CALCULATOR DE POLINOAME

Mihaila Gabriel

grupa 30229

UTCN

Prof indrumator: Marcel Antal

**Cuprins**

1. Assignment objective
2. Problem analysis and use case diagram
   1. Problem analysis
   2. Use case diagram
3. Design
   1. GUI
   2. Design decisions
   3. Packages
   4. Data structures
   5. UML Diagram
4. Implementation
   1. Main
   2. Monom
   3. Polinom
   4. DividePolinom
   5. ExceptieImpartire
   6. Controller
   7. View
5. Results
6. Conclusions
7. Bibliography

1. Assignment objective

Obiectivul acestui proiect este de a realiza o aplicatie de calcul polinomial cu o interfata interactiva a utilizatorului prin care acestuia ii este permis sa efectueze sase operatii polinomiale de baza: adunare, scadere, multiplicare, divizare, derivare si integrare. Utilizatorul ar trebui sa poata introduce singur aceste polinoame si sa observe cee ace se afiseaza. In viziunea mea, scopul acestor teme este de a permite studentilor sa practice conceptele Java si POO prin construirea aplicatiei mentionate, care necesita unele abilitati pe care studentii le dezvolta in timp ce fac acest lucru.

1. Problem analysis and use case diagram
   1. Problem analysis

Aceasta problema parea simpla la prima vedere, deoarece operatiile polinomiale obisnuite nu sunt o sarcina dificila in matematica, dar treptat am realizat ca, in programare, este necesara o abordare eficienta si modulara pentru a rezolva aceasta problema. M-am gandit mai intai la doua clase principale care ar modela obiectele: clasa Monom si clasa Polinom. Cele doua sunt discutate mai detaliat in capitolele viitoare. In ceea ce priveste interfata cu utilizatorul, am considerat ca este nevoie de o interfata grafica simpla si minimalista pentru a fi usor de inteles de catre orice utilizator.

* 1. Use case diagram

In randurile urmatoare, am analizat cateva use cases pe care le consider relevante pentru aplicatie:

Titlu: Efectuarea unei operatii intre doua polinoame.

Descriere: Utilizatorul introduce doua polinoame si alege operatia pe care doreste sa o efectueze din urmatoarea lista: adunare, scadere, inmultire, impartire, derivare, integrare. Toate aceste operatii se fac printr-un simplu clik pe butonul cu operatia corespunzatoare.

Actor principal: Oricare utilizator al aplicatiei.

Conditii prealabile: Utilizatorul este obligat sa introduca doua polinoame, fara spatii intre monoame si in ordinea descrescatoare a gradelor monoamelor.

Scenariul meu de succes:

1. Utilizatorul introduce doua polinoame corecte;

2. Utilizatorul face click pe operatia dorita;

3. Cele doua polinoame sunt citite si interpretate corect;

4. Se obtine si se afiseaza rezultatul corect;

Frecventa utilizarii: Foarte des, aceasta fiind functionalitatea de baza a aplicatiei.

Titlu: Efectuarea operatiei de derivare sau de integrare.

Descriere: Acest use case este inclus in primul, dar difera de faptul ca nu sunt necesare doua polinoame, deoarece operatia de derivare si integrare utilizeaza doar primul polinom, cu toate ca utilizatorul are si posibilitattea de a face aceste operatii la doua polinoame simultan.

Actor principal: Oricare utilizator al aplicatiei.

Conditii prealabile: Utilizatorul trebuie sa introduca unul sau doua polinoame, fara spatii intre monoame si in ordinea descrescatoare a gradelor monoamelor. Datele pot fi introduse atat in prima casuta, corespunzatoare primului polinom, cat si in cea de a doua casuta, corespunzatoare celui de-al doilea polinom.

Scenariul meu de succes:

1. Utilizatorul introduce un polinom corect.

2. Utilizatorul face click pe derivare /integrare.

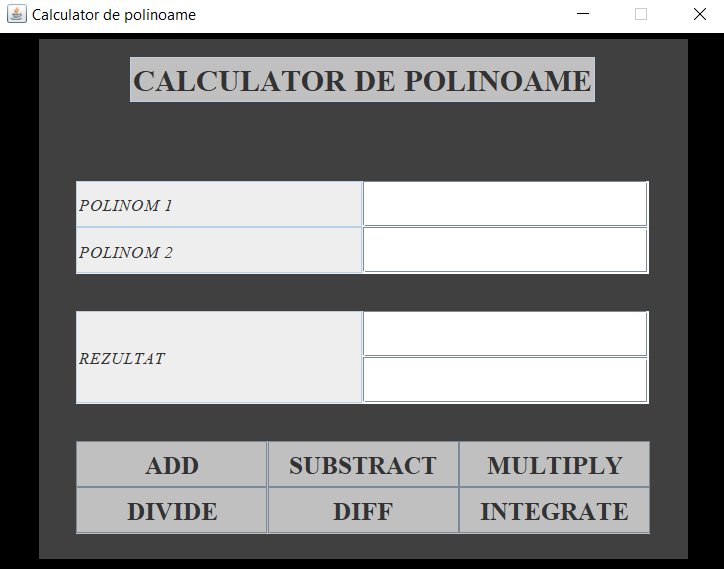
3. Polinomul este citit si interpretat corect.

4. Se obtine si se afiseaza rezultatul corect.

Frecventa utilizarii: Foarte des.

1. Design
   1. GUI

Interfata grafica este simpla si intuitiva, deoarece consider ca utilizatorul ar trebui sa-si poata da seama cum sa foloseasca aplicatia. Atasat mai jos este o captura de ecran a interfetei cu utilizatorul si explicatia ei imediat dupa.



Fereastra este un obiect JFrame simplu de 600 x 470 care contine 7 JPanel-uri, dupa cum urmeaza:

- Primul JPanel (titlePanel) contine titlul aplicatiei noastre si anume “Calculator de polinoame” – cu FlowLayout..

- Al doilea JPanel (inputPanel) contine reprezinta zona unde se pot introduce cele 2 polinoame prin intermediul carora vor fi efectuate operatiile – cu GridLayout.

- Al treilea JPanel (outputPanel) cuprinde JTextField-ul pentru rezultat - GridLayout.

- Al patrulea JPanel (resultArea) cuprinde cele 2 JTextField-uri unde vor fi afisate rezultatele – cu GridLayout.

- Al cincelea JPanel (buttonPanel) reprezinta zona de butoane – cu GridLayout.

- Al saselea si al saptelea JPanel (originPanel1 si originPanel2) au rolul de a unifica celelate 5 JPanel-uri si de a da textura GUI-ului prin alternanta de culoare negru-gri.

* 1. Design decisions

Am ales sa dezvolt aplicatia astfel, deoarece am vrut ca experienta utilizatorului sa fie cat mai placuta. Rezultatul operatiei este dat prin apasarea butonului de functionare, fara a fi nevoie sa se apese din nou pe alte butoane, spre deosebire de a avea un buton de egal. Mai mult decat atat, aplicatia are nevoie de cele doua polinoame (mai putin in cazul calculului derivatei si a integralei) de intrare si efectueaza calculele pe cele doua indiferent de numarul de apasari ale unui buton, spre deosebire de situatia in care, spre exemplu ar fi proiectat un singur camp de text pentru introducere si apasarea butonului de adaugare (ex: prin apasarea de cinci ori a butonului ar insemna ca polinomul de intrare sa fie adaugat de cinci ori). In viziunea mea, aceasta abordare este mai usor de inteles si implementat ca un exercitiu de invatare ce se comporta ca un sistem combinational, in loc de unul secvential. Am decis sa ma concentrez pe calitatea programarii din spatele aplicatiei si aceasta implementare mi-a permis sa scriu un cod Java usor de inteles. De asemenea, am decis sa permit intrari fara spatii, deoarece expresiile pot deveni greu de citit avand multe spatii. Nu exista nicio limita pentru gradul polinoamelor de intrare, dar monoamele lor trebuie sa fie in ordinea descrescatoare a gradelor lor.

* 1. Packages

Singurele pachete create de mine sunt pachetele de baza, pachetul principal, care contine toate clasele cu codul de implementare si pachetul de testare cu clasa de testare a aplicatiei. Am considerat ca folosirea de pachete pentru modelul de proiectare MVC este redundanta, intrucat am avut un singur Controller si un singur View.

* 1. Data structures

Unele structuri de date mai notabile pe care le-am utilizat sunt ArrayLists pentru listele de monoame ale unui polinom. Alte structuri de date sunt doar valorile primitive.

* 1. UML Diagram

Urmatoarea imagine reprezinta diagrama UML a proiectului:

A screen shot of a computer

Description automatically generated with medium confidence

1. Implementation
   1. Main

Clasa Main este clasa generata standard cu metoda principala. Singurul scop, din punctul meu de vedere al implementarii, este sa instantieze cele doua clase, View si Controller, si sa se faca legatura dintre ele.

Metode implementate:

public static void main(String args[]) – metoda main apelata la executarea proiectului.

* 1. Monom

Clasa Monom este prima clasa care proiecteaza un obiect, un monom. Scopul sau este de a detine un singur monom. Nu am implementat operatii cu monoame in aceasta clasa, deoarece le-am considerat banale si, prin urmare, redundante. Aceasta clasa respecta principiul incapsularii si toate campurile sunt private. Accesul la acestea este limitat prin intermediul getters si setters.

Campuri:

* private Double coeficient: coeficientul monomului
* private int putere: puterea monomului

Constructori:

* public Monom(Double coeficient, int putere): instantiaza un nou obiect de Monom cu un coeficient si o putere date ca parametru

Metode implementate:

* public Double getCoeficient: getter pentru coeficient
* public void setCoeficient(Double coeficient): setter pentru coeficient
* public int getPutere(): getter pentru putere
* public void setPutere(int putere): setter pentru putere
  1. Polinom

Clasa Polinom este a doua clasa care proiecteaza un obiect, un polinom. Un polinom este reprezentat ca un ArrayList de obiecte de tip Monom. Aceasta este, fara indoiala, cea mai importanta clasa a proiectului, deoarece implementeaza toate operatiunile pe polinoame. Aceasta clasa, ca si clasa Monom, respecta principiul incapsularii si toate campurile sunt private. Accesul la acestea este limitat prin intermediul getters si setters.

Campuri:

* private ArrayList<Monom> monom: lista de monoame
* private int gradPolinom: gradul maxim al monoamelor ce da gradul polinomului

Constructori:

* public Polinom(ArrayList<Monom> monom): instantiaza un obiect de tip Polinom caruia i se atribuie o lista data de monoame si se determina si gradul polinomului respectiv.

Metode implementate:

* public ArrayList<Monom> add(Polinom polynom): adauga un polinom la un alt polinom prin succesiunea de monoame avand grade identice si returneaza lista de monoame astfel rezultata.
* public ArrayList<Monom> sub(Polinom polynom): scade un polinom la un alt polinom prin succesiunea de monoame avand gradul identic si returneaza lista de monoame astfel rezultata.
* public ArrayList<Monom> sub(Polinom polynom): inmulteste un polinom la un alt polinom si returneaza lista de monoame astfel rezultata.
* public ArrayList<Monom> diff(): determina polinomul derivat din polinomul dat ca obiect al metodei diff si returneaza lista de monoame astfel rezultata.
* public ArrayList<Monom> integrate(): determina polinomul integral din polinomul dat ca obiect al metodei integrate si returneaza lista de monoame astfel rezultata.
* public DividePolinom divide(Polinom impartitor): imparte un polinom deimpartit la un al polinom impartitor si returneaza un obiect de tipul DividePolinom care retina practic doua polinoame distincte, reprezentand catul si restul impartirii.
* public String toString(): returneaza polinomul rezultat in urma uneia dintre operatii ca un String.
* gettere si settere pentru campurile private
  1. DividePolinom

Aceasta clasa a fost implementata exclusiv pentru operatia de impartire. Contine doua campuri de obiecte de tip Polinom, reprezentand catul si restul impartirii.

Campuri:

* private Polinom cat: retine catul
* private Polinom rest: retine restul

Constructori:

* public DividePolinom(Polinom cat, Polinom rest): instantiaza un nou obiect de tipul DividePolinom ce retine doua polinoame reprezentand catul si restul operatiei de impartire.

Metode implementate:

-gettere si settere pentru campurile private.

4.5. ExceptieImpartire

Aceasta clasa extinde clasa Exception si are ca scop depistarea impartirii cu 0 la polinoame, excpetie ce este prinsa intr-un try catch ce determina adnotarea utilizatorului ca impartirea nu este corecta.

4.6. Controller

Aceasta clasa corespunde controlerului in modelul de proiectare MVC si detine logica din spatele aplicatiei. Controller-ul contine algoritmul de analiza polinomiala si seteaza actiunile pentru butoanele din interfata grafica.

Campuri:

* private View gui: camp al interfetei grafice

Constructori:

* public Controller(View gui): instantiaza un nou Controller care primeste o interfata grafica gui

Metode implementate:

* public ArrayList<Monom> getMonoame(String polynom): parseaza String-ul primit de catre utilizator si il grupeaza pe 4 grupuri: primul grup daca are semnul minus sau plus in fata, al doilea grup daca are un coeficient, al patrulea grup daca are x^ si ultimul grup daca respectivul polinom are o putere data. Se va construi astfel un monom cu coeficientul si puterea respectiva. Functia va returna o lista de astfel de monoame create.
* private void OperationsListeners(): sunt setate actiunile de pe butoanele din interfata grafica.

4.7. View

Clasa View, dupa cum ii mentioneaza si numele, implementeaza partea de interfata a modelului de proiectare MVC.

Campuri: JFrame-ul pentru initializarea ferestrei, JPanel-uri pentru plasarea ordonata a JTextField-urilor si a butoanelor, precum si un camp de nume pentru interfata grafica.

Constructori:

* public View(String name): creeaza o instanta a clasei View si efectueaza operatiile pentru crearea si atribuirea de JPanel-uri si JFrame.

Metode implementate:

* gettere si settere pentru campurile private.

1. Results

Pentru a verifica corectitudinea rezultatelor, am creat o clasa JTest in pachetul de testare, care utilizeaza JUnit si are 6 teste, cate unul pentru fiecare operatie. Am creat doua polinoame care sunt utilizate pentru test si un polinom rezultat, precum si un dividePolinom in cazul operatiei de impartire. Toate testele au fost rulate cu succes.

1. Conclusions

Consider ca acest assignment m-a invatat multe despre impartirea codului in metode in asa fel incat nicio metoda sa nu contina o cantitate imensa de cod. De asemenea, am castigat si putina experienta in ceea ce priveste codarea in Java. Am invatat cum sa folosesc expresii regulat si cum sa configurez o interfata mai buna, lucru pe care l-am tot evitat.

In ceea ce priveste o dezvoltare viitoare a aplicatiei, cred ca este nevoie de un interpretare mai buna a polinomului de intrare, care sa accepte atat spatii intre monoame, cat si sa sorteze monoamele, astfel incat sa nu fie necesara introducerea polinomului intr-o anumita ordine a monommelor. O alta dezvoltare viitoare ar fi implementarea unui mod de utilizare a rezultatului ca polinom fara a fi necesar sa fie rescris manual, salvand practic polinoamele in memorie. Ar mai putea fi dezvoltate si operatii mai complexe, cum ar fi gasirea radacinilor.

1. Bibliografie

* <https://www.symbolab.com/>
* <https://stackoverflow.com/>
* <https://sites.google.com/site/videomeditatii/clase-liceale-9-12/clasa-a-12-a-materii-de-studiu/programa-scolara-pentru-matematica-clasa-a-xii-a/operatii-cu-polinoame-impartirea-polinoamelor>
* <https://www.wikipedia.org/>