

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

Calculator de polinoame

Documentatie Tema 1

Runcan Nicoleta

Grupa 30229

An 2019-2020

Cuprins

1.Obiectivul temei

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

**Scenariu de utilizare**

**Cazuri de utilizare**

3.Proiectare

**Decizii de proiectare**

**Diagrama de clase**

**Structuri de date**

**Interfete**

**Interfata Utilizator**

4.Implementare

**Clasa Monom**

**Clasa Polinom**

**Clasa Model**

**Clasa Controller**

**Clasa View**

5.Rezultate

6.Concluzii

7.Bibliografie

1.Obiectivul temei

Obiectivul acestei teme a fost de a crea o aplicatie care sa implementeze un sistem de procesare si de lucru cu polinoame. Mai exact aceasta aplicatie este un calculator de polinoame, in care sunt posibile operatii precum adunarea, scaderea, inmultirea si impartirea a doua polinoame, dar de asemenea si operatii exercitate pe un singur polinom, cum ar fi derivarea si integrarea. Polinoamele sunt construite la randul lor din unul sau mai multe monoame, fiecare monom avand specific cate un coeficient de tip real si cate un exponent intreg. De asemenea un obiectiv secundar, pe care aceasta tema il urmareste este acomodarea studentilor cu pasii care trebuie urmati pentru rezolvarea unei si probleme pentru atingerea obiectivului principal, si anume obtinerea rezultatului dorit.Acesti pasii implica analizarea problemei si dezvoltarea solutiilor posibile, impartirea pe clase, dezvoltarea algoritmilor, implementarea solutiei alese si testarea. De asemenea, aceasta tema a urmarit si familiarizarea studentilor cu proiectarea unei interfete grafice, care poate fin utilizata usor si confortabil de catre orice utilizator.

2. Analiza problemei, modelare, scenarii, cazuri de utilizare:

Un polinom reprezinta o inlantuire de monoame, care la randul lor sunt alcatuite dintr-un coeficient si un exponent. Exista o multidunie de metode prin care ar putea fi alcatuit un calculator de polinoame, pornind de la forma pe care trebuie sa o aiba polinoamele pe care o sa le introduca utilizatorul asupra carora se vor exercita operatiile,modul in care acesta isi va alege operatia pe care acesta sa o efectueze, si pana la forma in care rezultatul obtinut in urma acestor operatii ii va fi afisat utilizatorului, iar metoda aleasa trebuie sa fie una cat mai eficienta.

**Scenariu de utilizare:**

* Utilizatorul introduce in doua TextField-uri disponibile in interfata grafica, polinoamelesau polinomul pentru care doreste sa efectueze operatiile puse la dispozitie sub forma unui String.
* Dupa introducerea polinoamelor inntr-o forma valida, acesta va avea la dispozitie 6 butoane prin care sa aleaga operatia pe care o doreste
* Daca datele introduce de utilizator nu au o forma valida, programul va arunca o exceptie predefinita
* Daca datele introduse sunt valide, se va afisa rezultatul tot intr-un TextField asezat in partea de jos a interfetei, iar in cazul impartiri sunt disponibile doua TextField-uri, unul pentru afisarea catului, iar celalalt pentru afisarea restulu

**Cazuri de utilizare:**

Programul poate fi utilizat pentru polinoame de orice grad care au coeficientul real si exponentul intreg, doar ca monoamele care intra in component acestora trebuie sa aiba aceasta forma “CoeficientX^Exponent”, chiar daca vorbim despre monoame cu exponentul 0 sau 1, sau coeficient 1.

3.Proiectare

**Decizii de proiectare:**

Pentru proiectarea acestei aplicatii am ales modelul MVC, care presupune izolarea logicii interne fata de considerentele interfetei cu utilizatorul. In clasa Model sunt manipulare operatiile

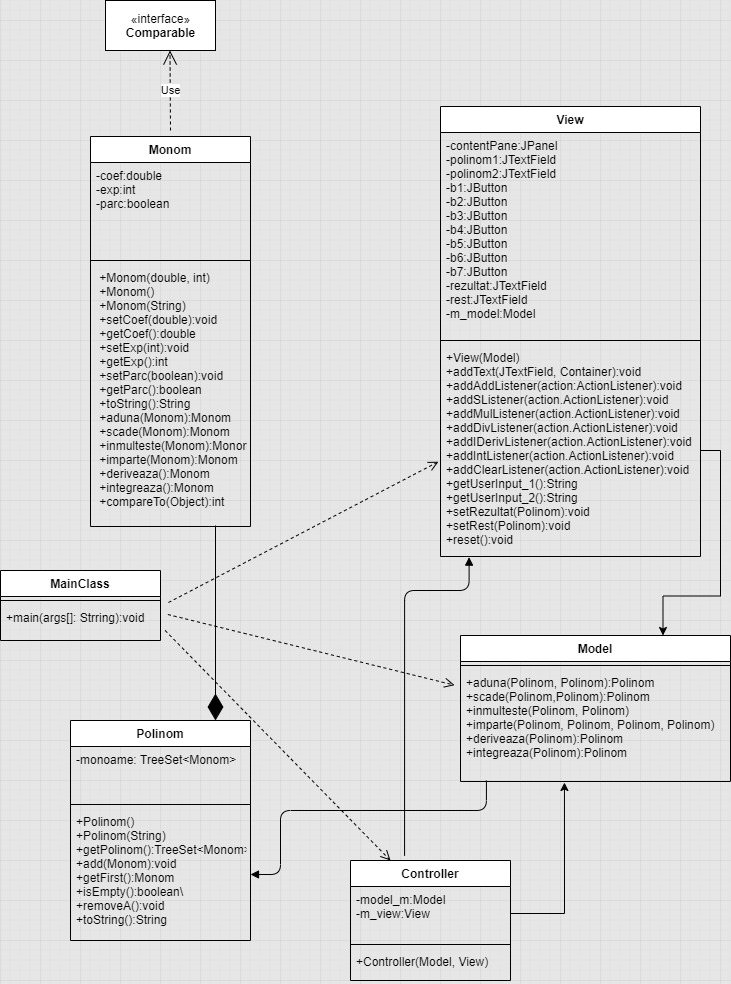


implementate in acest program, pentru o buna intelegere a programului; in clasa Controller se controleaza accesul la aplicatie, aici facandu-se legatura intre partea vizibila utilizatorului si programul care ruleaza in spatelele aceste interfete; iar clasa View corespunde reprezentarii grafice, sau mai bine zis exprimarea ultimei forme a datelor.

**Structuri de date:**

Structura de date pe care am folosit-o este un TreeSet de monoame, care ne faciliteaza munca prin faptul ca un TreeSet nu poate contine contine elemente duplicate, in cazul programului meu, acest TreeSet nu va contine doua monoame cu acelasi exponent si de asemena, aceasta structura ne usureaza munca de sortare a monoamelor in ordinea descrescatoare a exponentilor. TreeSet-ul va fi o variabila instant a clasi Polinom. Clasa Monom este o clasa separate care are ca si variabile instant un coeficient de tip double, un exponent de tip intreg, dar de asemenea si o booleana parc, pe care am folosit-o pentru implementarea operatiilor de adunare si scadere.

**Diagrama de clase:**

****

**Interfete:**

Nu a fost definite nicio interfata noua, singura interfata folosita a fost una predefinita:**Comparable**. Aceasta este implementata de clasa Monom, in care a a fost implementata si metoda **compareTo,** cu ajutorul careia

se vor sorta monoamele in ordinea descrescatoare a exponentilor.

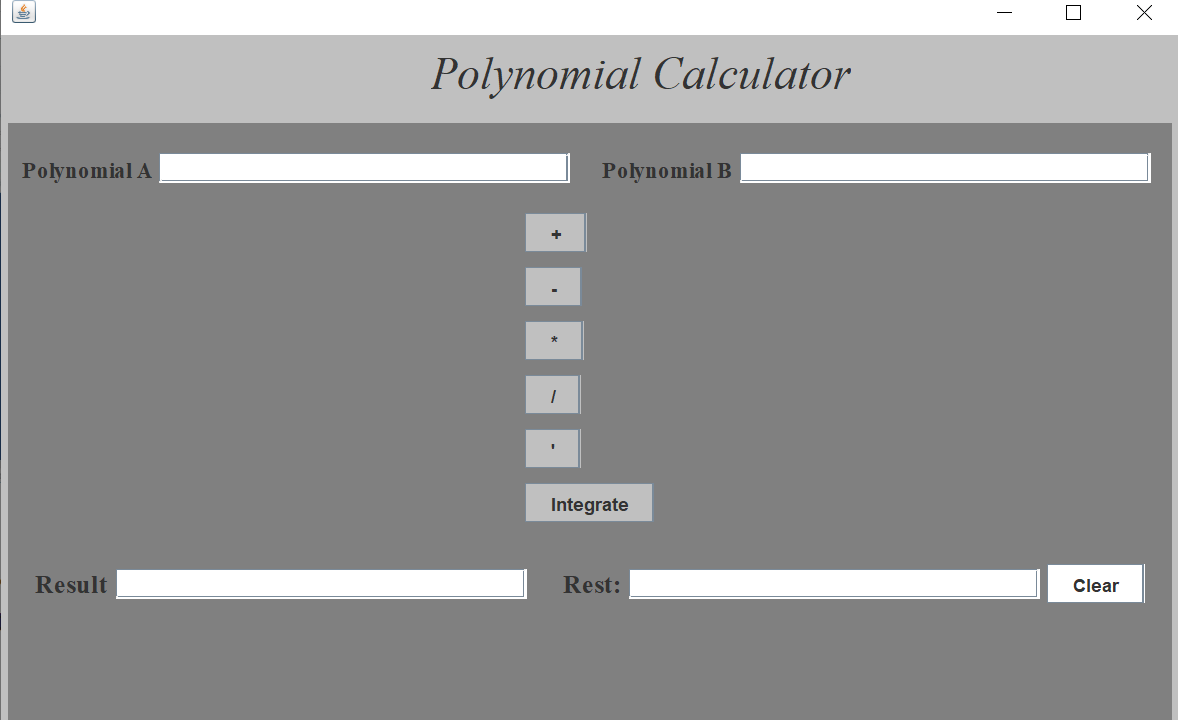
**Interfata Utilizator :**

Interfața grafică sau Graphical User Interface(GUI) este o [interfață cu utilizatorul](https://ro.wikipedia.org/wiki/Interfa%C8%9B%C4%83_cu_utilizatorul) bazată pe un sistem de afișaj ce utilizează elemente grafice. Interfața grafică este situata funcțional între utilizator și implementarea care se afla in spatele acestei interfete. Folosim o interfata grafica usor de manipulat de catre utilizator cu scopul de a putea fi folosit acest calculator de polinoame si de persoane non-specializate,

Interfata grafica contine urmatoarele elemente:

* *Un JFrame-*care reprezinta mai exact, fereastra propriu-zisa, in care se adauga toate celelalte elemente, care la apasarea butonului “x” din partea sus a ferestei, se va inchide.
* *4 TextField-uri-*doua dintre ele fiind folosite pentru introducerea celor doua polinoame initiale, iar celelate doua sunt folosita pentru afisarea rezultatelor, cel de-al doilea din partea de jos, fiind folosit doar pentru afisarea restului in cazul operatiei de impartire
* *7 Butoane*-6 dintre ele reprezentant functionalitatea celor sase operatii ale calculatorului(adunare, scadere, inmultire, impartire, derivare si integrare), iar al saptelea este un buton de “Clear” care este folosit pentru curatarea celor 4 TextField-uri la apasarea lui
* *JLabel-uri-*sunt folosite pe scop de etichete, pentru este efectiv o etichetă, care poate fi titlul, informații, indicații etc. care pot ajuta utilizatorul să folosească progamul.

Pentru fiecare button, s-a făcut o metodă nouă in clasa care implică ActionEvent. Aceasta clasa, de fiecare dată când are loc o acțiune la un buton, adica a fost apăsat, ActionEvent-ul transmite informația la ActionListener care așteaptă astfel de informații. Apoi, au loc evenimentele ce se afla în metodă respectiva ( în general: se declară un Polinom nou la care se atribuie rezultatul operației alese ).

****

4 Implementare:

In acest capitol se vor descrie clasele si metodele importante, componente a acestor clase, cu logica din spatele lor.

**Clasa Monom:**

Aceasta clasa este absolut necesara acestui program, fiecare instanta a ei reprezentand un monom care va aparea in TreeSet-ul ce formeaza polinomul. Aceasta clasa contine 3 contructori: un constructor gol, unul care primeste ca si parametru un string, de forma unui Monom, care va fi separate cu ajutorul unei expresii regulate Regex, in cele doua variabile instanta importante a acestei clase:coeficient si exponent. Aceasta separare are loc astfel: datorita faptului ca structura unui monom este urmatoarea “coefX^exp” se face un split pe string-ul dat ca si parametru contructorului, in functie de urmatorul separator “X^”. Apoi se va face conversia celor doua elemente rezultate in urma separarii: primul elemnt va fi convertit intr-un double,, iar cel de-al doilea intr-un intreg. Cel de-al treilea constructor este unul care primeste 2 parametrii, un double, si un intreg, care initializeaza cele doua variabile instant importante ale clasei

Tot aici este implementata o forma de baza a celor 6 operatii pentru doua monoame, sau pentru un singur monom(derivare, integrare) ( mentionate si in paginile anterioare) pe scheletul carora in clasa Model va fi implementata varianta finala a acestor operatii pentru a putea fi functionale pentru o inlantuire de monoame, adica pe un polinom mai complex .

O alta metoda din aceasta clasa este metoda toString(), care va fi suprascrisa in clasa Polinom, prin care se reuseste o afisare frumoasa a rezzultatelor operatiilor implementate in acest program.

Tot o metoda importanta implementata in aceasta clasa este metoda compareTo(), care a fost absolut necesara de implementat, odata cu implementarea interfetei “Comparable” de catre aceasta clasa. Prin aceasta metoda definim o forma de ordonare a obiectelor de tip monoam, in functie de exponentul corespunzator acestora.

**Clasa Polinom:**

Aceasta clasa are ca si variabila instanta un TreeSet de monoame si 2 constructori importanti. Unul gol, iar cel de-al doilea cu un parametru de tip String, asemenea clasei Monom. Si in acest constructor se aplica un split asupra String-ului. Care este o inlantuire de monoame conectate prim operatorul “+”. Datorita acestei structuri a polinoamelor, pentru inceput, in urma faptului ca unii coeficienti ar putea sa fie si negativi, iar in cazul acesta operatorul “+“ va lipsi dintre acest monom si precedentul lui, se va face o inlocuire a semnului “-“ cu “+-“, iar astfel split-ul se va face in functie de separatorul “+”. In urma acestei separarii, vor rezulta string-uri de forma monoamelor, apelandu-se pentru fiecare dintre ele constructorul clasei Monom cu parametrul de tip String.

O sa amintesc aici de o singura metoda importanta: toString(), care o suprascrie pe cea din clasa Monom, care, dupa cum zis si mai sus, are rolul de a furniza o afisare frumoasa a rezultatelor.

**Clasa Model:**

Aceasta clasa nu contine nicio variabila instant si niciun constructoe, ci doar implementarea celor sase metode pentru efectuarea celor 6 operatii puse la dispozitie de catre calculator.

*Metoda Aduna-*aceasta metoda realizeaza suma dintre doua polinoame, pe care le primeste ca si argumente. Pentru inceput se parcurg cele doua TreeSet-uri de la inceput, verificandu-se gradele monoamelor, daca acestea sunt egale, se va apela metoda de adunare a doua monoame cu acelasi exponent. Tot in clasa Monom variabila instanta parc va fi setata la “true”(adica aceste monoame vor fi setate ca si parcurse), astfel, la o adoua parcurgere a celor doua polinoame, se vor adauga doar elemtele care nu au indeplinit contitia de exponent comun cu un alt monom.

*Metoda Scade:-*realizeaza scaderea dintre doua polinoame. Primate ca si argument. In implementarea acestei metoda am urmat intocmai pasii pe care i-am urmat si in implementarea adunarii.

*Metoda Inmulteste-*realizeaza inmultirea dintre doua polinoame primate ca si argument. Implementarea acestei metoda a implicat parcurgerea celor doua TreeSet-uri, ce reprezinta polinoamele, Iar cu un al treilea for-each imbricat se parcurge rezultatul in care va fi polinomul final, se verifica daca exista deja un monom care are gradul calculat deja existent in TreeSet. Daca acesta exista, se vor aduna coeficientii. Daca aceasta conditie nu este indeplinita se va apela metoda de inmultire a doua monoame, iar monomul rezultat va fi adaugat in polinomul final despre care am amintit si anterior

*Metoda Imparte-*in implemetarea acestei metode am urmat intocmai algoritmul de impartire a doua polinoame. Datorita faptului ca aceasta metoda trebuie sa aiba ca si rezultat doua polinoame:un cat si un rest, ea are 4 parametrii: doi fiin polinoamele insasi, iar catul si restul sunt trimise tot ca si parametrii.

*Metoda Deriveaza-este o metoda care are ca si parametru un singur polinom. Se parcurge cu un for-each polinomul de la inceput, iar pentru fiecare monom component se apeleaza metoda de derivare din Clasa Monom:* coeficientul va fi produsul dintre exponent si coeficientul existent mai apoi se decrementeaza gradul exponentulu. Monomul rezultat va fi adaugat rezultatului.

*Metoda Integreaza*-la fel ca si in cazul derivarii, primeste un singur parametru(un polinom), se parcurge acest polinom cu un for-each, se apeleaza pentru fiecare monom, metoda de integrare din clasa Monom, in care coeficientul si exponentul iau forma specifica integrarii unui monom, iar fiecare monom rezultat este adaugat polinomului final ce va fi returnat.

**Clasa Controller:**

In aceasta clasa se face legatura dintre logica programului si interfata accesibila utilizatorului. Astel aceasta clasa contine inca alte 7 clase de acest tip, cate una pentru fiecare buton: “class AddListener implements ActionListener” fiecare dintre ele implementand cate o metoda care are ca si parametru cate un ActionEvent, in interiorul careia este descries c ear trebui sa se intample la apasarea fiecarui buton.

**Clasa View:**

In aceasta clasa se implementeaza interfata grafica despre care am povestit si in subcapitolul “Interfata Utilizator”. Aceasta clasa are un constructor in care se contureaza pas cu pas aceasta interfata, nu este una foarte sofisticata, am optat pentru una simplista si usor de folosit de catre orice utilizator. Tot aici mai sunt implementate cateva metode scurte, aflate in corelatie cu metodele din Clasa Controller.

5.Rezultate:

Corectitudinea operatiilor este testate cu ajutorul framework-ului Junit5. Se implementeaza o clasa de teste, in interiorul careia se va scrie cate o metoda de teste pentru fiecare operatie si se va putea rula oricand pentru a verifica corectitudinea rezultatului.

**Exemplu:**

void inmulteste() {  
 Polinom pol1=new Polinom("1X^4+2X^3");  
 Polinom pol2=new Polinom("1X^3+2X^1");  
  
 Polinom rezultat=Model.*inmulteste*(pol1, pol2);  
 *assertEquals*("+X^7+2.0X^6+2.0X^5+4.0X^4", rezultat.toString());  
}

6.Concluzii:

As putea concluziona aceasta documentatie prin sustinerea faptului ca ar putea exista o multitudine de modalitati de implementare a acestei aplicatii. De exemplu, polinomul introdus ar putea sa fie introdus intr-o forma mai frumoasa, sau inca multe altele.

Consider, de asemenea, ca aceasta tema m-a ajutat sa aprofundez paradigmele programarii orientate pe obiect, dar de asemenea si aprofundarea lucrului cu design-ul de proiectare MVC.

7.Bibliografie

* <http://www.mkyong.com/tutorials/junit-tutorials/>
* <http://zetcode.com/tutorials/javaswingtutorial/>
* <https://app.diagrams.net/>