MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Calculator de polinoame

Documentatie

Facultatea: Automatica si Calculatoare

Roman Alexandru | Grupa 30228

CUPRINS

1. Obiectivul acestui calculator

Polinoamele sunt construite din termeni numiți monoame, care sunt alcătuite dintr-o constantă (numită coeficient) înmulțită cu una sau mai multe variabile.

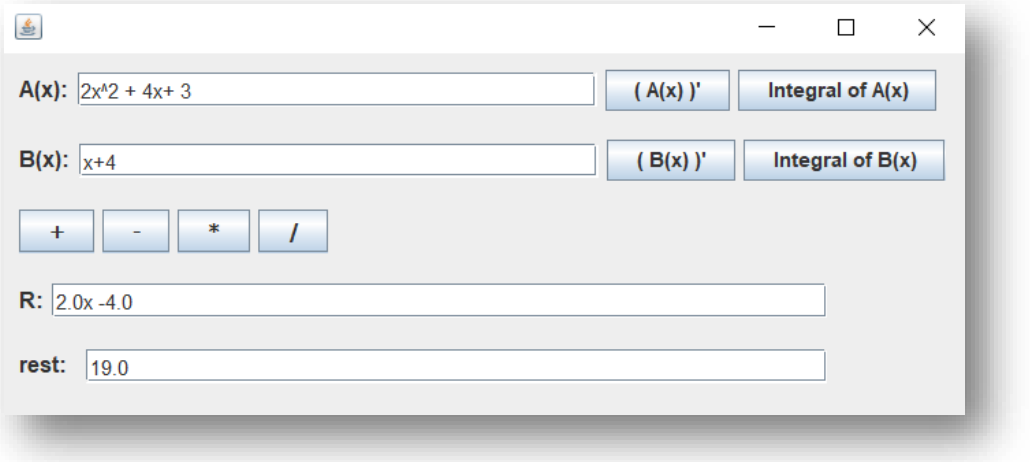
Acest proiect, calculatorul de polinoame, a fost creat pentru a face operatiile pe polinoame mai usoare sau pentru a putea face posibila verificarea operatiilor de catre orice utilizator, folosind o interfata grafica. In interfata grafica putem introduce 2 polinoame si calcula diferite operatii asupra lor, acestea fiind:

- Adunare
- Scadere
- Inmultire
- Impartire

De asemenea putem efectua operatii destinate fiecarui polinom in parte, operatiile acestea fiind:

- Derivare
- Integrare

Polinoamele sunt des intalnite in matematica, tocmai de aceea este necesara o interfata grafica usor de folosit care afiseaza rezultate calculate mult mai rapid si fara efort decat pe putem calcula noi. Interfata grafica a calculatorului o putem vedea in urmatoare imagine:



In interfata de mai sus observam 2 field text-uri in care vom introduce polinoamele numite A(x) si B(x). In dreapta fiecaruia este un buton pentru fiecare operatie: derivare, integrare. Operatia va fii facuta pentru polinomul respectiv si rezultatul si va afisa in textField-ul din dreptul lui “ R : ” . Urmatoarele 4 operatii se vor afisa in textField-ul din dreptul lui “ R : ” respectiv, in dreptul lui “ rest : ” pentru a afisa restul operatiei de impartire.

Polinoamele pot fii introduse in orice ordine ordine. Se pot introduce de la mare la mic, de la mic la mare, sau intr-o ordine amestecata a exponentilor. Rezultatul se va afisa in ordinea descrescatoare a exponentilor. Utilizatorul poate de asemenea sa aleaga daca vrea sa introduca polinoamele folosind caracterul * ca pentru inmutire precum “2 * x + 1” sau fara el. Calculatorul nu va face nic-o diferenta.Rezultatul fa fii afisat sub forma “ ax +b”.

2. Analiza problemei, proiectare

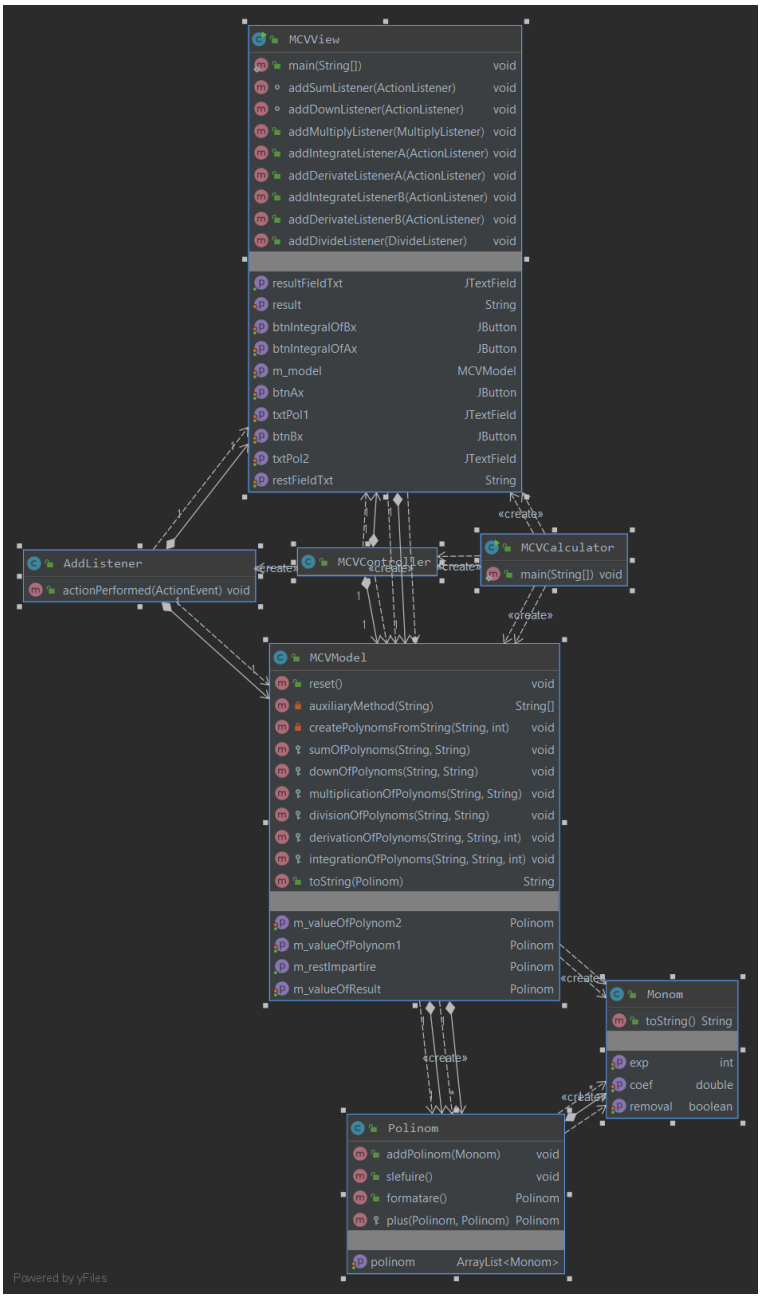
Back-end-ul calculatorului descris mai sus are o structura de tip MCV(Model-View-Controller) , aceasta fiind cea mai optima metoda de a proiecta o astfel de aplicatie. Clasa MCVModel reprezinta in mare cam ceea ce se intampla in spatele interfetei. In aceasta clasa sunt implementate metodele principale care se folosesc de clasa Polinom. In clasa MCVView este descris in cod interfata grafica afisata mai sus, aceasta reprezentand front-end-ul aplicatiei si se foloseste de clasa model pentru a ajuta partea de back-end sa afiseze rezultatele, precum si sa isi ia informatiile necesare pentru functionare. Clasa controller face legatura dintre interfata si ce se

petrece in spatele ei, practic controleaza functionarea calculatorului. Aceste 3 clase sunt apelate de main-ul unei alta clase MCVCalculator. Mai jos, voi explica detaliat aceste clase.

Proiectul a fost creat folosind principiile oop precum mostenirea si Incapsularea(dupa cum voindescrie putin mai jos).

3. Implementare

Dupa cum am mentionat si mai sus, proiectul are o structura de tip Model View Controller. Asta implica patru clase detaliate mai jos.



Imaginea de mai sus este diagrama UML si dependintele dintre clase. Am adaugat doar clasele relevante. Diagrama am realizat-o in Intellij iar claselor lea-am lasat ca si campuri varabilele si metodele. Campurile albastre reprezinta numele claselor, campurile rosii reprezinta metodele, iar cele mov variabilele clasei.

Principiul mostenirii este folosit dupa cum vedea si din diagrama uml, principiul incapsularii, variabilele fiind toate declarate private iar din alte clase sunt folosite prin getteri si setteri. Metodele sunt declarate private sau protected, care se pot vedea doar in interiorul pachetului (in cazul de fata default package).

Clasa View : Aceasta clasa reprezinta partea de front-end a aplicatiei si descrie interfata cu utilizatorul. Ea extinde clasa JFrame, adica practic devine un JFrame iar aici sunt declarate ca si parametrii toate butoanele, precum si JTextField-urile. Implementarea propriu-zisa a interfetei grafice este scrisa in constructor care este apelat din main-ul clasei " MCVCalculator " despre care voi vorbii mai jos. Aceasta clasa are si metodele de setText si GetText pentru a lua / afisa datele despre polinoamele intruduse si polinoamele rezultate. Am dat nume sugestive la butoane sa sa poata fii recunoscute usor. Tot in aceasta clasa se afla metodele care adauga butoanelor ActionListener -urile necesare, adica la fiecare apasare de buton se va intampla un anume lucru(se va executa o secventa de cod). Vom vorbii si despre asta doar putin mai jos.

```
private JTextField txtPol1;
private JTextField txtPol2;
private JTextField resultFieldTxt;
private JTextField restFieldTxt;

private JButton btnAx;
private JButton btnIntegralOfAx;
private JButton btnBx;
private JButton btnIntegralOfBx;

private JButton buttonPlus;
private JButton buttonMinus;
private JButton buttonMultiplication;
private  JButton buttonDivision;

private MCVModel m_model;
```

Clasa Model : aceasta clasa contine metodele care efectueaza operatiile intre polinoame, precum si polinoamele reprezentate prin clasa Polinom, care la randul sau este format dintr-un arrayList de monoame(o voi descrie mai jos).

Aici se afla, de asemenea si metoda de transformare din string in monoame, impreuna cu metoda toString( ) care transforma monoamele in string-uri. Metoda care transforma din string-urile introduse in polinom, imparte dupa semnele '+' si '-', stringul in stringuri mai mici care reprezinta monoamele, iar acestea sunnt verificare folosind mai multe if-uri.Sunt impartite dupa '+', iar apoi dupa '-', rezultand monoame de tip string iar dupa cazuri sunt verificate si impartite in coefient sii exponent. Aceasta metoda este apelata pentru fiecare polinom din fiecare metoda care efectueaza operatii in monoame deoarece acestea primesc polinoamele de tip String ca argumente. Metoda toString() face exact opusul si anume parcurge arrayList ul de monoame(din care este format un polinom) cu un for each si prin verificari multiple cu if-uri creaza stringul rezultat.

Metodele de adunare si scadere au o forma simpla, parcurgandu-se pentru fiecare monom al fiecarui polinom celalalt polinom in intregime. Nu este cea mai eficienta metoda dar este straight-forward.

Inmultirea polinoamelor se realizeaza in metoda "multilpicationOfPolinoms".Aici sunt inmultite fiecare monom cu fiecare si este aplicata peste string-ul rezultat metoda formatare() din clasa polinom care aduna toate string-urile ce au un exponent egal iar apoi le ordonaza descrescator dupa exponent.

```
protected void multiplicationOfPolynoms (String pol1, String pol2){

    createPolynomsFromString(pol1,  number: 0);
    createPolynomsFromString(pol2,  number: 1);

    Polinom polRezultat  = new Polinom();
    ///-------------------------------------------------------construim pozRezultat
    for(Monom e1 : m_valueOfPolynom[0].getPolinom()){
        for (Monom e2 : m_valueOfPolynom[1].getPolinom()){
            double coef = e1.getCoef()*e2.getCoef();
            int exp = e1.getExp() + e2.getExp();

            polRezultat.addPolinom(new Monom(coef, exp));

        }
    }


    m_valueOfResult = polRezultat.formatare();

}
```

Impartirea polinoamelor are exact acelasi concept si are aceiasi pasi cu o impartire a polinoamelor pe foaie. Verificam de cate ori intra primul monom din al doilea polinom in primul monom din primul polinom, il adaugam la rezultat, inmultim ce a rezultat cu al doilea polinom, dupa facem adunare etc . Se folosese pentru adunarea polinoamelor o metoda

numita plus(..) care primeste 2 polinoame si le aduna. Motivul pentru care am creat o noua metoda de anunare(desii este foarte asemanatoare cu prima) este pentru ca am avut nevoie de o metoda care sa primeasca direct 2 polinoame de tip Polinom si nu de tip String. Aceasta problema se mai putea rezolva, desigur, si prin apelare metodei toString pentru polinoame si folosind prima functie dar aceasta rezolvare mi s-a parut cea mai buna. Metoda este straight-forward. La aceasta operatie folosim al patrulea arraylist pentru memorarea restului si este folosit pe langa si metoda de slefuire() din cllasa polinom care “slefuieste polinomul” adica reduce numarul de zecimale la maxim 2.

```
public void slefuire(){

    for( Monom e : polinom)
        e.setCoef(Math.floor(e.getCoef() * 100) / 100);
}
```

Derivarea si Integrarea polinoamelor au fost cele mai usoare operatii de implementat in cod deoarece se creaza un nou polinom cu coeficientii si exponentii modificati corespunzator pentru tipul de operatie, fara alte complicatii sau cazuri.

Clasa MCVController

Clasa controller primeste un obiect de tipul model si un obiect de tipul View si si apeleaza metodele din clasa view prin care le adauga butoanelor ACtionListener-ul potrivit.

```
public void addDerivateListenerA(ActionListener derivate) {

    btnAx.addActionListener(derivate);
}
```

Imaginea de mai sus prezinta metoda addDerivateListener(..) care peimeste ca parametru o clasa care implementeaza ActionListener. (despre Clasele ActionListener vom vorbii mai jos).

Clasa Monom : este o clasa simpla care are rolul de a stoca informatiile unui monom si anume :coeficientul si exponentul.

Clasa Polinom : Clasa Polinom contine un ArrayList de monoame. Prin ea sun reprezentate polinoamele noastre, ca o liista de monoame. Aceasta clasa contine si cele trei functii mentionate si explicate mai sus si anume: adunare(…), slefuire( ) si formatare().

Exista si alte clase in proiect si anume clasele care implementeaza ActionListener. Acestea descriu ce se intampla atunci cand apasam pe un buton, precum butonul + sau *

pentru inmultire. Clase precum : **AddListener, MultiplyListener,** etc. Actiunile sunt descrise in medota implementate actionPerformed precum in imegineade mai jos :

AddListener:

```
@Override
public void actionPerformed(ActionEvent e) {
    String inputPol1;
    String inputPol2;

    inputPol1 = m_view.getTxtPol1().getText();
    inputPol2 = m_view.getTxtPol2().getText();

    m_model.sumOfPolynoms(inputPol1,inputPol2);
    m_view.setResult(m_model.toString(m_model.getM_valueOfResult()));
    m_model.reset();

}
```

Metoda este suprascrisa pentru ca este o metoda implementata de ActionListener. Dupa ce am declarat string urile am luat polinoamele introduse in format string cu metoda getText() din partea de front-end si le-am transmis functiei de adunare din m_model de tipul MCVModel.

Rezultatul a fost salvat in m_model in varialbilele m_ValueOfResult care va fii transmis linia urmatoare medotelor toString(..) si apoi setResult(..).

La sfarsit se reseteaza valorile prin functia reset() pentru urmatoare operatie.Functia reset() se apeleaza dupa fiecare operatie astfel incat sa se goleasca ArrayList-urile pentru a putea primi informatie noua si sa nu se amestece cu polinoamele vechi. Cand vrem sa introducem polinoame noi pur si simplu stergem ce se afla in textField-uri si sa introducem altele, iar daca vrem o alte operatie nu trebuie sa facem nimic decat sa apasam pe butonul operatiei, rezultatul se va actualiza. Legat de rezultat, textField-urile pentru rezultat le-am lasat cu setEditable(true), doar pentru felxibilitatea de utilizare a calculatorului, poate in caz ca utilizatorul are nevoie sa isi noteze un comentariu dupa rezultat in caz ca nu va face alta operatie.

JUNIT: Pentru a ma asigura ca proiectul functioneaza corespunsator, pe langa testarile pe care le-am facut in interfata introducand toate cazurile si tipurile de polinoame si am efectuat la fiecare tip fiecareoperatie in parte, am creat o testare JUnit. In aceasta testare am introdus o operatie care contine mai multe tipuri de monoame pentru fiecare operatie ca sa ma asigur ca e totul functioneaza cum trebuie.

Ca totul sa fie in regula trebuie sa treaca 6/6 teste. Cate 1 pentru fiecare operatie in parte.

4. Rezultate

Calculatorul are o functionalitate corecta, iar fiecare operatie functioneaza in mod optim. Daca utilizatorul introduce polinoamele gresit in fieldText-urile destinate rezultatului nu se va afisa nimic. Cand se vor introduce string-urile corespunzator abia atunci se va afisa rezultatul. Interfata grafica nu e colorata, am lasat-o cat mai simplu pobibil si am structurat-o astefel incat sa-I fie mai usor utilizatorului.

5. Concluzii si posibilitati de dezvoltare ulterioara

Acest proiect m-a ajutat sa-mi aprofundez atat posibilitatea de a mi transpune gandirea matematica in limbaj java / oop, cat si de a vedea unde fac greseli structurale sau de gandire / neatentie in general pentru ca am observat pe ce tipuri de bug-uri pierd vremea si ce ma opreste sa devin mai rapid si sa proiectez ceva cat mai corect. De asemenea m-am familiarizat cu ide-ul Intellij. Codul meu poate fii probabil mai eficient in unele locuri iar metodele puteau fii structurate si in alte moduri. De asemenea, proiectul l-am terminat dupa orimul termen deci pot sa zic ca am observat unele greseli in modul cum imi organizez timpul. Metoda sau partea de cod care mi-a luat cel mai mult timp a fost, cred, cea de despartire a polinomului String in monoame din cauza unor cazuri care nu erau acoperite de algoritm.

Ca si posibilitati de dezvoltare ulterioara acestui polinom i se pot adauga multe tipuri de operatii si moduri de afisare. Probabil o imbunatatire foarte usoara de facut ar fii introducerea unui buton de reset care atunci cand este apasat va golii textField-urile si va face reintroducerea unui alt polinom pentru urmatorul calcul ar fii mai usoara.

Daca as avea niste timp liber as putea sa incerc eventual sa implementez in interfata un grafic in care utilizatorul sa poate observa valorile lui x in axele de coordonate. Chiar daca nu stiu inca cum s-ar proiecta cred ca ar fii o dezvoltare buna a programului.

Cred ca prima dezvoltare care ar trebuii facuta acestui calculator ar fii sa afiseze valoare polinomului pentru un anumit x ceea ce ar avea un nivel de complexitate mai scazut.

Probabil cea mai buna dezvoltare ar fii implementarea unui sistem de calcul pentru a calcula valoarea lui x in cazul de egaliate cu 0. Acest lucru presupune un nivel de complexitate putin mai ridicata pentru un x mai mare decat 2 deoarece cunostintele matematice sunt necesare iar pentru polinoame mari, un algoritm care o executa o metoda pe caz general precum T. Horner ar fii poate mai incet.

Cea mai mare dezvoltare ulterioara ar putea reprezenta un calculator pentru operatii pentu mai mult decat polinoame.