DOCUMENTATIE

TEMA *1*

NUME STUDENT: RADU DARIUS - RAZVAN

GRUPA: 30225

# 

# CUPRINS

[1. Obiectivul temei 3](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Proiectare 3](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297887)

[4. Implementare 5](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[5. Rezultate 5](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297889)

[6. Concluzii 8](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[7. Bibliografie 9](file:///D:\Faculta\AN2_SEM2\TP\Laborator\PT2021-2022_Documentation_Template.doc#_Toc95297891)

# Obiectivul temei

Obiectivul principal al acestei teme este implementarea unui calculator de polinoame, ce poate efectua operatiile de: adunare, scadere, inmultire, impartire, derivare si integrare.

Obiectivele secundare ar fi:

* analiza si proiectarea modelului de clase ce va fi ulterior utilizat (cap 3)
* implementarea si testarea celor 6 operatii (cap 4)
* implementarea GUI-ului dupa modelul de arhitectura MVC (cap 4)
* implementarea unei metode de conversie a polinomului introdus de utilizator, sub forma de string, in modelul proiectat de noi (cap 4)
* testarea interactiunii utilizatorului cu GUI-ul si traducerii polinomului introdus (cap 4)
* rezolvarea si transmiterea de mesaje pentru utilizator, in cazul anumitor introduceri nepermise. (cap 4)

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Use case: adunarea a doua polinoame.

Actor principal: utilizator

Pasi:

* Introducem primul polinom.
* Utilizatorul selecteaza operatia dorita. (adunare in cazul nostru)
* Aplicatia citeste polinomul si il stocheaza in memorie.
* Utilizatorul introduce al doilea polinom.
* Utilizatorul apasa pe butonul „=”.
* Aplicatia citeste polinomul si efectueaza operatia selectata. (adunare in cazul nostru)
* Aplicatia afiseaza rezultatul pe „ecranul” calculatorului.

# Proiectare

**Diagram

Description automatically generatedDiagrama de pachete** folosite in cadrul aplicatiei noastre poate fi obseravata mai jos. Este o diagrama clasica pentru arhitectura MVC, ce prezinta dependentele pachetelor (view-ul este dependent de model, iar controller-ul este dependent atat de model, cat si de view).

Fiecare pachet contine clase ce au un anumit rol bine definit. In pachetul „controller” exista clasa „CalculatorController” ce are rolul de a implementa functionalitatile aplicatiei noastre (Listenerii butoanelor, printarea rezultatului, apelarea metodelor ce implementaza operatiile, si gestiunea input-ului si output-ului, acestor operatii). In pachetul „view”, avem clasa „Calculator” ce are rolul de a implementa vedea efectiva a calculatorului nostru si se ocupa de citirea datelor de la utilizator. Iar pachetul „model”, contine clasele „Monomial” si „Polynomial” ce reprezinta modelele noastre de date, unde sunt implementate si operatiile.

Diagram

Description automatically generated

Mai sus putem observa diagrama de clase a aplicatiei noastre.

Apliacatia contine 4 clase + Main (care nu este reprezentat in diagrama).

* Clasa monomial este folosita ca model de date. Ea descrie un monom, format dintr-un coefiecient, numar real, si un grad, numar intreg. Aceasta clasa implementeaza 5 din cele 6 operatii, pentru a fi mai usor operatiile pe polinoame (polinoamele sunt doar niste siruri de monoame, deci operatiile se fac de fapt pe simple monoame). De asemenea, observa metoda de equals, pentru egalitatea dintre monoame si metoda de toString, ce o folosim pentru printarea rezultatului final.
* Clasa polynomial este folosita ca model de date. Ea descrie un monom, adica o lista de monoame. In cadrul acestei clase sunt implementate toate 6 operatiile.
* Clasa Calculator implementeaza view-ul efectiv al aplicatiei. Metoda readData este folosita pentru extragerea si conversia input-ului pentru a fi procesat mai departe de catre controller. Atributul op, va stica ce operatie urmeaza a fi executata (am ales sa folosesc aceasta aboradare, pentru ulterioare dorinte de scalare. Cum ar fi operatii multiple pe mai multe polinoame).
* Clasa controller implementeaza functionalitatea din spate a aplicatiei noastre. Aici sunt implementate innerclasses pentru a servi drept Listeners pentru butoanele noastre din view. Metodele executeOperation, executa operatia ce doreste a fi executata, iar printResult, printeaza rezultatul final pe ecranul calculatorului.

Structurile de date folosite: ArrayList. Poate o alegere mai buna ar fi fost un HashMap, pentru a evita anumite cazuri particulare in operatia de inmultire spre exemplu. (in urma inmultirii monom cu monom, pot rezulta anumite monoame de grad egal, ce s-ar dori adunate intre ele, cu un HashMap, aceasta operatiune ar fi fost mai usoara decat cu un ArrayList, si mult mai eficienta).

Algoritmii folositi sunt efectiv algoritmii de adunare, scadere, inmultire, impartire, derivare, integrare pe polinoame. Acestia vor fi detaliati mai mult in capitolul de implementare.

Algoritmul de impartire este descris de pseudocodul de mai jos.

Text

Description automatically generated

# Implementare

La capitolul anterior am spus despre fiecare clasa in parte, despre campurile si metodele pe care le contin. In acest capitol vom intra in detaliu in ceea ce priveste functionalitatea fiecarei metode, cat si a interfetei utilizator.

Metodele ce efectueaza operatiile din clasa Monomial, sunt foarte simple. Acestea doar efectueaza operatiile de adunare, scadere, inmultire, derivare, integrare pe monoame.

1. Metodele din clasa Polynomial:

* Adunarea:

Metoda de adunare, efectueaza adunarea a doua polinoame. Unul este primit ca si parametrul, iar celelalt este obiectul pe care se apeleaza adunarea, iar rezultatul este returnat sub forma de polinom. Intai creem un polinom ce va servi drept rezultat.

Apoi parcurgem cele 2 liste cu un for in for pentru a aduna monoamele de grad egal. Intervine cazul in care in primul sau in al doilea polinom, exista monoame ce nu au grad egal cu niciunul din monoamele celuilalt polinom. In acest caz, aceste monoame trebuie adaugate in rezultat fara nicio schimbare.

Pentru a acoperi acest caz, am verificat pentru fiecare monom din A, daca exista monoame egale, iar daca exista se executa adunarea, daca nu exista adaugam monomul din A analizat. Am executat aceasta analiza si pe polinomul B.

De asemenea am adaugat in rezultat doar monoamele nenule.

La final am sortat lista de monoame, pentru a fi afisate in ordinea gradelor. Sortarea a fost efectuata cu o functie lamda.

Ultima verificare este facuta pentru a acoperi cazul in care adunarea da rezultatul 0, polinomul nul.

* Scaderea:

Scaderea este identica cu adunarea, doar ca in loc sa facem adunare pe fiecare monom, vom efectua operatia de scadere.

* Inmultirea:

In cadrula cestei metode, am declarat initial polinomul rezultat, ce va urma a fi returnat, si unde vom stoca toate rezultatele.

Am parcurs cu un for in for cele doua polinoame si am inmultit monom cu monom. In cazul in care inmultirea a dat un monom diferit de 0 trecem la a doua etapa, cea a verificarii daca un monom cu acest grad exista deja in resultat, in cazul in care exista, vom aduna monomul rezultat la cel ce exista deja, in caz contrar in adaugam pur si simplu la polinomul rezultat. Dar daca adunarea celor doua monoame va da 0, monomul este scos cu totul din polinomul rezultat pentru a nu avea termeni nuli.

Dupa efectuarea operatiei, sortam lista in functie de grade, si mai facem o ultima verificare pentru cazul in care rezultatul este un polinom nul.

* Impartire:

Impartirea a fost cea mai complexa parte in ceea ce priveste operatiile pe polinoame. Am implementat metoda dupa algoritmul prezentat in capitorul anterior.

Intai am verificat daca impartitorul este 0. In caz afirmativ, vom returna null, pentru a afisa un mesaj de eroare.

Apoi am sortat cele 2 polinoame pentru a fi sigur ca avem ordinea corecta pentru executia algoritmului de impartire.

Apoi am verificat cazul in care impartitorul are gradul mai mare decat deimpartitul. In acest caz, catul este 0, iar restul este chiar deimpartitul.

Mai departe am implementat algoritmul de impartire.

Primul pas a fost de a calcu primul monom din cat ca fiind raportul dintre monomul dominant din deimpartit si monomul dominant din impartitor. Apoi am calculat restul cu ajutorul operatiilor de inmultire si scadere ce au fost implementate anterior.

Mai departe cat timp gradul restului este mai mare decat gradul impartitorului si restul este diferit de 0. Am calculat fiecare monom din cat si l-am adaugat la rezultat, dupa care am calculat noul rest si tot asa pana cand am calculat rezultatul final.

La final am returnat lista formata din polinomul cat si polinomul rest.

* Derivare:

In derivare, parcurgem polinomul ce doreste a fi derivat, si derivam monom cu monom. La final sortam dupa gradele monoamelor si verificam daca rezultatul este polinomul nul.

* Integrare:

Integrarea este la fel cu derivarea, doar ca in loc de a deriva fiecare monom, il integram.

1. Metodele din clasa CalculatorController:

* executeOperation:

Rolul acestei metode este de a face se ocupa de apelurile metodelor ce efectueaza operatiile de adunare, scadere, inmultire, impartire.

Se realizeaza operatiile pe polinomul ce este deja in memorie si polinomul nou citit, in functie de ce operatie s-a ales. Dupa care se sterge polinomul din memorie si se inlocuieste cu polinomul rezultat.

Din cauza unei erori de proiectare, modelul acestei aplicatii este un ArrayList<Polynomial>, dar putea fi un simplu Polynomial.

* printResult:

Aceasta metoda se ocupa cu printarea efectiva a rezultatului pe ecranul calculatorului.

* Listenerii de pe butoane:

Am realizat anumite verificari pentru fiecare din metodele din listeneri, pentru a permite un format specific al polinomului ce doreste a fi introdus, asemenea unui calculator obisnuit de numere din Windows.

* Listenerii pentru stergerea caracterelor introduse:

Acestia sterg atat caracterele vizibile, cat si polinoamele ce au fost introduse in memorie inainte.

* Listenerii pentru butoanele de operatii:

In caz ca avem un polinom valid citit, avem doua scenarii. Primul este acela in care, acest polinom este primul polinom introdus in calculator. Daca este primul, stocam polinomul in variabila instanta model si setam operatia in functie de ce operatie a fost selectata.

Daca polinomul introdus este al doiea, efectual operatia, dupa care printam rezultatul. Efectuarea operatiei se realizeaza prin apelarea metodei executeOperation.

Pentru derivare

* Listener pentru butonul de egal:

Efectueaza ultima operatie stocata si afiseaza rezultatul pe ecran.

Metodele din clasa Calculator:

* readData:

Aceasta metoda se ocupa cu citirea si conversia string-ului introdus de utilizator intr-un model de date creat de noi, mai exact intr-un polinom.

Pentru despartirea input-ului in monoame am folosit o expresie regex ("-?(\\d\*)?[a-zA-Z]?(\\^\\d\*)?"). Acesta ne va despartii input-ul in monoame, iar in functie de ce monom este il vom converti si adauga in lista noastra de monoame.

Evaluam mai multe cazuri (constanta, variabila simpla, variabila + coeficient, variabila + putere, -variabila simpla sau coeficient + variabila + putere). In functie de unul din cazurile acestea vom convertii monomul respectiv.

Dupa conversie, polinomul nostru trece printr-o serie de verificari. Trebuie sa fim siguri ca nu am introdus din greseala aceasi putere de mai multe ori, in cazul in care avem monoame de grade egale, vom aduna coeficientii exact ca in cadrul operatiei de inmultire.

Apoi sortam monoamele in functie de grad si ne asiguram ca nu avem o lista vida, in caz ca lista este vida, inseamna ca returnam polinomul 0.

Graphical user interface

Description automatically generatedInterfata utilizator arata in felul urmator:

In campul de text vor fi introduse polinoamele cu ajutorul butoanelor (asemeni unui calculator normal de numere).

Sus sunt butoane pentru stergere.

In dreapta sunt butoanele pentru operatii.

# Rezultate

Am testat, atat operatiile din clasa Monomial, cat si cele din clasa Polynomial.

Am facut teste simple – in care aplic operatia pentru 2 monoame/polinoame.

Apoi am facut teste pentru cazuri particulare: impartire cu 0, inmultire cu 0, adunare cu 0, adunare P cu -P.

# Concluzii

Am invatat ca o buna structurare si proiectare a modelului de date este esentiala pentru evitarea unor complicatii in interiorul aplicatiei.

De asemenea am invatat despre managementul timpului.

Am creat aplicatia astfel incat sa poata fi scalabila, adica sa se poata efectua multiple operatii pe mai multe polinoame. O idee de imbunatatire, ar fi imbunatatirea aspectului aplicatiei, si adaugarea unor noi operatii.

De asemenea se poate adauga si inserarea directa a input-ului, de la tastatura, fara folosirea butoanelor. In acest caz trebuie adaugate validari suplimentare pentru formatul input-ului.

# Bibliografie

Materiale de curs.