**Facultatea de Automatica si Calculatoare**

**Tema nr. 1: Calculator de polinoame**

**Nume:** *Ilovan Bianca-Maria*

**Grupa:** *302210, semigr.1*

**An:** *2019-2020*

**Cuprins:**

**1.**Obiectivul temei

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

**2.1.**Analiza problemei

**2.2.**Use case-uri

**2.3**.Scenarii

**3.**Proiectare

**3.1.**Decizii de proiectare

**3.2.**Proiectare clase, pachete

**4.**Implementare

**5.**Rezultate si concluzii

**6.**Bibliografie

**1.Obiectivul temei**

Desi intr-o continua dezvoltare si evolutie, stim cu totii importanta bazelor matematice intr-un domeniu cum este Computer Science. Mai exact, orice problema sau aplicatie are la baza un principiu matematic fie el cat de simplist. Polinoamele si operatiile cu acestea sunt unele din cele mai intalnite notiuni datorita simplitatii lor, dar si datorita faptului ca pot “interpola” functii mult mai complexe. Astfel, cu siguranta in multe aplicatii putem regasi polinoame, intr-o forma sau alta, alaturi de alte elemente menite sa ajute la manipularea acestora.

Obiectivul acestei teme de laborator a pornit tocmai de la o idee simpla, de baza, dar totusi necesara pentru noi. Cerinta era sa propunem, iar mai pe urma sa implementam un calculator de polinoame care poate realiza diferite operatii intre 2 polinoame: adunare, scadere, inmultire sau derivare si integrare pe unul din cele 2 polinoame. De asemenea, era nevoie de o serie de operatii auxiliare care nu sunt in directa legatura cu functionalitatile clasei principale, aici fiind vorba despre modul parsarii string-urilor. Pe langa toata aceasta parte de logica ce tine fie de tratarea operatiilor, fie de modalitatea introducerii polinoamelor, necesara era si creearea unei interfete tocmai pentru a putea fi utilizata relativ usor de catre orice tip de utilizator.

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

**2.1.Analiza problemei**

Tinand cont ca programarea orientata pe obiecte se bazeaza pe conceptul obiectelor, este vorba despre o abordare care permite rezolvarea problemei prin cooperarea reciproca a obiectelor in cadrul programului.

Data fiind problema actuala, ar trebui precizat ca un polinom este o expresie compusa din mai multe monoame, monoame care la randul lor au caracteristic cate un coeficient si cate un exponent fiecare. Deciziile luate de cel care implementeaza aplicatia pot fi diferite, insa de avut in vedere de oricine este faptul ca rezultatul final al operatiilor trebuie sa fie acelasi indiferent de modul introducerii intrarilor. Pentru asta ar trebui sa cunoastem de ce intrari avem nevoie. In cazul de fata, stim ca rezultatul final este constituit din realizarea operatiilor elemntare, lucru pentru care avem nevoie de 2 polinoame.

**Aplicatia ar trebui sa faca posibila:**

• adunarea a 2 polinoame

• scaderea a 2 polinoame

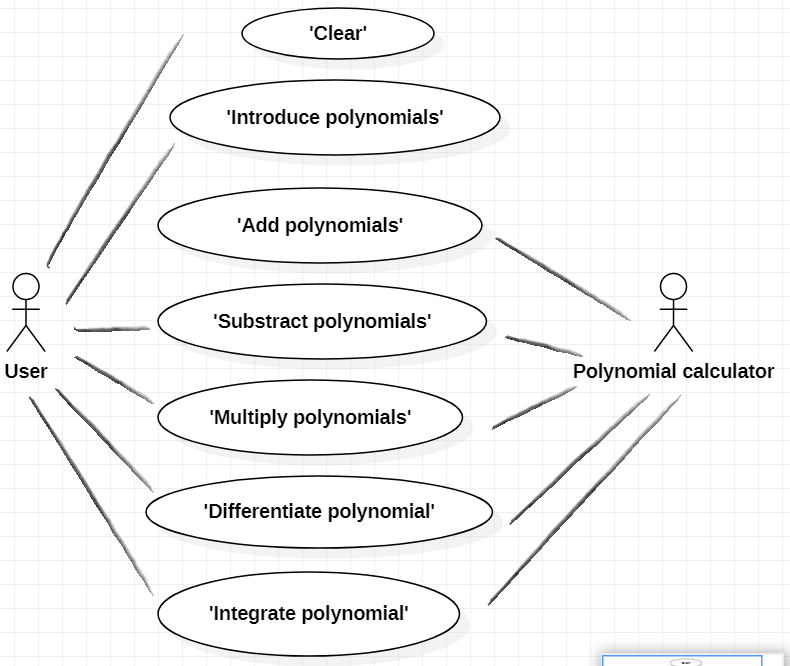
• inmultirea a 2 polinoame

• derivarea unui polinom

• integrarea unui polinom

**2.2.Use case-uri**

Functionalitea aplicatiei e descrisa bineinteles de diferite cazuri de utilizare ale acesteia, in cazul calculatorul de polinoame fiind vorba despre operatiile pe care el trebuie sa le efectueaza in functie de necesitate(sta la latitudinea utilizatorului sa aleaga ce operatie se realizeaza), fiecare indeplinindu-si functia specifica astfel incat rezultatul final sa fie cel dorit.



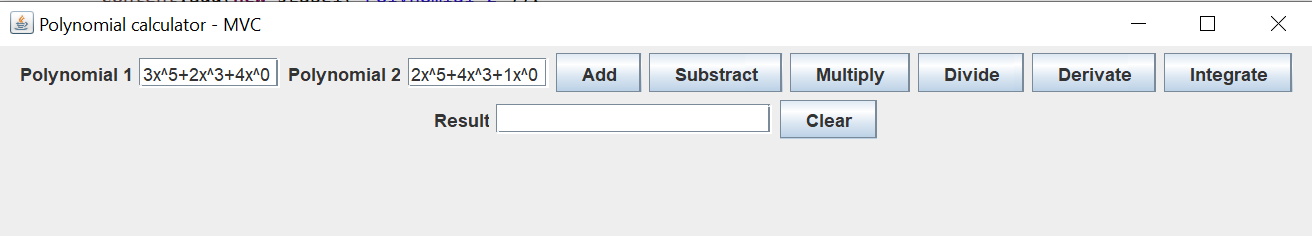
Colectia de use case-uri constituie posibilitatile de utilizare ale aplicatiei. Cu nume sugestive, acestea reprezinta practic operatii pe 2 polinoame.

User-ul introduce cele 2 polinoame, dupa care, in functie de decizia acestuia de a alege una din operatiile de mai sus, aplicatia incearca sa dea rezultatul dorit de utilizator ca raspuns pentru acesta. Utilizatorul poate alege orice operatie doreste, conditia fiind sa foloseasa butonul de “Clear” pentru stergerea rezultatului anterior, in cazul in care acesta vrea sa afle raspunsul pentru mai multe operatii.

Fiecare operatie este defapt o interactiune intre User si sistem, pentru un caz ideal, in asa fel incat totul merge exact asa cum e planuit.

**2.3 Scenarii**

Utilizarea in mod obisnuit a programului presunune introducerea primului polinom in text field-ul **“Polynomial 1”**, in urmatorul format: “3x^5+2x^3+1x^1+4x^0”(coeficientii si gradurile pentru fiecare monom trebuie sa fie explicite, nu conteaza ordinea in care monoamele se introduc). Dupa ce s-a introdus primul polinom operatiile care se pot realiza deja sunt cea de derivare, prin apasarea butonului **“Derivate”** si integrare, prin apasarea butonului **“Integrate”**, intrucat ele pot fi realizate doar pe primul polinom. In continuare, in cazul in care utilizatorul vrea sa mai efectueze alte operatii, va fi necesar sa mai introduca inca un polinom, in celalalt text field disponibil, **“Polynomial 2”** intr-un format asemanator ca al primului. Pentru a vedea rezultatul pentru una din operatiile de adunare, scadere sau inmultire, utilizatorul trebuie sa apese mai intai butonul de **“Clear”,** iar mai pe urma sa apese pe unul din butoanele **“Add”**, **“Substract”**, respectiv **“Multiply”**. Rezultatul dorit va aprea atat in consola cat si in text field-ul denumit **“Result”**.

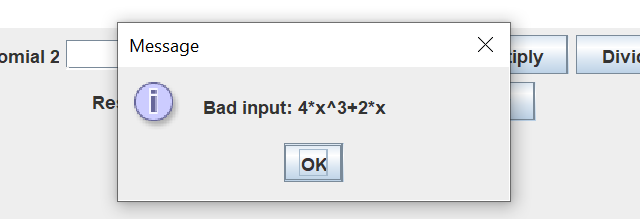


De mentionat este ca pot aparea si alte cazuri:

• scrierea polinomului “2x^3+1x^1+4x^0” intr-un format cum este “2\*x^3+1\*x+4\*x^0”, sau

“2\*x^3+1\*x+4”, scrieri des intalnite de altfel, dar care vor cauza aparitia unei casute cu “Bad input”, dat fiind faptul ca regex-ul folosit de mine a fost unul ce corespunde structurii primului polinom.

• introducerea de alte stringuri(nu polinoame) este posibila, insa nu va fi nimic afisat in casuta de **“Result”**. Acestea vor fi afisate doar in consola.



**3.Proiectare**

**3.1.Decizii de implementare, diagrama UML**

Calculatorul de polinoame are un design conform MVC(Model View Controller) pattern, format din 3 pachete: model, view si controller. In model se regasesc clasele: **Monomial**, **Polynomial** si **Operations**. In view regasim clasa **View**, iar in controller regasim clasa **Controller**. **App-ul** este o clasa pe post de main, unde sunt instantiate clasele Operations, View si Controller.

**3.2.Proiectare clase, pachete**

**Pachetul model:**

• clasa **Polynomial**: este cea care permite crearea polinoamelor ca un array list de monoame. Datorita acestei clase, calculatorul de polinoame recunoaste introducerea polinoamelor, putand sa deosebeasca coeficientii de exponenti, nefiind nevoie sa fie cate un text field separat in care sa fie nevoie sa introducem coeficientii pt fiecare exponent in parte.

• clasa **Monomial**: contine unitatea fundamentala pe care se bazeaza proiectul pana la urma: monomul, caracterizat de un exponent, coeficient(int-uri) cat si un hasPair(boolean) pentru a vedea daca monomul are sau nu pereche(un monom se considera a avea pereche daca intr-un alt polinom decat in cel care apare se intalneste un alt monom de acelasi grad, lucru ulterior folosit in clasa **operations**).

• clasa **Operations**: aici au loc toate operatiile, fie ca vorbim de operatii pe 2 polinoame cum sunt **adunarea**, **scaderea** si **inmultirea**, fie ca vorbim despre operatii care pot fi efectuate doar asupra primului polinom introdus: **derivarea** sau **integrarea**.

**Pachetul controller:**

**•** clasa **Controller**: raspunde de input-ul pe care user-ul il introduce si interactioneaza cu obiectele din clasa Operations.Se activeaza modulele in functia de optiunea aleasa de user.

**Pachetul view:**

**•** clasa **View**: prezinta modelul(Clasa Operations) intr-o anumita forma alesa de cel care implementeaza proiectul.Interfata grafica este datorata acestei clase, aici fiind locul in care se instantiaza listenereii separati fie ca vorbim de butoane sau de text field-uri.

**4.Implementare**

**4.1.Clasa Monomial**

Clasa elementara a acestui proiect intrucat, dupa cum am precizat pana acum, un polinom este format din mai multe monoame, fiecare caracterizat de un coeficient si un exponent care sunt int-uri, cat si de o variabila booleana hasPair pentru a verifica daca un monom are sau nu pereche. Ca metode, in aceasta clasa. Metoda toString() e folosita pentru afisarea corespunzatoare a monomului in functie de ce grad are: 0, 1 sau mai mare decat 1.

**4.2.Clasa Polynomial**

Este clasa care permite crearea polinoamelor ca array list-uri de monoame. Aici aparea problema parsarii string-urilor intrucat trebuia sa poata fi separati coeficientul si gradul unui polinom, astfel cu ajutorul metodei ***createMonomial(String monomial),*** avand ca input un string este posibil acest lucru si se va vedea si in consola. Medoda ***regexChecker(String theRegex, String strCheck)*** vine in ajutorul faptului ca aveam nevoie de o functie pentru crearea listei de monoame(regular expressions); strCheck este polinomul meu, caut monoamele si le afisez ca striguri **regexMatcher.group().trim()**, dupa care le adaug ca string-uri. Metoda ***readAsRegex(String polynomial)*** are ca input un polinom si este asemeni unei metode de validare ce verifica o expresie(un regex) pentru un polinom, de unde pot aparea si problemele in momentul in care user-ul introduce un polinom ce nu respecta un format precum cel precizat mai la inceput.

**4.3.Clasa Operations**

Este folosita pe post de Model din cadrul MVC pattern. Implementeaza principalele operatii ale calculatorului de polinoame.

• **Adunarea a 2 polinoame: addition(Polynomial p1, Polynomial p2)**

Presupunem ca avem polinoamele : p1(x) = 5x^3+2x^2+1x^1+4x^0

p2(x) = 2x^4+2x^3+3x^2+5x^1+2x^0

Aici pot avea mai multe cazuri si anume: cazul in care ambele polinoame au doar monoame de acelasi grad, fie unul din polinoame are monoame care sunt de grad diferit.

Ideea de baza a metodei este ca se parcurge fiecare polinom monom cu monom pentru a gasi monoame cu acelasi grad, hasPair(booalen) trebuind setat ca fiind adevarat in momentul gasirii a 2 monoame cu acelasi grad. Odata atins acest lucru, se creeaza un nou monom a carui nou coeficient ia valoarea insumata a celor 2 coeficienti a monoamelor cu acelasi grad, monomul fiind astfel adaugat in lista de monoame(ce constituie un polinom, mai exact rezultatul adunarii). Luand in considerare si cazul in care monoamele nu au acelasi grad, vom seta hasPair ca false(adica monomul nu are pereche) si va fi adaugat si el in rezultat.

• **Scaderea a 2 polinoame: substraction(Polynomial p1, Polynomial p2)**

Scaderea nu este decat un caz particular pentru adunare, singura diferenta fiind ca se scad coeficientii cu acelasi grad in loc sa se adune. De asemenea in cazul in care un monom din primul polinom nu va avea pereche va fi adaugat in rezultatul final, modificarile aparand doar in cazul in care in cel de-al doilea polinom apare un monom ce nu are pereche in primul polinom, deoarece va trenui adaugat in rezultat cu semn schimbat(m2.setCoefficient(0 – m2.getCoefficient());)

• **Inmultirea a 2 polinoame: multiply(Polynomial p1, Polynomial p2)**

Parcurg element cu element cu un foreach polynomial, inmultind fiecare monom cu fiecare(se inmultesc coeficientii monoamelor cu perechi si se aduna exponentii acestora).

• **Derivarea unui polinom: derivative(Polynomial p1)**

Aceasta e una din cele 2 operatii care se poate realiza pe un singur polinom, iar eu am ales ca derivarea sa se poata face pentru primul polinom. Astfel, pentru a obtine rezultatul dorit trebuie doar sa ne aducem aminte formula dupa care se derivau functiile polinomiale: parcurg polinomul, iar pentru fiecare monom din polinom va trebui schimbat coeficientul dat de produsul dintre vechiul coeficient al monomului si gradul acestuia. De asemenea noul grad pentru derivata va fi calculat ca fiind vechiul grad – 1.

• **Integrarea unui polinom : integrate(Polynomial p1)**

Din nou, este vorba despre o operatie care se poate efectua doar asupra primului polinom. Pentru ca rezultatul integrat sa poata fi obtinut, din nou, vom avea nevoie de formula matematica a integrarii unei functii polinomiale: pargurgem polinomul, iar pentru fiecare monom din polinom va trebui schimbat coeficientul, fiind dat de impartirea vechiului coeficient la gradul monomului actual. De asemenea noul grad pentru integrata va fi calculat ca fiind vechiul grad + 1.

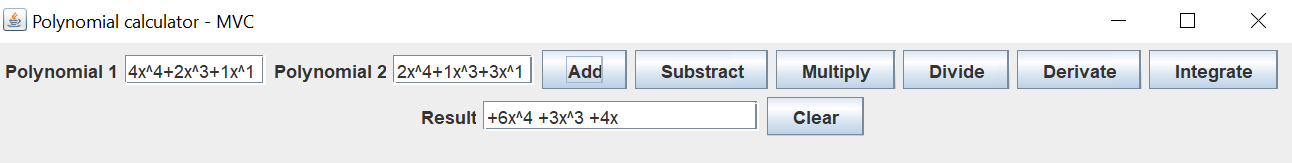
**4.4. Clasa View**

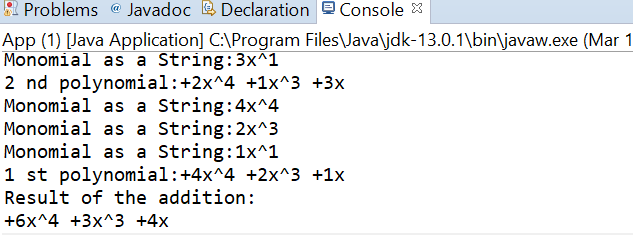
Constituie interfata grafica a aplicatiei constituita din 2 text field-uri in care se vor introduce 2 polinoame pe care se realizeaza 3 din cele 5 operatii posibile si butoane cu nume sugestive, pentru ca user-url sa poata alege ce operatie doreste sa realizeze. Mai pe scurt, prin intermediul acestei clase este posibila interactionarea utilizatorului cu sistemul. Aici sunt instantiati ascultatorii, cu rolul de a surprinde actiunile utilizatorilor. Fiecare text field si buton precizat mai sus are cate un ascultator separat.

**4.5. Clasa Controller**

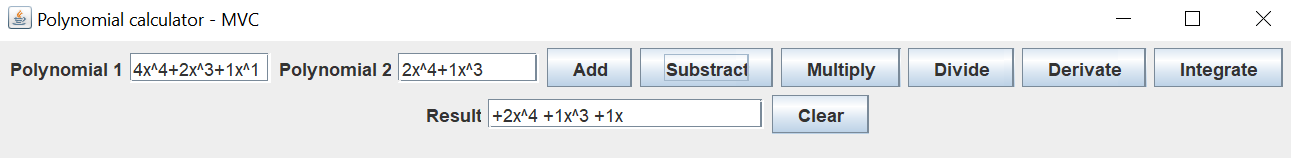
Aceasta clasa este cea care leaga modelul de view, gestionand toate actiunile venite din partea utilizatorilor. Aceasta clasa implementeaza interfata **ActionListener()** cat si metoda **actionPerformed().** Exista cate o clasa pentru fiecare operatie in parte, clasa care la randul ei implementeaza ActionListener. De exemplu pentru adunare: **class AdditionListener implements ActionListener**, cu o metoda **actionPerformed**. Aici trebuie sa ne asiguram ca user-ul nu poate introduce in acele 2 text field-uri decat expresia ce corespune regex-ului, motiv pentru care cele 2 user input-uri date de utilizator vor fi verificate de acea functie de validare **readAsRegex.** In cazul nerespectarii formatului impus, programul va arunca o exceptie. La fel se va proceda si pentru celelalte operatii. Singurele diferente de la **derivare** si **integrare** constau in faptul ca vom avea doar **un singur “user input”** intrucat acestea 2 se realizeaza doar pe primul polinom.

• **Adunarea**

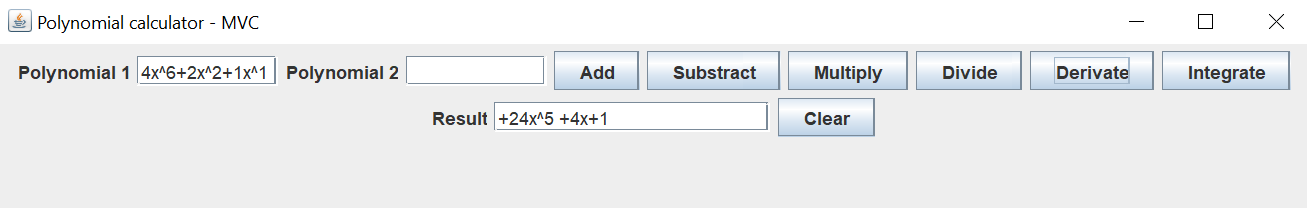


****

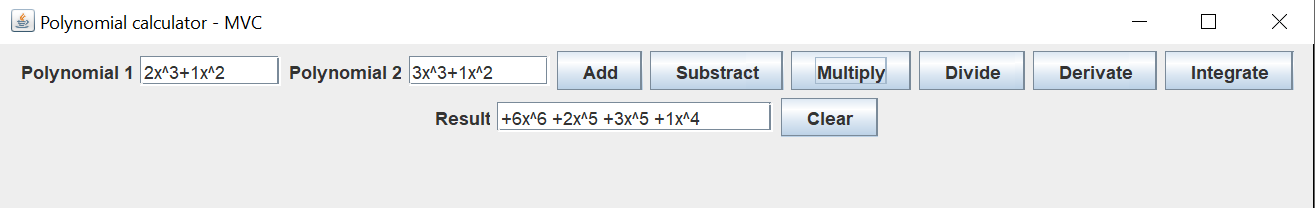
• **Scaderea**

****

• **Derivarea**

****

• **Inmultirea**

****

**5.Conluzii si rezultate**

Cred ca a fost o tema din care fiecare avea ceva de invatat indiferent de nivelul la care era. Esential este ca am invatat initial ce e un regex si mai pe urma sa lucrez cu tot ce inseamna aceste regular expressions, reusit sa separ coeficientii si exponentii monoamlor, lucru pe care il consideram la inceput destul de dificil si m-as fi agndit la orice alta metoda ca sa evit parsarea string-urilor.

De asemenea, mi-am reamintit si cum se realizeaza impartirea a 2 polinoame, dar nu m-a ajutat suficient de mult incat sa o si implimentez.

Important cred ca este sa iti stabilesti inainte sa te apuci de proiectul efetiv ce vrei sa faci, dar si cum ai vrea sa implementezi la ceea ce te gandesti. Cred ca asta a fost una din problemele mele, faptul ca nu m-am gandit suficinet inainte sa ma apuc de proiectul propiu zis, gandindu-ma ca ma voi descurca pe parcurs, motiv pentru care a trebuit sa reiau de mai multe ori proiectul.

**6.Bibliografie**

www.githhub.ro

www.stackoverflow.com

<http://users.utcluj.ro/~igiosan/teaching_poo.html>

<http://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/concepts/use_case_BB199D1B.html?fbclid=IwAR0BxhhLa2v_0RMpZIuHwBB2Kjnq7cPN1m2M7j9EE1nIzSmgixncu_d37BA>

<https://www.javatpoint.com/java-regex>

<https://www.youtube.com/watch?v=s_PfopWcMwI&feature=youtu.be>

<https://www.youtube.com/watch?v=sa-TUpSx1JA>

<https://www.cs.cmu.edu/afs/cs.cmu.edu/user/gchen/www/download/java/LearnJava.pdf>