Logo, company name

Description automatically generated

**Documentatie tema 1**

**Nume student: Maracine Stefania Maria**

**Grupa: 30223**

# CUPRINS

[1. Obiectivul temei **.**](#_Toc95297885)

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

[3. Proiectare **.**](#_Toc95297887)

[4. Implementare **.**](#_Toc95297888)

[5. Rezultate **.**](#_Toc95297889)

[6. Concluzii **.**](#_Toc95297890)

[7. Bibliografie **.**](#_Toc95297891)

1. **Obiectivul temei**

Obiectivul principal al primei teme de laborator este un calculator de polinoame capabil sa faca un anumit numar de operatii cum ar fii: adunare, scadere, inmultire, impartire, derivare si integrare. In acest proiect s-au realizat doar adunarea, scaderea si derivarea. Polinoamele sun alcatuite din monoame, coeficientul si puterea fiecarui element din polinom. Obiectivele secundare ale proiectului ar fii implementarea unui cod capabil sa calculeze operatiile mentionate mai sus, proiectarea unei interfete grafice care sa fie usor utilizabila de utilizator si conectarea modelului de date cu interfata grafica. Obiectivele secundare vor fii explicate ulterior in capitolele ce urmeaza.

1. **Analiza problemei, modelare, scenario, cazuri de utilizare**

Proiectul implementat, avand rolul de a face anumite operatii intre doua polinoame, este un proiect simplu de utilizat, avand in interfata grafica butoane sugestive. Utilizaztorul introduce de la tastatura in doua textfield-uri unul sau doua polinoame de forma +- a1 ^ x1 +- a2 ^ x2 +- a3 ^ x3 … +- an ^ xn urmand apoi sa apese pe butonul pentru operatia pe care doreste sa o faca.

In partea de jos a interfetei grafice exista un al treilea textfield unde va aparea polinomul rezulat in urma operatiei sub forma [+- a1 ^ x1, +- a2 ^ x2, … , +- an ^ xn].

Pe langa butoanele care reprezinta operatiile implementate utilizatorul mai are si un buton de iesire care va opri programul si inchide interfata.

Pentru adunare si scadere input-urile sunt doua polinoame, iar output-ul este un singur polinom. La derivare ca input avem un singur polinor si ca output tot un singur polinom.

1. **Proiectare**

Diagram

Description automatically generated

In acest capitol este prezentata diagrama UML, clasele existente si structurile de date folosite.

Ca si clase avem cele 2 clase principale Monomial si Polynomial, apoi clasele Model, View si Controller iar la finar clasa TestClass.

Ca si structuri de date au fost folosite listele.

Modelul de design

MVC (Model-View-Controller) este un model în proiectarea software-ului utilizat în mod obișnuit pentru implementarea interfețelor cu utilizatorul.

Cele trei părți ale modelului de proiectare software MVC sunt:

Model: Gestionează datele și logica de funcționare.

Vizualizare: gestionează aspectul și afișarea.

Controler: direcționează comenzile către model și interfața grafică.

1. **Implementare**
2. **Clasa Monomial**

In aceasta clasa au fost declarate doua variabile private de tip int: “coeficient” si “putere” care reprezinta coeficintul si respective puterea unui element din polinom. Mai multe monoame vor alcatui un polinom, acest lucru se va explica in detaliu in clasa Polynomil.

Deoarece variabilele sunt private avem nevoie de settere gettere. Pentru variabila coeficient avem getterul “getCoeficient” si setterul “setCoeficient” iar pentru variabila putere avem getterul “getPutere” si setterul “setPutere”.

De asemenea in aceasta clasa avem metoda toString suprascrisa pentru a ne afisa in textfiel-ul 3 din interfata noastra grafica coeficientul si puterea impreuna cu caracterele necesare unei afisari de tipul: [+- a1 ^ x1, +- a2 ^ x2, … , +- an ^ xn].

1. **Clasa Polynomial**

Acesta clasa este clasa in care au loc operiatiile implementate. Incepem prim a importa libariile necesare pentru utilizarea listelor. S-a declarant o lista de monomiale care v-a reprezenta polinomul nostru. Astfel in aceasta lista vom avea o insiruire de numere grupate doua cate doua care reprezinta coeficientul si puterea fiecarui element din polinom, cate grupuri de monoame avem, atatea elementem v-a avea polinomul.

Dupa declararea listei avem cei doi contructori ai clasei, cel simplu, fara parametrii, si cel modificat de noi, cu parametru. Urmeaza apoi declararea setterului si getterului pentru lista.

Inante de inceperea implementarii metodelor pentru operatii mai avem o metoda numita “add” ce este folosita in clasa Model si care adauga coeficientul si puterea, adica monoamele, in polinomul nostru dupa convertirea lor dintr-un string.

Metoda “adunarePolinom” implementeaza adunarea dintre doua polinoame. Metoda este de tip Polynomial deoarece returneaza un polinom. Metoda are doi paramentrii amandoi de tipul polinom. Idea pe care am mers este urmatoarea: se ia doua variabile p1 si p2 unde se v-a salva gradul cel mai mare al fiecarui polinom. Se ia un foreach pentru a parcurge fiecare polinom si se verifica gradul maxim. Dupa stabilirea gradelor maxime se va face verificarea intre polinoame, care are gradul mai mare. Daca primul polinom are gradul mai mare atunci adunarea celor doua polinoame se va face in primul. Se ia rand pe rand fiecare element din primul polinom si se cauta in al doilea polinom elementul cu aceesi putere, in momentul in care l-am gasit, le adunam in elemental din primul polinom. Daca cel de-al doilea polinom este mai mare adunarea se face in cel de-al doilea. La fel se ia element cu element si se verifica sa aiba aceeasi putere pentru a le putea aduna, rezultatul adunandu-se in elemental din ce-l de-al doilea polinom. Dupa ce s-au parcurs polinoamele cu ajutorul unor foreach-uri intr-o alta variabila de tip Polynoial p3 se salveaza rezultatul final care v-a fii afisat mai tarziu. Am implementat metoda astfel deoarece adaugand in polinomul de grad mai mare sunt sigura ca nu voi rata niciul element din niciun polinom. Daca implementam intr-un polinom random trebuia sa verific ca am toate gradele de la ambele polinoame.

Metoda “scaderePolinom” implementeaza scaderea dintre doua polinoame. . Metoda este de tip Polynomial deoarece returneaza un polinom. Metoda are doi paramentrii amandoi de tipul Polynomial. Idea pe care am mers este urmatoarea: se ia doua variabile p1 si p2 unde se v-a salva gradul cel mai mare al fiecarui polinom. Se ia un foreach pentru a parcurge fiecare polinom si se verifica gradul maxim. Dupa stabilirea gradelor maxime se va face verificarea intre polinoame, care are gradul mai mare. Daca primul polinom are gradul mai mare atunci scaderea celor doua polinoame se va face in primul. Se ia rand pe rand fiecare element din primul polinom si se cauta in al doilea polinom elemental cu aceesi putere, in momentul in care l-am gasit, le scadem in elementul din primul polinom. Daca cel de-al doilea polinom este mai mare scaderea se face in cel de-al doilea. La fel se ia element cu element si se verifica sa aiba aceeasi putere pentru a le putea aduna, rezultatul scazandu-se in elemental din ce-l de-al doilea polinom. Dupa ce s-au parcurs polinoamele cu ajutorul unor foreach-uri intr-o alta variabila de tip Polynoial p3 se salveaza rezultatul final care v-a fii afisat mai tarziu. Am implementat metoda astfel deoarece adaugand in polinomul de grad mai mare sunt sigura ca nu voi rata niciul element din niciun polinom. Daca implementam intr-un polinom random trebuia sa verific ca am toate gradele de la ambele polinoame.

Mai departe avem metoda de derivare. Metoda este de tip Polynomial deoarece returneaza un polinom. Metoda “derivarePolinom” implementeaza derivarea unui polinom. Aceasta are un singur paramentru te tipul Polynomial. In aceasta metoda se ia un foreach pentru a parcurge lista de monoame a polinomului nostru. Derivarea este simpla, pentru fiecare element coeficientul se inmukteste cu puterea si puterea scade cu 1. Exact asa am procedat si in metoda noastra, pentru fiecare coeficient din polinom modificam valoarea acestuia inmultind-o cu valoarea puterii, iar abia apoi decrementam puterea. Polinomul rezultat se salveaza intr-o noua variabila de tip Polynimial care va fii ulterior afisata.

Ultima metoda este metoda suprascrisa toString care ne ajuta sa afisam rezultatul sub forma unui string in textfield3. Am modificat metoda astfel incat sa afiseze polinomul nostru alcatuit din monoame.

1. **Model**

Clasa model implementeaza o singura metoda, “conversiaDinStringInPolinom”, o metoda de tip Polynomial deoarece la final se va returna un polinom, care converteste un string intr-un polinom. Metoda primeste un singur parametru acela fiind un stirng ce v-a continue polinomul nostru si care v-a fii preluat din unul dintre cele doua textfield-uri. Idea din spatele aceste metode este urmatoarea: se ia string-ul introdus de utilizator si se converteste intr-ul array de caractere de forma: [+-, cifra, \*, x, ^ cifra, …]. Apoi parcurgem acel array element cu element si verificam mai multe conditii: daca elemental la care suntem este cifra atunci daca are inaintea lui un semn de plus sau minus si dupa el un semn de inmultire si la 3 elemente dupa el un semn de putere atunci elemental la care suntem este coeficientul si al 4 element dupa el este puterea. Salvam aceste 2 valori c[i] si c[i+4] in doua variabile locale si formam o noua variabila de tip monomial in care introducem aceste doua valori salvate. Apoi introducem noul monom format in polinomul nostru ce va fii afisat ulterior c metoda “add” din clasa Monomial ce a fost descrisa mai devreme.

1. Graphical user interface, application, Word

   Description automatically generated**View**

Clasa View extinde JFrame. Clasa View a fost facuta cu ajutorul unui program si anume Apache NetBeans IDE. In acest program am creat un form unde am folosit drag and drop pentru a pozitiona fiecare buton, label si textfield in locul droit, programul apoi generandu-mi codul necesar implementarii interfetei grafice. In aceata clasa avem declarant fiecare buton in parte, fiecare label si fiecare textfiled. Avem fiecare buton , label si textfield initializate. Am creat apoi o metoda pentru fiecare buton in parte pentru a sesiza cand butonul este apasat. Au fost setate textile pentru fiecare buton in parte ex: “Adunare”, “Derivare”, “0”, “6”. Cea mai mare bucata de cod din aceasta clasa este codul generat de Apache NetBeans IDE pentru interfata grafica, pentru dimensiunile ei, dimensiunile si pozitionarea fiecarui buton, label sau textfield.

1. **Controller**

In clasa controller am important libariile ActionEvent si ActionListener pentru a putea implementa fiecare buton in parte.

Ne-am declatat un obiect de tip model si unul de tip view si apoi in constructor am modificat fiecare buton in parte dandu-le fiecaruia cate un rol special. Nu au fost implementate toate butoanele ce apar in interfata grafica(vezi poza de mai sus) unele dintre ele find doar de aspect, deoarece polinoamele nu se introduce din butoane si ce la tastatura.

**Butonul unu** este butonul pentru adunare, astfel ca aici avem 2 variabile var1 si var2 in care se vor salva string-urile preluate din cele doua textfield-uri in care utilizatorul poate introduce polinoamele dorite. Apoi de declaram 2 obiecte de tip polynomial deoarece stringurile de mai sus trebuie convertite in polinomi. Ne declaram un nou obiect de tip model pentru a putea apela metoda de convertire din string in polinom din clasa Model descrisa mai sus. Apelam metoda de conversie pentru fiecare string in parte si salvam noile polinoame in cele 2 obiecte de tip polinom declarate mai devreme. Apelam apoi intr-o noua variabila de tip polinom metoda de adunare din clasa Polynomial descrisa mai sus si afisam apoi in cel de-al treilea textfield rezultatul obtinut.

**Butonul doi** este butonul pentru scadere, astfel ca aici avem 2 variabile var1 si var2 in care se vor salva string-urile preluate din cele doua textfield-uri in care utilizatorul poate introduce polinoamele dorite. Apoi de declaram 2 obiecte de tip polynomial deoarece stringurile de mai sus trebuie convertite in polinomi. Ne declaram un nou obiect de tip model pentru a putea apela metoda de convertire din string in polinom din clasa Model descrisa mai sus. Apelam metoda de conversie pentru fiecare string in parte si salvam noile polinoame in cele 2 obiecte de tip polinom declarate mai devreme. Apelam apoi intr-o noua variabila de tip polinom metoda de scadere din clasa Polynomial descrisa mai sus si afisam apoi in cel de-al treilea textfield rezultatul obtinut.

**Butonul trei** este butonul pentru derivare, avem o singura variabila in care vom salva stringul din textfield1 si pe care il vom converti in polinom asemanator ca si la cele 2 butoane anterioare, declarand un polinom si un model nou, apelanf metoda de conversie si apoi intr-ul nou polinom metoda de derivare, urmand apo isa afisam ultimul polinom creat in textfield3.

**Butonul patru** este butonul de exit care opreste programul si inchide interfata.

1. **Rezultate**

Nu s-a folosit folosit nicio metoda pentru testare.

1. **Concluzii**

Concluzia principala este importanta modelului de design pe care la-m ales in acest proiect si anume MVC deoarece este un design foarte bine gandit , codul fiind usor de inteles si de modificat.

1. **Bibliografie**

<https://www.geeksforgeeks.org/mvc-design-pattern/>

<https://www.microsoft.com/ro-ro/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling>