea cel mai important lucru am invatat sa implementez arhitectura MVC.

Dezvoltari ulterioare:

Polinomul reprezentat in casuta de text din GUI ar putea fi repezentat in permanenta sub forma unui polinom, deci sa nu necesite transformarea intr-o instanta Polynomial decat in momentul in care utilizatorul introduce sau modifica un polinom. De asemenea interfata grafica ar putea oferi utilizatorului posibilitatea de a copia polinomul rezultat intr-unul de intrare.

Metoda de afisare a polinoamelor poate fi imbunatatita prin afisarea sub forma de fractie acolo unde se poate sau afisarea puterilor in mod corespunzator ()

Calculatorul mai poate fi imbunatatit printr-o modalitate de a vizualiza grafic polinoamele.

De asemenea s-ar putea afisa pasii intermediari pentru fiecare operatie (cel mai probabil cu scop didactic).

# Bibliografie

<https://www.youtube.com/watch?v=19xwjgsC_58&t=665s>

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>