**DOCUMENTATIE TEMA 1**

**CALCULATOR POLINOAME**

**Petra Adrian Mariel**

**30225**

**Profesor Laborator Assist Moldovan Dorin**

Contents

[1. Cerinte Functionale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv Principal: 3](#_Toc476131447)

[2.2. Obective Secundare: 3](#_Toc476131448)

[3. Analiza Problemei 3](#_Toc476131449)

[4. Proiectare 3](#_Toc476131450)

[4.1. Structuri de date 3](#_Toc476131451)

[4.2. Diagrama de clase 3](#_Toc476131452)

[4.3. Algoritmi 3](#_Toc476131453)

[5. Implementare 4](#_Toc476131454)

[6. Testare 4](#_Toc476131455)

[7. Concluzii si Dezvoltari Ulterioare 4](#_Toc476131456)

[8. Bibliografie 4](#_Toc476131457)

# Cerinte Functionale

Dezvoltati un calculator de polinoame care sa perimita urmatoarele operatii:

* Citirea unui polinom de la tastatura sub forma 2 x ^ 3 – 3 x ^ 2 – x + 2
* Adunare , Scadere , Inmultire , Impartire , Derivare , Integrare

# Obiective

## Obiectiv Principal:

Scopul principal consta in realizarea unei aplicatii in Java care implementeaza operatii matematice folosind polinoame . Formarea acesteia presupune respectarea paradigmei de programare orientata pe obiect si principiile pe care aceasta le pune la dispozitie . Pe langa interactiunea dintre clase , obiecte si metode , aplicatia presupune cunostinte pentru manipularea obiectelor clasei String , alegerea unor structuri de date adecvate pentru reprezentarea polinoamelor si organizarea intr-o interfata prietenoasa pentru interactiunea cu utilizatorul.

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii | Dezvoltarea unei liste de comportare pe care ne asteptam ca programul sa o aplice | 3 |
| Alegerea structurilor de date | Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite si interfetele definite. | 4 |
| Impartirea pe clase | Alegerea potrivita a claselor cat si motivarea acestora | 4 |
| Dezvoltarea algoritmilor | Descrierea si motivarea alegerii algoritmilor potriviti | 4 |
| Implementarea solutiei | Gandirea celei mai optime strategii de rezolvare a problemei propuse | 5 |

# Analiza Problemei

Un use case este o metodologie utilizată în analiza sistemului pentru a identifica , clarifica și organiza cerințele sistemului . Un use case este alcătuit dintr-un set de posibile secvențe de interacțiuni între sisteme și utilizatori într-un anumit mediu și legate de un anumit scop . Se compune dintr-un grup de elemente (de exemplu clase și interfețe) care pot fi utilizate împreună într-un mod care va avea un efect mai mare decât suma elementelor separate combinate . Use case-ul ar trebui să conțină toate activitățile sistemului care au o semnificație pentru utilizatori. Un use case poate fi considerat ca o colecție de scenarii posibile legate de un anumit scop , într-adevăr , use case și scopul sunt uneori considerate a fi sinonime .

Activarea:   
 Utilizatorul alege una din cele 5 operatii disponibile .

Preconditii:  
 Introducerea unui polinom de forma „3x^2+1x^1-2x^0” in ambele spatii care pot fi completate de catre utilizator .

Post-conditii :  
 Va fi afisat rezultatul operatiei alese .

Decursul normal :  
 1 . Dupa introducerea ambelor polinome sub forma indicata mai sus utilizatorul alege operatie pe care doreste sa o obtina ( avand la dispozitie adunarea , scaderea , inmultirea si derivarea polinomului 1 sau 2);

2 . Rezultatul operatiei este afisat la finalui ferestrei unde initial scrie „ Rezultat ” ;

3 . Utilizatorul poate realege o alta operatie pentru a vedea alt rezultat , de asemeanea poate schimba polinoamele o data introduse ;

4 . Terminarea programului se face prin apasarea casetei „ X ” din coltul dreapta sus .

1 B . In cazul nerespectarii formatului prezentat mai sus programul va inceta sa mai functioneze .

# Proiectare

## Structuri de date

Pentru implementarea aplicatiei am utilizat structura de Model-View-Controller intru - cat ofera accesibilitate atat pentru utilizator , cat si pentru implementarea proiectului . Clasele sunt mai bine organizate , iar interactiunile intre obiecte sunt mai evidente . Modelul contine clasele Monom , Polinom , Model care reprezinta tot ce tine de modelul matematic si realizarea calculelor. Inferfata grafica este reprezentata prin Menu , ea constituie principala interactiune cu utilizatorul . Intre Model si View exista o legatura stabilita prin Controller . Acesta pe baza actiunilor utilizatorului preia datele introduse in Menu si le transmite mai departe catre Model pentru a fi procesate . Ca si structuri de date am folosit *int* ( pentru putere , coeficient ) , *String* ( pentru strp1, strp2, result ) , *Polinom* ( pentru p1 , p2 ) , *ArrayList < Monom >* ( pentru monoame ) , *Jpanel* ( pentru panel , panel1 , panel2 , panel3 , panel4 ) *, JtextField* ( pentru field1 , field2 ) *, JComboBox* ( pentru choice ) *, Jlabel* ( pentru result ) *.*

**Proiectare clase**

Aplicatia mea este alcatuita din 4 pachete denumite sugestiv dupa modul de implementare ales ( Model-View-Controller ) fiecare continand clasele utilizate in tot procesul de calcul , iar pe langa acestea se adauga Main Class ( clasa principala care le uneste pe toate ).

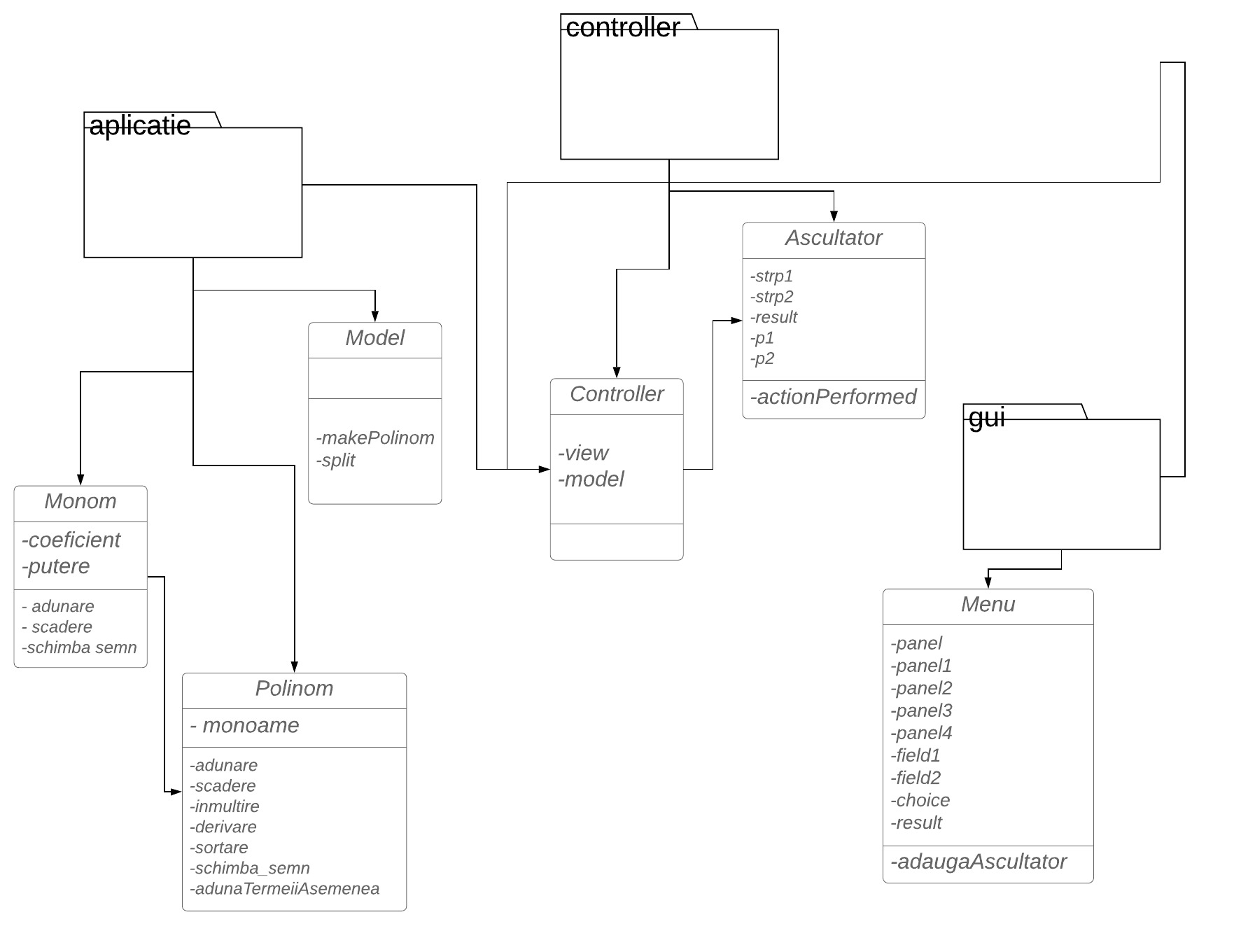
**Relatii**

In cadrul claselor implementate se observa usor relatiile de dependenta pe diagrama UML prezentata mai sus, cele mai evidente sunt dependentele intre clasele , Monom , Polinom dar si Model , View , Controller .

**Interfata utilizator**

Pentru interfata cu utilizatorul am ales o implementare simpla si sugestiva pentru a fi usor de folosit de catre oricine doreste sa utilizeze calculatorul de polinoame . La fiecare pas avem o bordura cu titlu care ne spune ce sa facem ( ce ar trebui sa introducem sau ce ar trebui sa selectam ) .

## Diagrama de clase

Din aceasta diagrama puteam vedea ca package ul aplicatie contine clasele Monom , Model , si Polinom ( mosteneste Monom ), este importata in clasa Controller . Package ul controller contine clasele ascultator si controller , unde Controller foloseste ascultator . Si package ul gui care contine clae Menu ; gui este importat de asemenea in clasa controller . 

# Implementare

Clasa Monom

Are doua campuri de intregi, coeficient si putere. Implementeaza interfata Comparable , reprezinta cea mai mica subunitate din cadrul proiectului .

Constructorul clasei primeste doi intregi , care vor reprezenta atributele obiectului in ordinea prezentata mai sus .

Avem getters pentru coeficient cat si pentru puterea , aceste atribute fiind private .

Ca si metode avem adunare care primeste un int , iar coeficientul obiectului este adunat cu intregul primit .

Si scadere unde scadem din coeficientul monomului intregul primit .

schimbareSemn ( ) este folosita pentru a schimba semnul coeficientului obiectului .

Am metoda toString ( ) pentru o afisare a obiectului si din cauza ca implementam interfata Comparable < Monom > trebuie sa dam override la metoda compareTo , cu ajutorul careia sortez polinoamele in ordine descrescatoare dupa puterile lui „ x ” .

Clasa Polinom

Ca si atribute are doar un private ArrayList < Monom > .

Metoda adunare ( Polinom p ) face adunarea obietului de tip Polinom cu Polinom p .

Metoda scadere ( Polinom p ) face scaderea obiectului de tip Polinom cu Polinom p .

Metoda inmultire ( Polinom p ) face inmultirea obiectului de tip Polinom cu Polinom p .

Metoda derivare ( ) face derivarea de ordin 1 a obiectului de tip Polinom .

Metoda toString ( ) folosita pentru conversia unui polinom in textul care va fi afisat pe ecran .

Metoda sortarea ( ) folosind sort ( ) din java . util . Collections .

SchimbaSemn ( ) folosita utilizata pentru functia de scadere .

adunaTermeniiAsemenea ( ) unde adunam termenii cu aceeasi putere rezultati din inmultire .

Clasa Model

Se ocupa cu formarea polinomului dupa ce utilizatorul il introduce ca si String . Sunt folosite patru functii : validare ( String ), validareMonom ( String ), formeazaPolinom ( String ) si split ( String ) . Cand un string este introdus in casutele text este preluat de vedere si transmis mai departe pentru a putea fi prelucrat de model . Metoda validare ( String ) verifica in primul rand daca printre caracterele introduse exista altele inafara de cifrele 0 – 9 , x , ^ , + , - sau daca sirul contine mai multe carctere consecutive care nu sunt cifre iar in cazul in care exista metoda returneaza false. Daca primele doua conditii sunt respectate atunci se merge mai departe prin separarea String - ului in parti mai mici in functie de + sau - . Prin utilizarea metodelor din clasele Pattern , Matcher si a expresiilor regex separam String-ul in mai multe " monoame " apoi se face validarea separata a fiecarui subsir . In cazul in care fiecare subsir este valid atunci tot polinomul este valid si mergem mai departe cu formarea lui . Functia formeazaPolinom ( String ) se foloseste de o functie ajutatoare split ( String ) . Dupa ce monomul a ajuns valid in corpul functiei trebuie analizat si trebuie sa extrag din el coeficientul si puterea , lucru realizat de functia split ( String ) care returneaza un vector de doua elemente primul fiind coeficientul, al doilea fiind puterea . Dupa ce a primit rezulatul din functia split ( String ) , functia formeazaPolinom ( String ) instantiaza un nou obiect din calsa Monom si il adauga la ArrayList pentru a forma polinomul . Functia split ( String ) trateaza toate cazurile posibile a formei monomului ( de exemplu daca un monom este introdus ca + x ^ 6 sau x ^ 6 cele doua sunt echivalente in ceea ce priveste coeficientul si puterea ) .

Adunarea si scaderea sunt implementate similar , in ambele cazuri toate monoamele din primul si al doilea polinom se introduc intr-o lista iar apoi se apeleaza metoda adunaTermeniiAsemenea ( Polinom ) generand astfel rezultatul . Singura diferenta consta in faptul ca la scadere se inverseaza semnul monoamelor din polinomul care se scade si apoi se face adunarea termenilor asemenea.

Impartirea se ajuta de functiile de scadere si inmultire deja implementate. Algoritmul este realizat in felul urmator : prima oara polinoamele se sorteaza apoi se ia primul monom din deimpartit si se imparte la primul monom din impartitor . Rezultatul va fi inmultit cu impartitorul si noul rezultat se scade din deimpartit . Primul monom din deimpartit e impartit cu primul monom din impartitor . . . se repeta din nou pasii pana cand cea mai mare putere a lui " x " din rezultat este mai mica decat cea mai mare putere din impartitor . Metoda returneaza un String care contine catul si restul . Metoda se foloseste de obtinerea primului monom din fiecare lista utilizand indexul din ArrayList si apoi imparte cele doua monoame prin intermediul puterilor si al coeficientilor .

Clasa Controller

Este folosita pentru a face legatura intre calculele executate si utilizatorul care dupa ce a introdus datele asteapta un rezultat . Astfel proprietatile obiectului din clasa Controller sunt reprezentate de alte doua obiecte , unul din clasa Menu si altul din clasa Model .

Cea mai importanta functionalitate a clasei Controller consta in implementarea metodei actionEvent ( Event ) pe care eu am ales sa o integrez intr-o clasa continuta , pentru o mai buna organizare . Aceasta metoda sta la baza interactiunii intre persoana si aplicatie . Ea verifica datele introduse de utilizator si in functie de operatia selectata din JComboBox se va folosi de clasa Operatii sau va seta un mesaj de eroare in cadrul etichetei ( label ) din clasa View .

# Concluzii si Dezvoltari Ulterioare

O aplicatie care ne pune la dispozitie un calculator de polinoame poate fi util pentu verificarea rapida a calculelor , imbunatatirea timpului de obtinere a rezultatului . Ca si dezvoltarea ulterioara este posibila integrarea aplicatiei intr-un sistem de calcul mai complex , ea poate reprezenta doar un modul care se ocupa cu partea de operatii a polinoamelor si se pot adauga metode in plus ( de exemplu calculul valorii polinomului intr-un anumit punct , gasirea radaciniilor functiei polinomiale ) .

In ceea ce priveste invatarea , cel mai important lucru pentru mine a fost dobandirea cunostintelor necesare pentru parsarea unui String si utilizarea expersiilor regex care in Java constituie o alternativa foarte buna , in favoarea unei functii lungi si costisitoare de verificare a fiecarui caracter si secventelor de caractere dupa cum trebuie sa urmeze .

# Bibliografie

1. <https://www.geeksforgeeks.org/split-string-java-examples/>
2. <https://stackoverflow.com/questions/3481828/how-to-split-a-string-in-java>
3. <https://google.github.io/styleguide/javaguide.html>
4. http://zetcode.com/tutorials/javaswingtutorial/