DOCUMENTATIE

ASSIGMENT 1

NUME STUDENT: Neaca Radu-Sabin

GRUPA:30222

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 6](#_Toc95297887)

[4. Implementare 7](#_Toc95297888)

[5. Rezultate 9](#_Toc95297889)

[6. Concluzii 9](#_Toc95297890)

[7. Bibliografie 9](#_Toc95297891)

Obiectivul temei

Obiectiv principal:

Scopul lucrarii consta in crearea unui calculator polinomial ce dispune de diverse operatii si care se realizeaza pe 2 polinoame date de la tastatura.

Obiective secundare:

Cateva obiective secundare in indeplinirea proiectarii unui calculator polinomial ar fi:

* Verificarea a cat mai multor cazuri de implementare a polinoamelor: Un polinom poate fi scris sub forma coeficient\*X^exponent, insa exista diferse cazuri in care acesta sa nu mearga precum: prezenta unui exponent flotant, a unui caracter special pus neregulamentar si multe alte cazuri.
* Organizarea codului corespunzaor: Precum structura MVC (Model-View-Controller). Fiecare clasa creata nu trebuie sa depaseasca o dimensiune mai mare de 300 de linii, cu exceptia clasei Controller. Fiecare metoda trebuie sa cuprinda maxim 30 de linii. Clasele trebuie impartite in pachete astfel incat sa existe o vizibilitate si organizare a proiectului mult mai buna.

Analiza problemei, modelare, scenarii, cazuri de utilizare

Primul pas pe care trebuie sa il facem inainte de a incepe de implementat operatiile si interfata, ar fi modul cum salvam polinoamele. Cele 2 polinoame sunt date de la tastatura prin TextField-uri, si reprezinta 2 siruri de caractere ce trebuie salvate si impartite in monoame. Asadar, interfata programului ar trebui sa cuprinda in primul rand, 2 TextField-uri pentru polinoame, si inca unul pentru rezultatul operatiilor. Pe langa acestea, este necesara si prezenta mai multor butoane, care sa reprezinte cifrele de la 0 la 9, diverse caractere speciale (+ , - , \* , / , . , ^, del) fiecare cu un rol important, dar si butoane ce sa reprezinte operatiile pe care dorim sa le efectuam. Astfel, modelul de calculator polinomial ar arata in felul urmator.

Table

Description automatically generated

Fiind un string initial, polinomul poate avea forme gresite, asadar este necesara verificarea fiecarui string astfel incat polinomul inserat detina o forma specifica. Un polinom este alcatuit din cel putin un monom, asadar un prim caz ar fi daca unul din TextField-uri este gol. Un monom are structura: „Semn”„Coeficient”\*X^”Putere”, asadar, un polinom este declarat cu succes daca fiecare monom respecta structura data. In cazul in care unul dintre cele 2 stringuri nu respecta cerintele pentru un polinom, se va afisa un mesaj sub TextField-ul cu rezultatul operatiilor.

Operatiile prezente in calculatorul polinomial sunt: adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea.

1)Adunarea: Pentru realizrea adunarii a 2 polinoame este nevoie sa adunam monoamele ce au exponentii egali.

Exemplu:

(2\*X^3+3\*X^2-5\*X+1) +( 1\*X^2+3\*X^4-5\*X+6 ) = 3\*X^4+2\*X^3+4\*X^2-10\*X+7

2)Scaderea: Pentru realizarea scaderii a 2 polinoame este nevoie sa scadem monoamele ce au exponentii egali.

Exemplu:

(2\*X^3+3\*X^2-5\*X+1) +( 1\*X^2+3\*X^4-5\*X+6 ) = 3\*X^4-2\*X^3-2\*X^2+5

Exemple rulate in proiectul java:

A picture containing table

Description automatically generated Table

Description automatically generated

3)Inmultirea: Pentru realizarea inmultirii a 2 polinoame este nevoie sa inmultim fiecare monom dintr-un polinom cu fiecare monom din celalalt polinom. Dupa efectuarea acestor calcule urmeaza a fi adunate monoamele cu exponenti egali din polinomul rezultat.

Exemplu:

(2\*X^4+2\*X^3) \* (2\*X^4+2\*X^3) = 4\*X^8 + 4\*X^7 + 4\*X^7 + 4\*X^6

= 4\*X^8 + 8\*X^7 + 4\*X^6

Exemplu rulat in proiectul java:

Table

Description automatically generated

4)Derivarea: Pentru realizarea derivarii unui polinom este necesara parcurgerea acestuia iar la fiecare monom selectat, trebuie inmultit coeficientul acestuia cu exponentul, iar la urma, decrementarea exponentului cu 1.

Exemplu: 2\*X^4 + 2\*X^3 = 8\*X^3 + 6\*X^2

5)Integrarea: Pentru realizarea integrarii unul polinom este necesara parcurgerea acestuia, iar la fiecare monom selectat, trebuie incrementat exponentul cu 1 urmand apoi impartirea coeficientului la noul exponent.

Exemplu: 2\*X^4 + 2\*X^3 = 0.4\*X^5 + 0.5\*X^4

Exemple rulate in proiectul java:

Table

Description automatically generated Table

Description automatically generated

Proiectare

La proiectarea calculatorului polinomial am ales sa ma folosesc de mai multe clase precum: Monomial, Polynomial,Operations.

In Monomial am salvat coeficientul, exponentul si semnul fiecarui monom din polinomul inserat. Am folosit gettere si settere pentru a le memora si inca 3 metode ce returneaza date dintr-ul monom salvat ca un String, adica un subString al unui polinom. In clasa Polynomial am salvat un polinom de monoame cu ajutorul unei liste de siruri,fiecare element din sir reprezentand un monom din polinom. Astfel, apeland obiectul polinom, sa am la dispozitie orice monom in functie de indexarea data, dar si informatiile legate de acesta, precum valoarea coeficientului, a exponentului si a semnului. In clasa Operations, am creat metodele specifice pentru realizarea fiecarei operatii cerute(adunare, scadere, inmultire, impartire, derivare, integrare), dar si alte metode ajutatoare precum sortarea unui polinom, ce pozitioneaza monoamele descrescator in functie de exponentii lor. Pentru conectarea acestor 3 clase la interfata, am creat o noua clasa unde am folosit comenzi puse la dispozitie de javaFX.

Mai jos este prezenta diagrama UML a claselor:

Diagram

Description automatically generated

Implementare

Clasa „Calculator” este clasa ce contine main-ul programului, astfel, in aceasta clasa se realizeaza legatura Model-View-Controller. Aici se apeleaza clasa CalculatorController ce se ocupa cu interfata proiectului.

Pe partea de implementare, m-am folosit de librariile oferite de javaFX in clasa CalculatorController ( import javafx.fxml.FXML ). Pe langa aceasta, am folosit si „javafx.scene.control.Button” ce imi pune la dispozitie un set intreg de comenzi asupra unui buton din interfata. Librariile „ javafx.scene.control.TextField „ si „ javafx.scene.text.Text ”, la randul lor, pun la dispozitia utilizatorului un set de instructiuni ce-l ajut pe acesta sa programeze interfata programului mult mai usor. Aplicatia SceneBuilder ofera utilizatorului crearea unei ferestre a interfetei, avand la dispozitie toate blocurile necesare. Acestea blocuri dispun de mai multe functionalitati, precum actiuni pe care le indeplinesc la apasarea lor cu mouse-ul. Astfel, la fiecare apasare de mouse pe butonul respectiv se va realiza instructiunea introdusa din proiectul java. Butoanele din partea de jos a calculatorului polinomial au ca scop introducerea cifrei sau a caracterului special in TextField-ul selectat cu mouse-ul. Deasupra acestor butoane, se afla operatiile cerute. La apasarea unui buton dintre acestea, se apeleaza metoda respectiva fiecareia, urmand ca rezultatul sa fie afisat in TextField-ul corespunzator.

Pe langa acestea, mai este prezent si un mesaj ce precizeaza daca cele 2 polinoame sunt sau nu, corect inserate. Metoda acestui mesaj te conduce cantre o noua clasa „CheckPolynomial” ce contine la randul ei alte metode. In metoda „rulesPolynomial” este prezenta forma unui polionom inserat de la tastatura, astfel incat sa returneze o valoare negativa in cazul in care s-a inserat un polinom gresit. Metoda „checkString” verifica, care dintre cele 2 polinoame este gresit, urmand ca in metoda „checkMessage” sa se returneze un String cu mesaj corespunzator.

Pe langa aceasta clasa de verificare a polinomului, am creat inca 2 clase, „Monomial” si „Polynomial” fiind esentiale in rezolvarea operatiilor. Clasa „Monomial” are ca scop principal folosirea setterelor si getterelor, in care am salvat coeficientul, exponentul si semnul fiecarul monom. Pe langa acestea, am facut inca 3 metode, pe care le-am folosit ca sa salvam coeficientul, exponentul si semnul unui monom, in cazul in care acestea sunt salvate intr-un string de forma „coeficient \*X^ exponent”. In clasa Polynomial, m-am folosit de libraria de liste din java. Am creat o lista ce contine monoame, adica un polinom impartit in monoame. In metoda „createListPolynomial” avem un parametru ce reprezinta String-ul cu polinomul inserat de la tastatura. Aici impart String-ul in subsiruri, astfel incat sa am numai subsiruri de monoame. Fiecare monom rezultat este adaugat in lista, fiind salvate valorile coeficientilor si a exponentilor, dar si daca sunt negativa sau nu.

Revenint la operatii, exact cum am spus mai sus, dupa apasarea butoanelor respective, acestea te conduc catre metodele respective.

Butonul de „Addition” realizeaza adunarea celor 2 polinoame, astfel, dupa apasarea acestuia se creeaza 2 Stringuri ce retin cele 2 polinoame inserate de la tastatura. Se verifica daca respecta forma unui polinom, iar dupa se apeleaza printr-un obiect, clasa Operations cu metoda respectiva „addSubTwoPolynomial”. Aici am salvat coeficientii si exponentii in liste de monoame. M-am folosit de motode ajutatoare in cazul in care primul monom din lista nu este cel cu exponentul cel mai mare. Metoda „sortPolynomial” returneaza o lista de monoame ce continea toate monoamele in ordine descrescatoare a valorii detinute de exponent. Am salvat exponentii fiecarui monom intr-un array ajutator, astfel incat sa stiu de fiecare data care dintre cele 2 polynoame are cel mai mare exponent, si cati sunt mai mari sau egali cu 0. In care un exponent este 0, monomul reprezinta o constanta. Tot aici am si cazul in care un coeficient este 1, la afisare prezenta acestuia este inutila. Cand un monom are coeficientul 0, acesta nu exista, iar in cazul in care cele 2 liste de monoame au la o indexare comuna coeficientii 0, la afisare nu mai este necesara scrierea lor. Metoda returneaza un String ce este salvat in String-ul „resultPolynomial” ce va afisa resultatul adunarii dintre cele 2 polinoame in blocul din interfata.

Butonul de „Substract” realizeaza impartirea celor 2 polinoame. Similar cu butonul de adunare, acesta verifica daca cele 2 String-uri sunt polinoame sau nu. Se apeleaza aceeasi metoda, avand ca parametru in plus caracterul special - . Prezenta acestuia schimba in metoda partea cand cele 2 monoame cu exponenti egali se aduna, operatia dintre acestia fiind de scadere acum. Ca si la adunare, se respecta aceleasi reguli de afisare a rezultatului, fiind returnat sub forma de String de catre metoda.

Butonul de „Multiplication” realizeaza inmultirea dintre 2 polinoame. Aceasta metoda seamna spre deosebire de celelalte, salveaza intr-o lista temporara, monoamele rezultate in urma inmulririi. Aceasta lista temporara poate sa contina monoame ce au exponenti egali. Pentru rezolvarea acestei probleme, se ordoneaza lista temporara in asa fel incat sa contina monoamele in ordine descrescatoare valorii pe care o detin exponentii lor. In urma acestei sortari, se parcurge iar lista temporara, adunand monoamele ce au coeficienti egali, urmand ca rezultatul sa fie salvat intr-un string in asa fel incat sa respecte forma unul polinom. Aceasta metoda ca oricare alta, returneaza un String ce urmeaza a fi afisat in blocul corespunzator din interfata.

Butonul de „Division” are ca scop impartirea celor 2 polinoame. Intrucat nu am reusit implementarea unei metode ce sa rezolve aceasta problema, nu am ce sa prezint la aceasta operatie.

Butonul de „Derivative” realizeaza derivarea unui polinom selectat. Similar ca la celelalte operatii, se verifica daca este polinom, iar apoi se apeleaza metoda „derivateTwoPolynomial”. Diferit fata de celelalte operatii, aceasta metoda parcurge lista o singura data. Seteaza noua valoare a coeficientului ca fiind rezultatul inmultirii dintre coeficient si exponent. Valoare exponentului se decrementeaza. Aceste operatii se intampla cat timp exponentul monomului este mai mare sau egal cu 0, iar monoamele cu exponentul intial 0 urmand a fi eliminate din lista.

Butonul de „Integration” realizeaza integrarea celor 2 polinoame. Similar ca la celelalte operatii, se verifica daca polinomul inserat respecta forma unui polinom. Aceasta operatie este similara cu cea de derivare, singura diferenta fiind la noile valori alea coeficientului si exponentului. Coeficientul monomului in urma integrarii o sa aibe valoarea (oldCoeficient \* (1 / oldExponent)). Noul exponent al monomului o sa fie incrementat cu 1. Iar mai apoi String-ul in care am salvat polinomul o sa fie returnat si afisat in blocul din interfata.

O metoda folosita de aproape toate operatiile, o reprezinta „createResultStringFromList” in care salvam sub forma de String toate monoamele din lista de monoame. Astfel se returneaza un String cu noul polinom creat in urma operatiilor realizate.

Rezultate

Pentru fiecare dintre cele 5 operatii efectuate in calculatorul polinomial, am realizat 3 teste de inserare pentru fiecare in parte. Pentru aceste teste m-am folosit de JUnit.  
 La operatia de adunare am implementat diverse cazuri printre care si cazul in care unul din monoamele unui polinom este negativ, iar la celalalt polinom avem numai monoame pozitive. Un caz in care monoamele de la un polinom are exponentii diferiti fata de monoamele de la celalt polinom. La operatia de scadere am realizat exemple similare cu cele de la adunare. La inmultire am verificat cazuri in care exista intr-un polinom mai multe monoame iar in altul mai putine. La integrare si derivare am folosit exemple similare, amandoua urmand reguli similare. Am verificat daca un monom are gradul 0, dupa derivare, acesta dispare. Am verificat daca am o constanta, aceasta dupa integrare inmultindu-se cu X.

Concluzii

Aplicatia realizeaza cu succes operatii aritmetice pe 2 polinoame, cu exceptia operatiei de impartire. O metoda de imbunatatire a acestui proiect ar fi atat proiectarea operatiei de impartire, cat si mici reglementari la afisare precum: la derivare si integrare, sa se realizeze operatia si pe polinomul inserat in al 2-lea textField, nu doar in primul. Pe langa acestea, am reusit sa ma familiarizez mai mult cu listele din java, dar si cu structura de creare a unui proiect, avand diverse pachete cu clase specifice. Familiarizarea cu javaFX mi-a fost de folos, fiind primul proiect in care folosesc acesta metoda de creare a interfetei. O ultima concluzie ar fi modul de organizarea a codului, in conditiile in care la inceputul proiectului am creat haotic anumite clase, am ajuns spre final sa le rearanjez incat sa nu ma incurc in denumiri sau in prezenta unor metode fara rost pe care am incercat sa le sintetizez cat posibil de mult.

Bibliografie

Resurse pe care le-am folosit in realizarea proiectului:

<http://csfaculty.tcu.edu/comer/20803/MVCDemo.pdf>

<https://www.geeksforgeeks.org/arraylist-in-java/>

<https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf>

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