MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTATIE
CALCULATOR DE POLINOAME

BORZA DIANA-CRISTINA
GRUPA 30225 | AN 2 SEMESTRUL 2

Cuprins

1.Obiectivul temei

1.1.Obiectivul principal al temei

Obiectivul principal al temei este acela de a creea o aplicatie care implementeaza diferite operatii efectuate pe polinoame cu coeficienti intregi. Aplicatia trebuie sa dispuna de o interfata prietenoasa accesibila oricarui utilizator si in care acesta va putea introduce unul sau doua polinoame in functie de operatia aleasa si sa vizualizeze rezultatul operatiei. Operatiile care vor fi accesibile utilizatorului sunt: adunare, scadere, inmultire, impartire, derivare si integrare.

1.2.Obiective secundare

Obiectivele secundare reprezinta pasii care trebuie urmati pentru a indeplini obiectivul principal, iar ele sunt definite in cele ce urmeaza.

- Analiza problemei si identificarea cerintelor

Programul trebuie sa permita utilizatorului sa introduca doua polinoame si sa selecteze operatia care se va efectua asupra celor doua polinoame. De asemenea, programul trebuie sa permita utilizatorului sa stearga datele introduse si de a introduce unele noi. Interfata programului trebuie sa fie prietenoasa, usor de utilizat si intuitiva, astfel incat oricine sa o poata utiliza. De asemenea, interfata trebuie sa contina toate cele sase optiuni pentru fiecare operatie, iar calculatorul sa implementeze acele operatii in scopul de a returna pe ecranul calcuatorului(al interfetei calculatotului de polinoame) rezultate corecte. Acest obiectiv va fi prezentat pe larg in capitolul 2.

- Proiectarea calculatorului polynomial

Intregul sistem la un moment dat de timp trebuie sa aiba trei intrari si o iesire. Intrarile se definesc ca fiind cele doua polinoame indroduse de la tastatura de catre utilizator si operatia aleasa de acesta, iar iesirea este polinomul rezultat al operatiei selectate. Interfata trebuie sa fie una usoara de inteles si de utilizat. Obiectivul va fi detaliat in capitolul 3.

- Implementarea calculatorului polinomial

Pentru implementarea calculatorului este necesara cunoasterea conceptului de polinom. Un polinom este alcatuit din mai multe monoame, iar un monom contine un coeficient, o variabila(de obicei x) si un exponent. O suma de monoame alcatuiesc un polinom, iar din aceste aspecte matematice se porneste implementarea claselor aplicatiei java. Obiectivul va fi explicat amanuntit in capitolul 4.

- Testarea calculatorului polynomial.

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

Cerinta functionala a proiectului este aceea de a implementa un sistem care proceseaza diferite oepratii pe polinoame cu coeficienti intregi si de o singura variabila. Operatiile pe care sistemul trebuie sa le efectueze sunt:

- Preluarea unui polinom sub forma unui sir de caractere de forma $a_nx^n + a_{n-1}x^{(n-1)} +\ldots+ a_1x + a_0$

Unde a_n, a_{n-1} , ... , a_0 sunt numere intregi iar puterile n, n-1, … , 1, 0 sunt numere naturale pozitive.

- Extragerea coeficientilor si a puterilor pentru a forma un polinom din sirul de caractere introdus de catre utilizator.
- Adunarea a doua polinoame cu coeficienti intregi.
- Scaderea a doua polinoame cu coeficienti intregi.
- Inmultirea a doua polinoame cu coeficienti intregi.
- Impartirea a doua polinoame cu coeficienti intregi.
- Derivarea unui polinom cu coeficienti intregi.
- Integrarea unui polinom cu coeficienti intregi.

Pentru a face interfata cat mai usor de folosit si de inteles, utilizatorul va introduce forma polinomului de la tastatura, iar programul va folosi Regex pentru a extrage fiecare grup de monoame in parte. Regex reprezinta o expresie regulata; o secventa de caractere care specifica un model de cautare. Astfel, cu ajutorul acesteia se pot extrage grupurile de monoame, dar se poate si verifica daca expresia introdusa de utilizator este corecta.

Cazuri de utilizare: un utilizator insereaza doua polinoame de la tastatura si apasa un buton corespunzator operatiei dorite. In caz de success polinoamele sunt corecte, iar rezultatul este afisat in interfata grafica. Un caz de esec reprezinta introducerea gresita a polinoamelor sau un polinom necompletat. In acest caz se afiseaza un mesaj si se verifica din nou cele doua polinoame la apasarea pe buton.

3.Proiectare

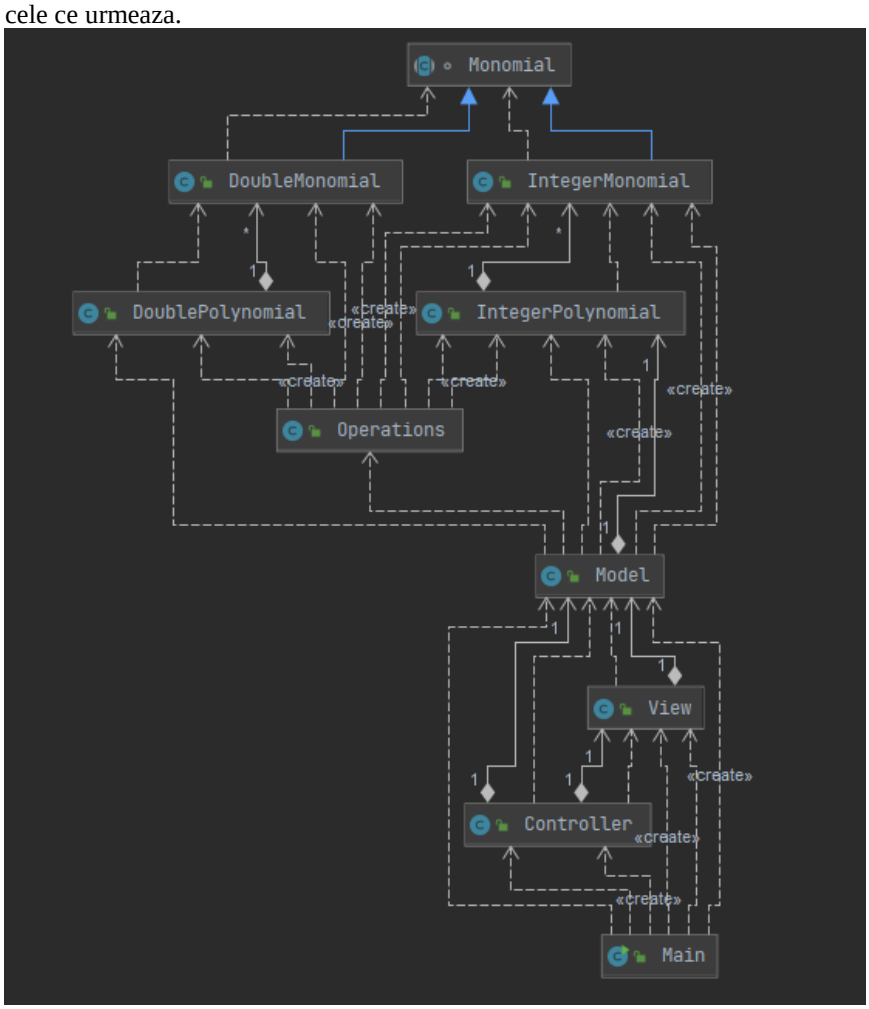
3.1.Decizii de proiectare

Pentru proiectarea aplicatiei am decis ca va fi nevoie de doua tipuri de monoame, anume unul cu coeficienti intregi si unul cu coeficienti reali. Polinoamele vor fi de o singura variabila, x. In cazul in care de la tastatura se introduce o alta variabila se va afisa un mesaj corespunzator. Astfel, variabila fiind mereu x, ea nu este stocata intr-un monom. Un obiect de tip monom va contine coeficientul si exponentul(puterea). De asemenea va mai exista o variabila a clasei ‘marked’ care va fi utila in implementarea operatiilor. Astfel, va exista un tip abstract Monomial si doua tipuri IntegerMonomial si DoubleMonomial, diferenta intre cele doua care mostenesc Monomial va fi tipul coeficientului. Vom avea nevoie de coeficienti intregi pentru adunare, scadere, inmultire si derivare, iar de coeficienti reali pentru impartire si integrare.

Pentru ca un polinom este o suma de monoame, vor exista doua tipuri corespunzatoare IntegerPolynomial si DoublePolynomial fiecare continand o lista de monoame intregi sau reale.

3.2.Diagrama UML

Diagrama specifica aplicatiei este vizibila in imaginea urmatoare, iar componentele acesteia, anume clasele vor fi discutate in cele ce urmeaza.



3.3.Structuri de date

Structura de date folosita predominant in aplicatie este ArrayList<> . Un polinom va contine un ArrayList de monoame de tipul int sau double in functie de tipul polinomului. Am ales aceasta structura de date deoarece este foarte utila, avand deja functii predefinite utile precum: add, isEmpty, get, set, etc.

3.4.Proiectare clase

Aplicatia modeleaza obiecte de tipul monom, definite in clasa Monomial. Aceasta clasa este o clasa abstracta care are doua subclase, IntegerMonomial si DoubleMonomial necesare procesarii operatiilor pe obiecte de tip polinoame intregi. Prin urmare, exista si doua clase de tip polinom, IntegerPolynomial si DoublePolynomial care contin fiecare cate o lista de monoame intregi sau reale, in functie de tipul clasei.

Clasa Operations contine toate functiile necesare implementarii operatiilor. Functiile primesc ca si parametru variabile de tip polinom intreg si returneaza, in functie de tipul operatiei, un polinom de tip intreg sau real.

Pentru interfata grafica, am ales sa implementez clasele intr-o structura de tipul Model View Controller. Astfel, exista o clasa Model unde sunt implementate functii care se apeleaza ca si raspuns al unei actiuni in interfata, o clasa View unde este definite interfata efectiva si design-ul sau si o clasa Controller unde sunt implementate ActionListener pentru butoane si care fac legatura intre model si view.

Clasa Main instantiaza modelul, viewul si controllerul sub forma Model View Controller. Este clasa care contine metoda main asupra careia se va efectua rularea aplicatiei.

3.5.Interfete

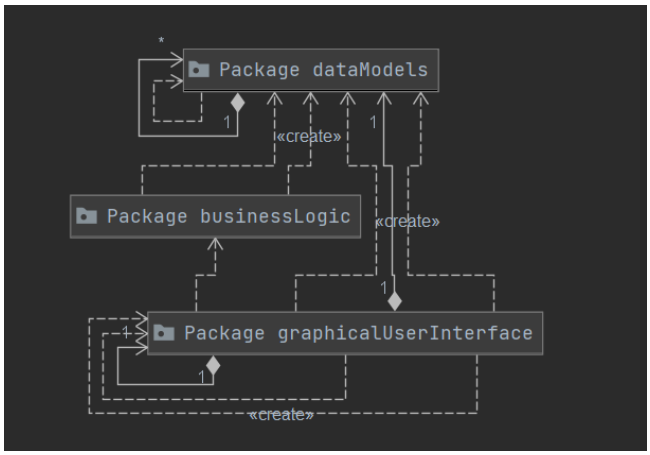
Clasele aplicatiei nu implementeaza interfete definite proprii, dar clasele IntegerMonomial si DoubleMonomial implementeaza interfata Comparable<> si implicit metoda compareTo(). Aceasta implementare de interfata este foarte utila deoarece metoda compareTo() a fost suprascrisa astfel incat va compara exponentul a doua obiecte de tip monom si va returna -1 daca primul monom are un exponent care este mai mare decat exponentul celui de-al doilea monom, 0 daca sunt egale si 1 in cazul in care exponentul primului monom este mai mic decat exponentul celui de-al doilea. Aceasta metoda suprascrisa ne ajuta in ordinarea unui polinom, care este de fapt o lista de monoame. Astfel, folosind metoda din Collections.sort() polinomul se va sorta in ordine descrescatoare in functie de exponent, deoarece functia sort() va sorta polinomul utilizand metoda compareTo() suprascrisa. Sortarea polinomului este necesara atat pentru o afisare riguroasa, matematica, cat si pentru unele metode, precum metoda care efectueaza impartirea a doua polinoame.

3.6.Relatii

Intre clasa IntegerMonomial si IntegerPolynomial exista o relatie de tip agregare. Clasa IntegerPolynomial contine o lista de obiecte de tipul IntegerMonomial care sunt create inafara clasei si transmise ca parametru in functia addValue(). Daca clasa IntegerPolynmial nu ar exista sau s-ar sterge, obiectele de tip IntegerMonomial ar exista in continuare independent. La fel este si in cazul claselor DoubleMonomial si DoublePolynomial.

3.7.Pachete

Aplicatia contine trei pachete care ajuta la diferentierea claselor in functie de utilitatea lor. Astfel, vom regasi un pachet numit dataModels in care se afla clasele care modeleaza aplicatia(clasele Monomial, IntegerMonomial, DoubleMonomial, IntegerPolynomial si DoublePolynomial), un pachet numit businessLogic care contine clasa care implementeaza operatiile matematice(clasa Operations) si un pachet numit graphicalUserInterface care contine clasele care implementeaza interfata grafica pentru utilizator(clsele Model, View, Controller si Main).



3.8.Interfata utilizator

Interfata pentru utilizator va contine 6 butoane corespunzatoare celor 6 operatii, un buton pentru resetarea valorilor polinoamelor si trei textField-uri. Doua dintre acestea sunt editabile si in ele se vor introduce datele de la tastatura de catre utilizator, iar unul este destinat afisarii rezultatului, iar acesta nu poate fi editat de catre utilizator. Pentru operatiile de derivare si integrare se va loa in considerare doar unul din polinoamele introduse(primul).

4.Implementare

Aplicatia contine zece clase structurate astfel:

Clasa **Monomial**: este clasa abstracta care contine variabilele de clasa exponent si marked. De asemenea aici sunt definite si functiile pentru set si get a variabilelor si constructorul clasei, dar si o metoda abstracta toString() care va fi implementata de fiecare subclasa. Variabila marked va marca faptul ca un exponent a fost verificat la un moment dat de timp in parcurgerea polinomului in vederea efectuarii operatiilor.

Clasa **IntegerMonomial** mosteneste clasa abstracta Monomial, implementeaza interfata Comparable si are variabila de clasa coefficient de tip intreg. Implementeaza metoda toString() care returneaza polinomul sub forma unui string precum cel introdus de la tastatura de catre utilizator. Constructorul clasei apeleaza constructorul superclasei si seteaza coeficientul si exponentul transmisi ca paremetru, iar variabila marked este initializata cu false. Clasa implementeaza metode de set si get pentru coeficienti, dar si metoda compareTo si metoda toString(). Metoda compareTo() compara doua monoame in functie de exponenti si este utila pentru a sorta o lista de monoame, iar metoda toString() returneaza polinomul sub forma unui string.

Clasa **DoubleMonomial** mosteneste clasa abstracta Monomial, implementeaza interfata Comparable si are variabila de clasa coefficient de tip double. Clasa contine aceleasi metode precum clasa IntegerMonomial, dar adaptate pe tipul de date double al coeficientului.

Clasa **IntegerPolynomial** defineste obiecte de tip polinom cu coeficienti intregi. Ea contine ca si variabila de clasa o lista(ArrayList) de monoame cu coeficienti intregi. Pe langa metoda get care returneaza polinomul, in clasa se mai gasesc si constructorul clasei, functia addValue() care adauga un monom in lista de monoame a clasei, functia sortPolynomial() care efectueaza sortarea in ordine descrescatoare a polinomului in functie de exponentul sau, functia getFirstMonom() care returneaza monomul cu cea mai mare putere(primul monom) , metoda clearMarks() care seteaza toate variabilele marked cu fals si metoda getPolynomialString() care returneaza intreg polinomul sub forma unui string ca si suma de monoame.

Clasa **DoublePolynomial** defineste obiecte de tip polinom cu coeficienti reali. Ea implementeaza aceleasi metode precum clasa IntegerMonomial adaptate tipului de date double. Variabila clasei este o lista de monoame cu coeficienti double.

Clasa **Operations** contine toate metodele utilizate pentru efectuarea operatiilor, Sunt implementate metodele pentru adunare, scadere, inmultire, impartire, derivare si integrare. Metoda addition() aduna doua polinoame de tipul IntegerPolynomial si returneaza un polinom de acelasi tip. Metoda subtraction() scade doua polinoame de tipul IntegerPolynomial si returneaza un polinom de tipul IntegerMonomial. Metoda subtractionDouble() scade doua polinoame de tipul DoublePolynomial si returneaza un obiect de tipul DoublePolynomial. Operatia de scadere a trebuit implementata pentru ambele tipuri de polinoame deoarece pentru operatia de impartire este necesara o metoda de scadere a doua polinoame cu coeficienti reali. Metoda multiplication() inmulteste doua polinoame de tipul IntegerPolynomial si returneaza rezultatul sub forma de IntegerPolynomial. Si aceasta operatie are o alta functie coresponenta, anume multiplicationDouble() care inmulteste un polinom de tipul DoublePolynomial cu unul de tip IntegerPolynomial si returneaza un rezultat de tipul DoublePolynomial. Si aceasta metoda este folosita in cadrul metodei care implementeaza operatia de impartire. Metoda derivation() primeste ca parametru un polinom de tipul intreg, IntegerPolynomial, efectueaza operatia de derivare si intoarce ca rezultat un obiect de tipul IntegerPolynomial. Metoda integration() primeste la fel precum metoda derivation() un polinom cu coeficient intreg, efectueaza operatia de integrare si intoarce ca rezultat un polinom IntegerPolynomial. Metoda division() efectueaza impartirea a doua polinoame cu coeficienti intregi si returneaza o lista de tipul DoublePolynomial care va contine doua elemente, catul si restul impartirii care sunt de tipul DoublePolynomial deoarece coeficientii polinoamelor cat si rest pot fi reali.

Clasa **Model** contine trei variabile de tipul String polinomInput1, polinomInput2, resultOutput care reprezinta cele doua polinoame ce vor fi preluate din interfata grafica si rezultatul operatiei care se va afisa in interfata grafica sub forma unui sir de caractere. De asemenea, clasa contine si doua variabile de tipul IntegerPolynomial polinom1, polinom2 care reprezinta polinoamele sub forma unor obiecte de tipul polinom cu coeficienti intregi. Clasa contine metode get si set pentru polinoamele input de tip string si o metoda get pentru

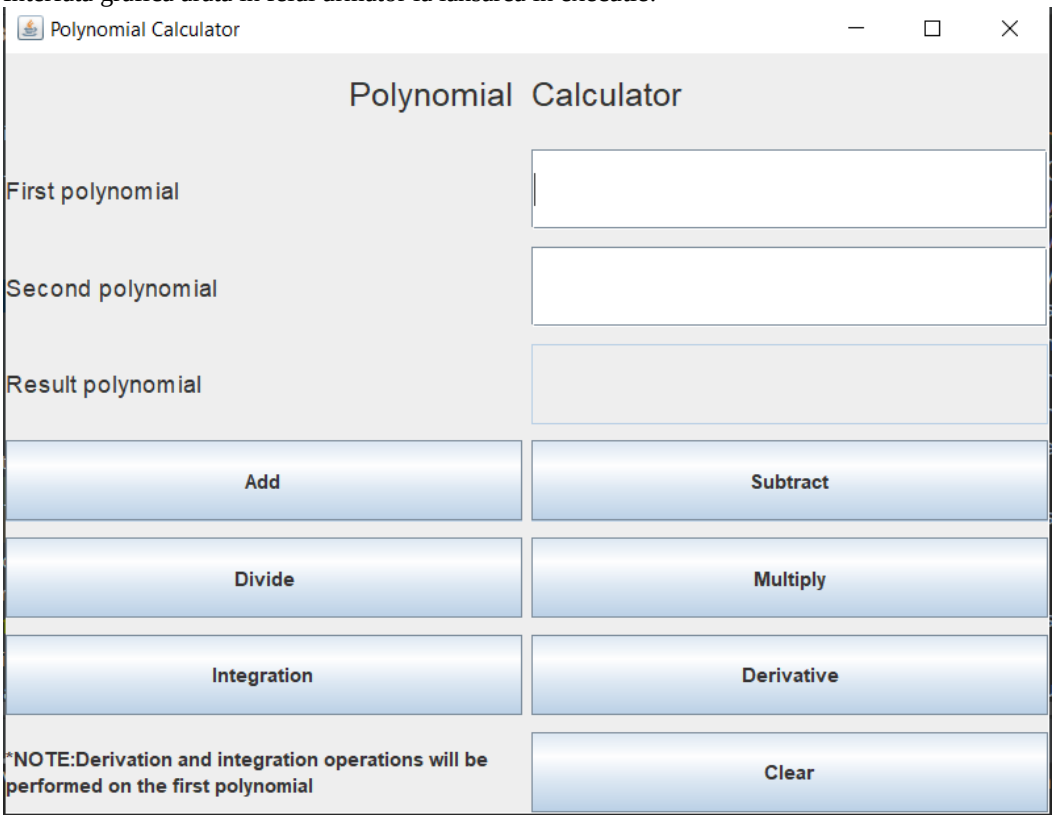
polinomul rezultat de tip String. Metoda detPolynomial() primeste ca si parametru un sir de caractere si foloseste Regex pentru a desparti sirul in grupuri bazate pe un pattern dat. Astfel, se obtin subsiruri cu fiecare monom din polinom, se creeaza un monom cu coeficientul si puterea extrase si se adauga polinomului intors ca rezultat de tipul IntegerPolynomial. Metoda este private si nu poate fi accesata inafara clasei. Pentru a determina cele doua polinoame primate din interfata se apeleaza functia setPolynomials() care apeleaza metoda detPolynomial() pentru fiecare din cele doua stringuri polinomInput1, polinomInput2 si seteaza valorile celor doua polinoame de tipul IntegerPolynomial. In clasa se regasesc si metodele addition(), subtraction(), multiplication(), derivation(), integration() si division() care efectueaza fiecare operatiile specifice din clasa Operations si seteaza resultOutput cu rezultatul coresounzator operatiei sub forma de string. Metoda checkGrades() verifica daca primul polinom are gradul mai mare decat cel de-al doilea polinom. Aceasta verificare este necesara in vederea efectuarii operatiei de impartire.

Clasa **View** reprezinta vizualizarea datelor care sunt continute in model. Ea contine ca si variabile un frame, trei JTextField-uri corespunzatoare celor doua polinoame si a rezultatului, sapte butoane corespunzatoare celor sase operatii si a operatiei de clear, dar si o variabila de tipul Model. Clasa contine metode de set si get pentru variabilele de tip JTextField, o metoda de setare a acestora ca fiind goale (clear()), si metode de adaugare de ActionListener pentru fiecare buton in parte.

Clasa **Controller** controleaza fluxul de date in obiectul de tip Model si actualizeaza obiectul de tip View ori de cate ori se modifica datele. Ea contine subclase pentru fiecare nou tip de Listener care implementeaza ActionListener pentru toate cele sapte butoane. Pentru fiecare buton de operatie valorile de intrare se testeaza folosind Regex daca respecta pattern-ul implementat. In caz contrar se afiseaza un mesaj de eroare. La apasarea pe un buton datele se transmit modelului, iar raspunsul este transmis interfetei grafice.

Clasa **Main** leaga toate cele trei clase Model, View, Controller prin instantierea a cate unui obiect din fiecare tip. Este clasa care contine metoda main() si care se va lansa in executie.

Interfata grafica arata in felul urmator la lansarea in executie:



5.Concluzii

Implementarea programului ilustreaza tehnicile de programare orientate obiect, modul de impartire al claselor, de stabilire a relatiilor, de implementare eficienta si concisa, dar si de implementare a unei interfete

accesibila oricarui utilizator. Prin aceasta tema mi-am aprofundat cunostintele in limbajul java si a tehnicilor de programare.

Aplicatia ar putea fi dezvoltata ulterior cu alte operatii precum extragerea radacinilor unui polinom sau ridicarea la putere.

6.Bibliografie

https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html
https://docs.oracle.com/javase/tutorial/java/concepts/index.html