**Implementarea unui sistem de procesare a polinoamelor**

**-documentație-**

# Ciceu Bogdan-Ioan

Grupa 30221

Cuprins

# Obiectivul proiectului

1. Analiza problemei
2. Modelare
3. Scenarii
4. Proiectare
5. Rezultate
6. Dezvoltări ulterioare
7. Concluzii, ce s-a invățat din tema
8. Bibliografie

# Obiectivul proiectului

Obiectivul proiectului este acela de a propune, proiecta și implementa un sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi.

Acest proiect va fi realizat folosind limbajul de programare Java . Se așteaptă ca la finalizarea proiectului să existe o interfață grafică realizată cu Swing, care să ofere posibilitatea de a efectua următoarele operații pe polinoame:

* Aflarea valorii polinomului într-un punct dat;
* Derivare;
* Integrare;
* Adunare;
* Scădere;
* Înmulțire;
* Împărțire.

1. Analiza problemei

Problemele de proiectare ce o să apară trebuie rezolvate folosind principiile OOP, pentru eficiență și pentru oferirea unei soluții ușor de implementat. Acest lucru se va realiza prin procedeul de abstractizare. Abstractizarea este termenul folosit pentru mecanismul care ne oferă posibilitatea de a simplifica o problemă complexă. De exemplu, limbajul de asamblare este realizat prin abstractizarea limbajul cod mașină, iar limbajele de nivel jos (C, Fortran, Pascal) reprezintă abstractizarea limbajului de asamblare.

Citirea obiectivului într-un mod superficial nu mi-a impus multe probleme de proiectare, chiar mi-a dat impresia că aplicația ar putea fi realizată destul de ușor scriind totul într-o singură clasă. Ce înseamnă acest lucru? Având în vedere că s-a cerut implementarea unui sistem de procesare a polinoamelor cu coeficienți întregi, implementarea pare simplă și ușor de realizat prin porgramarea procedurală sau imperativă.

Însă, analizând mai în detaliu operațiile ce trebuie realizate, putem observa că în urma efectuării operațiilor de împărțire și integrare, polinomul rezultat poate avea și coeficienți reali. Astfel compexitatea crește și trebuie găsită o modalitate de a asigura eficiența și precizia rezultatelor .

1. Modelare

Am hotărât să fac o clasa ce va reprezenta schița unui polinom și o clasă ce va reprezenta schița unui Monom. Clasa Polinom va folosi, prin intermediul relației de agregare, obiecte de tip Monom. De exemplu, avem un obiect Polinon, P(x)=X^3+6x^2+1. Acest polinom este o listă de Monoame. În acest caz, avem 3 Monoame, M(0)=1x^0, M(1)=6x^2 și M(2)=x^3

După cum spuneam mai sus, la capitolul de proiectare, este necesar să mă asigur că fiecare Monom are un coeficient de tip corespunzător, întreg sau real. Prin urmare, am hotărât să modelez două clase, una pentru coeficienții întregi și alta pentru coeficienții reali . Aceste clase vor moștenii clasa Monom, care va fi o clasă abstractă se va afla în relația de agregare slabă cu clasa ce reprezintă un polinomul

După cum spuneam, un coeficient poate fi de tip real sau întreg. Prin urmare am creat clasa MonomReal și clasa MonomInt. Pentru o implementare eficientă am folosit wrapere peste int și double, astfel :

* pentru MonomReal coeficientul este de tip Double;
* pentru MonomInt coeficientul este de tip Integer.

Constructorul clasei MonomReal primește ca și parametru un număr de tip Double și inițializează variabila coef a clasei părinte. Constructorul clasei MonomInt primește ca și parametru un număr de tip Integer și inițializează variabila coef a clasei părinte.

Pentru a putea realiza inițializarea variabilei coef din clasa Monom, tipul acestei variabile trebuie să fie Number (Number - clasa abstractă din care se extinde atâtclasa Integer cât și clasa Double).

Pentru o implementare mai clară, clasa Monom (care este o clasă abstractă ) are următoarele metode:

* public getCoef();
* public getDegree();
* public setCoef();
* public setDegree();
* public abstract add(Monom y);
* public abstract subtract(Monom y);
* public abstract multiply(Monom y);
* public divide(Monom y);
* public integrate();
* public abstract derivation();
* public abstract toStringBuilder();
* public abstract checkInteger();
* public abstract negativeValue().

Clasele MonomInt și MonomReal implementează metodele abstracte din clasa Monom.

Clasa Polinom conține un ArrayList de Monoame. Am optat pentru această opțiune datorită facilităților oferite de pachetul: java.util.ArrayList. Astfel, fiecare obiect polinom va avea starea reprezentată printr-un ArrayList. Clasa Polinom are următoarele metode:

* public getCoefs();
* public getValue(double x);
* public toString();
* public derivate();
* public integrate();
* public add(Polinom Q);
* public subtract(Polinom Q);
* public multiply(Polinom Q);
* public divide (Polinom Q);
* public doPolinom(String stringP);
* private fixCoefs(ArrayList<Monom> coefs);
* public round(ArrayList<Monom> coefs);

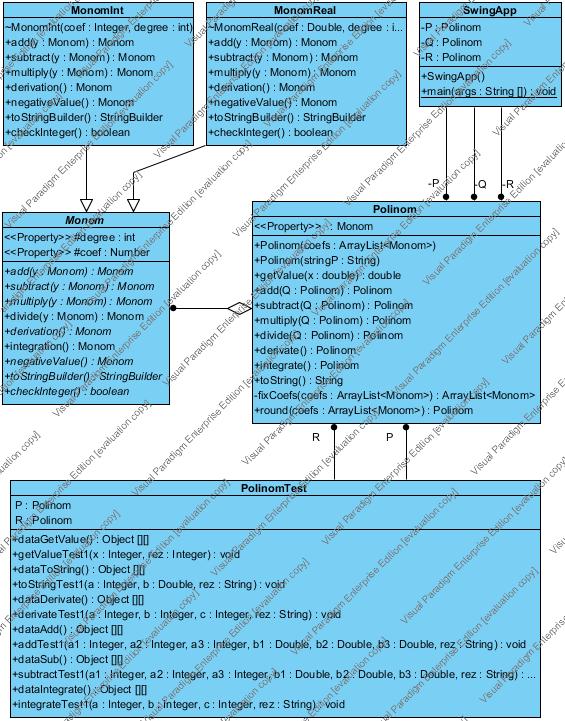
1. Cazuri de utilizare

Aplicația va putea fi folosită în mai multe contexte. De exemplu, aplicația poate fi folosită de către elevii de liceu care doresc să-și verifice propriile calcule.

Sau ar putea fi folosită pentru a sta la dispoziția oricui, deci pusă pe un site, pe internet, iar oricine dorește și are nevoie o va putea folosi oricând.

Aplicația ar putea fi de un real folos inginerilor ce au de efectuat operații cu polinoame mai ales în domeniul fizicii. Pentru acest gen de utilizare ar fi necesare modificări ulterioare, precum: realizarea unui grafic pe baza polinomului sau elaborarea unui algoritm de calculare a rădăcinilor

1. Proiectare (diagrame UML, structuri de date, proiectare clase, algoritmi și implementare , interfață utilizator)

În urma implementării a rezultat următoarea diagrama UML a claselor:

1. **Structuri de date**

Structurile de date folosite sunt:

* Clasa Polinom ce conține o listă de Monoame;
* Clasa Monom ce conține un coeficient real sau întreg și puterea variabilei necunoscute;
* Clasele MonomInt și MonomReal, care modelează comportamentul unui Monom și folosește variabilele din clasa părinte, care este clasa Monom;
* Clasa SwingApp ce conține 3 obiecte Polinoamele, dintre care, doua vor fi instanțiate de către utilizator, de la tastatură.

1. **Proiectare clase**

După cum se observă din imaginea de mai sus, clasele MonomReal și MonomInt moștenesc clasa Monom. Între clasa Monom și clasa Polinom există o relație de agregare, pentru că clasa Polinom conține o listă de Monoame.

Clasa Monom conține doua variabile, ambele cu specificatorul de acces protected, coef de tip Number și degree de tip int. Aceste două variabile sunt instanțiate de către clasele MonomInt și MonomReal.

1. **Algoritmi și implementare**

# c.1) MonomInt vs MonomReal

Comportamentul acestor două clase este îndeplinit în întregime prin moștenirea metodelor implementate în clasa Monom și implementarea metodelor abstracte din clasa Monom. Toate metodele verifică dacă coeficientul rezultat în urma operației își poate schimba tipul. De exemplu, dacă un coeficient real adunat cu un coeficient real rezultă un coeficient ce poate fi transformat în număr întreg, această conversie va avea loc cu siguranță. (ex: 2.3+3.7=6.0 , 6.0 va fi salvat ca și

coeficient de tip Integer)

* **public** Monom add(Monom y);

Această metodă funcționează doar pentru Monoame cu același grad. A fost gândită în acest mod deoarece nu are sens să adunăm două monoame cu grade diferite.

MonomInt-Această funcție verifică dacă coeficientul parametrului primit este de tip întreg. În caz afirmativ coeficientul rezultatului va fi de tip întreg. Iar, in caz negativ, oceficientul rezultatului va fi de tip real

MonomReal-Această funcție calculează rezultatul presupunând că ambi coeficienți sunt de tip real. Se verifică dacă rezultatul are coeficentul de tip intreg. În caz afirmativ se returnează rezultat cu coeficientul în formă de întreg, altfel se returnează rezultatul calculat anterior.

* **public** Monom subtract(Monom y);

MonomInt-Această funcție verifică dacă coeficientul parametrului primit este de tip întreg. În caz afirmativ coeficientul rezultatul va fi de tip întreg. Iar, in caz negativ, coeficientul rezultat în urma scăderii va fi de tip real.

MonomReal-Această funcție calculează rezultatul scăderii presupunând că ambi coeficienți sunt de tip real. Se verifică dacă rezultatul are coeficentul de tip intreg. În caz afirmativ se returnează un rezultat cu coeficient întreg, altfel se returnează rezultatul calculat anterior.

* **public** Monom multiply(Monom y);

Ambele metode returnează un nou Monom cu variabila degree egală cu suma variabilelor degree primite de la cele doău monoame ce s-au înmulțit.

MonomInt-Această funcție verifică dacă coeficientul parametrului primit este de tip întreg. În caz afirmativ coeficientul rezultatul în urma înmulțirii va fi de tip întreg. Iar, in caz negativ, coeficientul rezultat în urma înmulțirii va fi de tip real.

MonomReal-Această funcție calculează rezultatul înmulțirii presupunând că ambi coeficienți sunt de tip real. Se verifică dacă rezultatul are coeficentul de tip intreg. În caz afirmativ se returnează un rezultat cu coeficient întreg, altfel se returnează rezultatul calculat anterior.

* **public** Monom derivation();

Această metodă returnează un nou Monom conform următoarei expresii derivCoef=coef\*degree și derivDegree=degree-1. Astfel se returnează Monomul derivat.

* **public** Monom negativeValue();

Această metodă este folosită pentru realizarea unei implementări mai clare a metodei de scădere din clasa Polinom, pentru a asigura o depanare mai ușoară a codului rezultat. Este folosită pentru Monoamele din Polinomul primit ca parametru în metoda add din clasa Polinom care au un grad diferit de orice Monom din obiectul Polinom ce apelează această funcție.

* **public** StringBuilder toStringBuilder();

Această funcție transformă Monomul sub formă de String. Am decis să folosesc StringBuilder pentru a folosi mai puțină memorie, având în vedere faptul că variabilele de tip String realocă memorie la fiecare concatenare a variabilei. StringBuilder este conceput pentru a înlătura această realocare inutilă.

* **public** **boolean** checkInteger().

Metoda checkInteger() are scopul de a verifica dacă coeficientul Monomulu este de tip întreg.

MonomInt- această metodă returnează tot timpul valoarea booleană true;

MonomReal- returnează true dacă valoarea coeficientului convertită în double este egală cu valoarea coeficientului convertită în int, altfel returnează false.

# c.2)Monom

Clasa Monom conține variabilele coef și degree. Ambele variabile au specificatorul de acces de tip protected și sunt instanțiate în clasele copii. Clasa Monom este de tip abstract, prin urmare nu are nevoie de constructor.

* **public** Monom divide(Monom y)

Metoda publică divide returnează un nou obiect de tip Monom. Împărțirea a două monoame se realizează făcând împărțirea coeficienților și scăzând gradul împărțitorului din deîmpărțit. Mai întăi facem împărțirea și salvăm coeficientul rezultat sub formă de număr real. Verificăm dacă acest număr poate fi transformat în întreg. În caz afirmativ vom reutrna un obiect de tip MonomInt, altfel vom returna un obiect de tip MonomReal.

* **public** Monom integration(Monom y)

Metoda publică integration returnează un nou obiect de tip Monom. Integrarea unui Monom se face prin împărțirea coeficientului la grad+1 (astfel integrCoef = coef / (degree+1)) și incrementarea variabilei degree. Mai întăi facem integrarea și salvăm coeficientul rezultat sub formă de număr real. Verificăm dacă acest număr poate fi transformat în întreg. În caz afirmativ vom returna un obiect de tip MonomInt, altfel vom returna un obiect de tip MonomReal. Rezultat împărțirii are mari șanse să fie de tipul double și riscăm să avem multe zecimale după virgulă. Pentru îndepărtarea acestui discomfort vizual am folosit Functia Math.floor. Această funcție rotunjește valoarea în funcție de numărul la care înmulțesc și împart, de exemplu: daca vreau o zecimala implementez val\*10/10, iar daca vreau doua zecimale => val\*100/100 etc.

# c.3)Polinom

Clasa Polinom conține o listă de Monoame implementată cu ArrayList, importând din utilitarul Java pachetul ArrayList. Astfel am folosit metodele add(index),get(index),set(index, element),remove(index),size() și am înțeles mai bine cum funcționează acestă structură de date.

Metodele implementate sunt:

* public Polinom(ArrayList<Monom> coefs)

Această metodă este un constructor. Un constructor poate fi apelat folosind cuvantul rezervat new. În momentul în care se execută new, Java Virtual machine alocă spațiu pentru un nou obiect după care se apelează constructorul care urmează după new și se execută instrucțiunile din interiorul acestuia. Constructorii pot fi mai mulți, deoarece parametri primiți pot să difere. Acest prim constructor primește ca și parametru o listă de Monoame și copiază această listă în noul spațiu alocat de new. Astfel, monoamele noului Polinom creat sunt primite ca și parametru.

* public Polinom(String StringP)

Acesta este al doilea constructor, el primește ca și parametru o variabilă de tipul String ce conține defapt un șir de numere de tip întreg separate prin virgule și liniuță.

După fiecare număr ce se află pe o poziție impară (numărătoarea începând cu 1) urmează o liniuță, iar după fiecare număr ce se află pe o poziție pară urmează virgulă, deoarece numerele de pe poziția impară reprezintă coeficienți polinomului, iar numerele de pe poziția pară reprezintă puterea lui x ce are coeficientul precedent. Prin urmare Stringul are următoarea formă: coef-degree,coef-degree,coef-degree, etc.

Scopul acestui constructor este de a transforma acest String într-o listă de monoame. Pentru a realiza această operație am folosit metoda split a clasei String, mai întâi am folosit-o pentru a separa grupurile de coef-degree, astfel am apelat StringP.split(„,”), după care am apelat într-un for, din nou, metoda split - StringP.split(„-”). Folosind funcția parseInt am transformat Stringurile în numere întregi. Această transformare are loc într-un block try, care este precedat de un block catch. Dacă nu se reușește transformarea Stringului în numere întregi atunci constructorul va returna un Polinom null, adică ArrayList-ul Polinomului va fi gol. Dacă toate tranformările se desfășoară corect, atunci se apelează metoda fixCoefs(ArrayList), funcție ce este explicată mai jos. Rezultatul acestei metode se salvează în this.coefs .

* public ArrayList<Monom> getCoefs()
* public double getValue(double x)

Metodă ce calculează valoarea polinomului înr-un punct dat de utilizator.

* public Polinom add(Polinom Q)

Add este metoda ce adună două polinoame și creează un nou Polinom ce este returnat în urma parcurgerii metodei. Rezultat=this+Q.

Pentru realizarea acestei operații am folosit patru instrucțiuni for și două variabile boolean.

Mai întâi am folosit un for în for. Pentru fiecare Monom din this, parcurg fiecare Monom din Q și verific dacă există un Monom din Q care să aibă gradul egal cu Monomul din this (this.degree==Q.degree). Dacă există un astfel de monom variabila boolean sum devine true, ceea ce înseamnă că în rezultat s-a salvat suma monoamelor this și Q, care au același grad. În caz contrat this.monom nu este adăugat la rezultat. Pentru a adăuga this.monom în Polinomul rezultat verificăm dacă sum==false. Dacă verificarea returnează true atunci adaugăm this.monom la rezultat.

După parcurgerea acestui prim pas am salvat în rezultat toate Monoamele din this și monoamele din Q care au avut gradul egal cu un monom din this. Ne rămâne să adunăm monoamele din Q care nu au gradul egal cu niciun monom din P. Pentru această operație am folosit un for un for și o variabilă booleană. Pentru fiecare Monom din Q verific dacă există în rezultat un monom cu același grad. În caz că nu există, adaug la rezultat monomul din Q.

* public Polinom subtract(Polinom Q)

Subtract este metoda ce scade două polinoame și creează un nou Polinom nou ce este returnat în urma parcurgerii metodei. Rezultat=this-Q.

Pentru realizarea acestei operații am folosit patru instrucțiuni for și două variabile boolean.

Mai întâi am folosit un for în for. Pentru fiecare Monom din this, parcurg fiecare Monom din Q și verific dacă există în Q un monom care să aibă gradul egal cu Monomul din this (this.degree==Q.degree). În caz afirmativ variabila boolean sum devine true, ceea ce înseamnă că în rezultat s-a salvat diferența monoamelor this și Q, monoame care au același grad. În caz contrat this.monom nu este adăugat la rezultat. Pentru al adăuga în Polinomul rezultat verificăm dacă sum==false. Dacă verificarea returnează true atunci adaugăm this.monom la rezultat.

După parcurgerea acestui prim pas am salvat în rezultat toate Monoamele din this care nu au corespunzător ca și grad ăn Q și diferențele monoamelor din this și Q. Ne rămâne să adunăm monoamele din Q care nu au gradul egal cu niciun monom din P. Pentru această operație am folosit un for un for și o variabilă booleană. Pentru fiecare Monom din Q verific dacă există în rezultat un monom cu același grad. În caz că nu există, adaug la rezultat monomul din Q cu valoarea coeficientului său cu semn schimbat.

* public Polinom multiply(Polinom Q)

Funcția de înmulțire a două polinoame este realizată prin această metodă. Multiply este metoda ce înmulțește două polinoame și creează un nou Polinom ce este returnat în urma parcurgerii metodei. Rezultat=this\*Q.

* public Polinom divide(Polinom Q)

Funcția de împărțire a două polinoame este realizată prin această metodă. Divide este metoda ce împarte două polinoame și creează un nou Polinom ce este returnat în urma parcurgerii metodei. Rezultat=this/Q.

* public Polinom derivate()

## Derivează polinomul this și returnează rezultatul sub formă de un nou polinom. Rezultat=this’.

* public Polinom integrate()

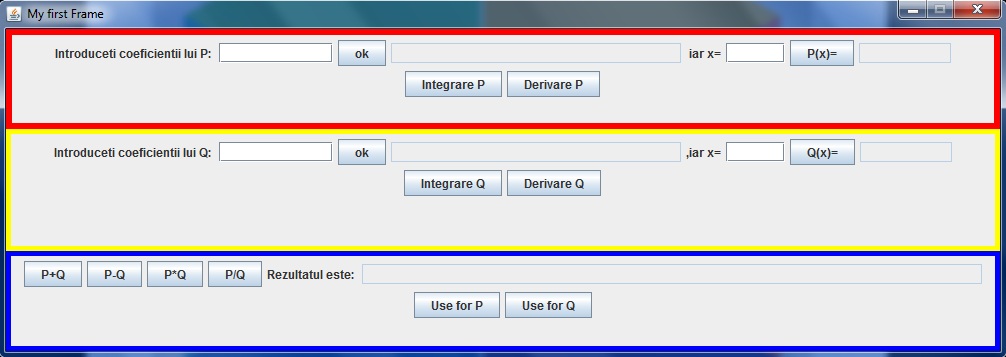
Integrează polinomul this și returnează rezultatul sub formă de un nou polinom.

* public Polinom round(ArrayList<Monom> coefs)

Această metodă folosește funcția Math.floor explciată mai sus pentru funcția de integrare. Este folosită pentru rotunjirea rezulatului, deoarece în cazut împărțirii majoritatea rezultatelor au foarte multe numere după virgulă, iar pentru o vizibilitate mai bună am optat pentru construcția acestei funcții.

* public String toString()

Metodă ce ne salvează într-un string polinomul this.

1. **Interfață utilizator**

După cum se vede, aceasta este interfața grafică care oferă utilizatorului o variantă prietenoasă de a folosi aplicația.

Pași de utilizare:

1. Introduceți coeficienții și gradele
2. Apasați butoanele ok
3. Apăsați pe butonul cu numele operației pe care doriți să o efectuați
4. Daca refolosiți rezultatul pentru a realiza alte operații, apăsați „Use for P” sau ”Use for Q” și reveniți la punctul 3
5. Rezultate

Acest proiect este cel mai amplu proiect realizat de mine în domeniul programării. Rezultatul cel mai important este faptul că mi-am îmbunătățit cunoștințele în programare orientată obiect, iar rezultatul practic este o aplicație care poate să facă operațiile de baza pe polinoame cu coeficienți reali sau întregi.

7. Dezvoltări ulterioare

Dezvoltările ulterioare ce se pot face asupra acestui proiect sunt:

1. Să introducem și alte funcții. În loc de monoame să putem avea termeni care să reprezinte una din următoarele funcții: sinus, cosinus, tangenta, arcsinus, arcosinus, ln, exponențiala. Pentru aceasta am putea să facem o interfață care să declare operațiile de adunare, scădere, inmulțire, împărțire, integrare, derivare și să poată fi folosită de toate clasele (ce reprezință funcțiile de mai sus) pentru implementare.
2. Implementarea unei funcții de aflare a rădăcinilor.
3. Introducerea Monoamelor cu coeficienți complecși.
4. Îmbunătățirea interfeței grafice pentru a putea face operații cu mai mult de doi polinomi odată.

8. Concluzii, ce s-a invățat din tema

Pot spune că am învățat multe.

Consider că cele mai importante lucruri învățate sunt: ce este polimorfismul (practic), ce sunt wraperele, mai multe detalii despre moștenire, să îmi organizez timpul, monotasking, implementare interfață cu Swing.

Folosind clasa Monom, MonomInt și MonomReal, am avut ocazia să văd cum funcționează conceptul de moștenire pe un exemplu mai amplu, implementat de mine, pot spune că mi-am îmbunătățit cunoștințele legate de downcasting, clase abstracte, metode abstracte și suprascriere.

Am învățat să îmi organizez timpul, nu în sensul în care mi-am luat liber întreaga săptămână pentru a proiecta și implementa acest proiect ci în sensul în care mi-am folosit eficient timpul în care am lucrat la proiect, o oră de muncă concentrată este echivalentă cu 3 ore de stat pe facebook, mai scris două lini de cod, intrat pe youtube, scris încă o clasă și vizualizat alte filmulețe pe 9Gag. Mi-am dezvoltat abilitatea de a mă concentra pe un singur proiect un timp îndelungat, la început părea copleșitor de multă muncă, dar după ce începi vine curajul și cheful.

Am învățat să implementez o interfață pentru utilizator cu ajutorul claselor din Swing, am înțeles cum se organizează componentele în Swing, cum aranjez butoanele și am reușit să folosesc interfața ActionListener.

9. Bibliografie

<http://en.wikipedia.org/>

<http://docs.oracle.com/>

<http://www.tutorialspoint.com>

<http://stackoverflow.com/>