**- TEMA I -**

**PROCESAREA POLINOAMELOR**

Marc Andrei – Aurel

Grupa 30224

Laborator – Tehnici de Programare

Cuprins

1. Obiectivul temei
2. Analiza problemei
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

Obiectivul temei

Un polinom este o expresie construită dintr-una sau mai multe variabile și constante, folosind operații de adunare, scădere, înmulțire și ridicare la putere constantă întreagă pozitivă. Polinoamele sunt construite din termeni numiți monoame, care sunt alcătuite dintr-un coeficient constant și un exponent constant întreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile în acel monom. În particular, gradul unui termen constant este 0.

Polinoamele au trei proprietăți elementare:

* suma a două polinoame este un polinom;
* diferența a două polinoame este un polinom;
* derivata unui polinom este un polinom;
* primitiva unui polinom este un polinom.

Toate polinoamele de o variabilă sunt echivalente cu un polinom de forma



Aceasta este considerată forma generală a polinoamelor de o singură variabilă

**Obiectivul temei** este dezvoltarea unei aplicații ce procesează polinoamele de singură variabilă cu coeficienți întregi și realizează operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare, operațiile putând fi utilizate prin intermediul interfeței.

Analiza problemei

Tema presupune în implementarea operațiilor pe polinoame cu coeficienți întregi, dar, spre exemplu, la operațiile de împărțire si integrare pot apărea coeficienți reali. Așadar, în loc să definim coeficienții de tip întreg (int), îi vom defini de tip real (float). Toate operațiile pot fi folosite prin intermediul interfeței, având un buton specializat pentru fiecare funcție. Rezultatul operației efectuate va fi disponibil într-un JTextArea, împreună cu toate rezultatele anterioare. Afișarea sa este posibilă prin apelarea unei funcții ce generează un String echivalent cu polinomul respectiv.

Algoritmii de adunare și scădere constă în parcurgerea celui de-al doilea polinom și verificarea existenței fiecărui monom din polinomul 2, în polinomul sumă (inițializat cu valorile primului polinom). În cazul în care se găsește un monom în polinomul sumă cu același grad cu monomul luat din al doilea polinom, se updatează valoarea cu suma/diferența dintre coeficienții celor două monoame. În caz contrar, este pus monomul pur și simplu în polinomul sumă.

În cazul înmulțirii, se parcurg ambele polinoame și se pune valoare coeficienților înmulțiți pe poziția sumei gradelor. Se verifică și dacă există gradul în polinomul înmulțire, deoarece, pe parcursul înmulțirii, vor rezulta monoame de același grad, care vor trebui să fie adunate.

Împărțirea este o operație mai complexă și a necesitat implementarea a două operații adiționale: înmulțirea unui polinom cu un monom și adunarea unui polinom cu un monom. De asemenea, mai folosește și operația de scădere, implementată anterior. Algoritmul folosit este Long Division, pseudocodul său putând fi găsit pe Wikipedia.

Derivarea și integrarea lucrează pe un singur polinom. În cazul derivării, se scade gradul, în timp ce la integrare, va crește. Operațiile sunt realizate prin parcurgerea fiecărui polinom și inserarea fiecărui monom după derivare/integrare, în polinom rezultat.

Pentru procesarea polinomului introdus în JTextField, am folosit o expresie regex, care returnează monoamele polinomului introdus, mai apoi fiind împărțite pentru a obține gradul și coeficientul.

Diagrama Use Case

O imagine care conține text

Descriere generată automat

Un user obișnuit va putea, conform diagramei, să introducă expresiile calculate, acestea implicit fiind stocate într-un HashMap, să efectueze operația și să vizualizeze răspunsul și toate răspunsurile operațiilor anterioare (exceptând răspunsurile șterse în cazul apăsării butonului de Clear).

Proiectare

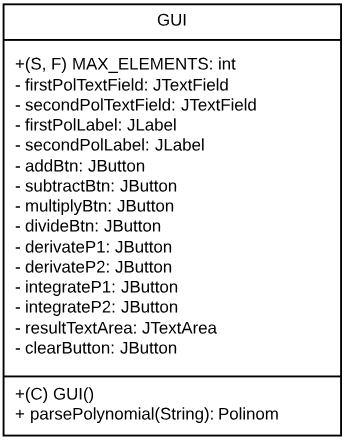
O imagine care conține chitanță

Descriere generată automatDiagrama **UML** a fost realizată cu ajutorul site-ului [www.lucidchart.com](http://www.lucidchart.com).

Clasa **Polinom** este clasa care reține polinomul propriu – zis cu ajutorul atributului său.

Clasa **Monom** este folosită pentru manipularea unui singur element dintr-un polinom, având ca atribute coeficientul și gradul.

Clasa **Operations** conține implementarea tuturor operațiilor ce pot fi efectuate pe polinoame, astfel neavând nevoie de atribute. Ca excepție, clasele *addMonomToPolinom* și *multiplyMonomToPolinom* nu reprezintă metode ce pot fi apelate din interfață, dar contribuie la implementarea operației de împărțire.

Clasa **DivisionResult** este folosită pentru stocarea polinoamelor rezultate după împărțire, anume: câtul și restul. Clasa are ca metode doar constructorul, getter-ele și setter-ele.

Clasa **GUI** este folosită pentru implementarea interfeței cu utilizatorul. Aceasta conține butoanele ce vor declanșa apelarea metodelor ce realizează operațiile cerute, zone de text pentru citirea polinoamelor și pentru afișarea rezultatelor, și un buton pentru curățarea zonei de afișare.

Structura proiectului dezvoltat este Maven, un sistem de build și administrare a proiectelor, scris în Java. Proiectul este descris într-un fișier XML, denumit POM (Project Object Model), care conține informații despre module, precum și despre dependențele proiectului. Acesta îi definește proiectului o denumire unică, formată din group ID și artifact ID.

Implementare

Am ales să stochez polinomul cu ajutorul unui HashMap și nu al unei liste de monoame, deoarece mi s-a părut mai ușoară implementarea, precum și verificarea existenței unui monom (a unei chei sau a unui coeficient) în HashMap.

Clasa ***Polinom*** – folosită pentru stocarea polinomului, reținut într-un HashMap în care gradul este cheia, iar coeficientul este valoarea.

* **Polinom**() si Polinom(HashMap<Integer, Float>) sunt constructorii clasei
* **addToPolinom**(Monom) returnează void, iar funcția ei este de a adăuga un monom în map (în polinom)
* **getPolinom**() returnează HashMap<Integer, Float>) și este getter-ul pentru polinom
* **setPolinom**(HashMap<Integer, Float>) returnează void și este setter-ul pentru polinom
* **polynomialToString**() returnează String și este metoda care transpune polinomul într-un String echivalent
* **getMaxDegree**() returnează int și funcția ei este de a obține gradul polinomului, prin căutarea maximului din HashMap
* **copyPolinom**(Polinom) returnează void și realizează o copie a polinomului dat ca parametru în instanța curentă de Polinom

Clasa ***Monom*** – folosită pentru referirea la un singur element din polinom și pentru parcurgerea simplificată a HashMap-ului ce reține polinomul.

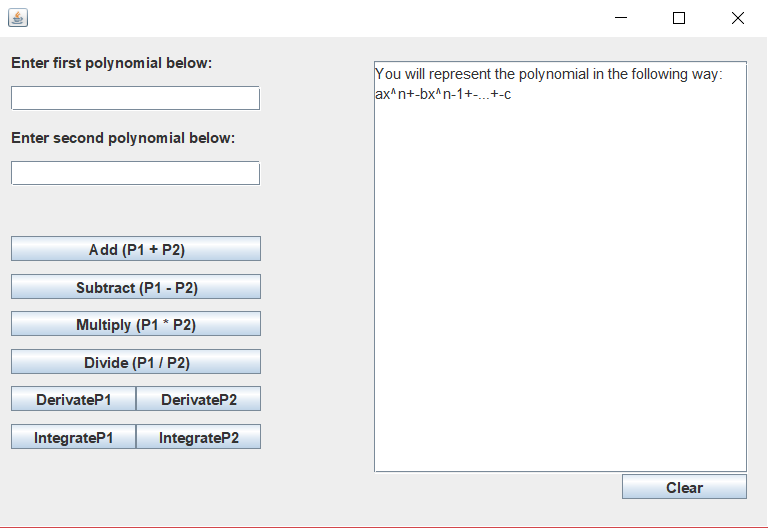
* **Monom**() și **Monom**(float, int) sunt constructorii clasei
* **getCoefficient**() returnează float și este getter-ul pentru coeficient
* **setCoefficient**(float) returnează void și este setter-ul pentru coeficient
* **getDegree**() returnează int și este getter-ul pentru grad
* **setDegree**(int) returnează void și este setter-ul pentru coeficient
* **monomToString**() returnează String și funcția ei este de a transpune monomul într-un String echivalent

Clasa ***DivisionResult*** – folosită pentru stocarea polinoamelor rezultate în urma împărțirii a două polinoame. Are ca atribute două polinoame: polinomul cât și polinomul rest.

* **DivisionResult**() este constructorul clasei
* **getR**() returnează un Polinom și este getter-ul pentru polinomul rest
* **setR**(Polinom) returnează void și este setter-ul pentru polinomul rest
* **getQ**() returnează un Polinom și este getter-ul pentru polinomul cât
* **setQ**(Polinom) returnează void și este setter-ul pentru polinomul cât

Clasa ***Operations*** – folosită pentru implementarea tuturor operațiilor pe polinom.

* **add**(Polinom, Polinom) returnează un Polinom și este metoda care realizează operația de adunare. Adunarea este realizată prin parcurgerea celui de-al doilea polinom primit ca parametru și adunarea fiecărui monom găsit la polinomul sumă.
* **subtract**(Polinom, Polinom) returnează un Polinom și este metoda care realizează operația de scădere. Scăderea este realizată asemănător adunării. Polinomul al doilea este parcurs și monoamele din el se scad din polinomul diferență, acesta fiind inițializat cu elementele polinomului 1.
* **multiply**(Polinom, Polinom) returnează un Polinom și este metoda care realizează operația de înmulțire. Se verifică monoamele din primul, respectiv al doilea polinom, două câte două și se înmulțesc corespunzător. În același timp, se verifică dacă mai există gradul rezultat după înmulțirea monoamelor, în polinomul înmulțire, pentru a fi adunate, dat fiind faptul ca vor rezulta mai multe monoame de același grad prin înmulțirea monoamelor două câte două.
* **divide**(Polinom, Polinom) returnează DivisionResult și este metoda care realizează operația de împărțire. Algoritmul implementat este Long Division, dezvoltat după pseudocodul de pe [www.wikipedia.org](http://www.wikipedia.org). Operația de împărțire a fost un motiv pentru care am ales să folosesc float pentru stocarea coeficienților, în loc de int, datorită faptului că împărțirea a două polinoame va returna, în cele mai multe cazuri, coeficienți reali. DivisionResult va reține atât câtul împărțirii, cât și restul. Sunt necesare 2 metode adiționale pentru a ajuta implementarea operației de împărțire, și anume **addMonomToPolinom**(Polinom, Monom), care realizează adunarea dintre un monom și un polinom, și **multiplyMonomToPolinom**(Polinom, Monom), care realizează înmulțirea unui monom și un polinom.
* **derivate**(Polinom) returnează Polinom și este metoda care realizează operația de derivare. Polinomul primit ca parametru este parcurs și fiecare monom rezultat va fi pus corespunzător în polinomul derivat: gradul va scădea cu o unitate, iar coeficientul va fi înmulțit cu gradul precedent derivării. Este verificat și cazul special, în care gradul unui monom este 0.
* **integrate**(Polinom) returnează un Polinom și este metoda care realizează operația de integrare. Polinomul primit ca parametru este parcurs, iar fiecare monom găsit va fi pus în polinomul integrat după cum urmează: gradul va creste cu o unitate, iar coeficientul va fi împărțit cu gradul obținut după integrare.
* **addMonomToPolinom**(Polinom, Monom), și **multiplyMonomToPolinom**(Polinom, Monom) sunt metodele utilizate la împărțire, referite mai sus.

Interfața pentru utilizator a fost creată în clasa **GUI**.

Este o interfață ușor de folosit. Utilizatorul trebuie sa introducă cele două polinoame pe care se va opera, după modelul dat în dreapta. Fiecare buton are un nume sugestiv pentru operația pe care o îndeplinește, anume: Add va realiza operația de adunare, Subtract cea de scădere și așa mai departe. În cazul în care un câmp necesar nu este completat, sau este completat în mod eronat, utilizatorul va primi un mesaj de avertizare, iar operația nu se va efectua și i se va cere introducerea corectă a datelor.

Tot aici este realizată parsarea polinomului introdus în zonele de text. În această metodă, **parsePolynomial**(String), care returnează un Polinom din String-ul primit ca parametru, este folosit un format Regex. Acest format și utilizarea sa poate fi găsită pe [www.stackoverflow.com](http://www.stackoverflow.com). String-ul va fi procesat și se vor genera toate monoamele ce îl formează. Mai apoi, monoamele vor fi procesate pentru obținerea coeficientul și gradul fiecărui monom, după care vor fi introduse în polinom. În cazul în care va fi aruncată o excepție, aceasta este tratată la apăsarea fiecărui buton (în afară de Clear), afișând un mesaj de avertizare pentru utilizator.

Rezultate

Testarea metodelor (operațiilor) a fost realizată cu **JUnit**, un framework de testare pentru Java. Pentru fiecare operație, se dau polinoame random, iar rezultatul metodelor apelate este comparat cu rezultatul corect, calculat, cu ajutorul metodei **assertEquals**, care primește, în cazul testării fiecărei operații, polinomul calculat pe hârtie, așteptat la răspuns, și polinomul primit ca rezultat în urma apelului metodei testate. În cazul în care rezultatul primit după efectuarea operației este unul diferit față de cel pe care l-am definit drept corect, testul va eșua. În caz contrar, testul va trece, arătându-ne astfel corectitudinea implementării.

**Testarea operației de adunare**

Se va testa pe polinoamele P1 = 4x^3 + 3x^2 – 2

P2 = 6x^2 + 2x^1 + x

Rezultatul va fi -1x^0+2x^1+9x^2+4x^3, rezultat corect.

**Testarea operației de scădere**

Se va testa pe polinoamele P1 = 4x^5 + 2x^2

P2 = 6x^4 - 7x^2 + 1

Se testează pentru scăderea în ambele sensuri, și se va obține:

P1 – P2 = -1x^0+9x^2-6x^4+4x^5

P2 – P1 = 1x^0-9x^2+6x^4-4x^5, ambele rezultate fiind corecte.

**Testarea operației de înmulțire**

Se va testa pe polinoamele P1 = 2x^3 + 4x^1

P2 = 1x^3 - 2x^2 + 1

Se obține rezultatul 4x^1-6x^3+4x^4-4x&5+2x^6, rezultat corect.

**Testarea operației de împărțire**

Se va testa pe polinoamele P1 = 3x^3 + 3x^2 + 2x

P2 = 2x^2 – 3x^1 + 5

Se obține rezultatul R = -21,25x^0+7,25x^1

Q = 4,25x^0 + 1,5x^1, rezultat corect.

**Testarea operației de derivare**

Se va testa pe polinomul P1 = x^5 + 5x^4 + 2x^2 – 5x – 3

Se obține rezultatul -5x^0+4x^1+20x^3+5x^4, rezultat corect

**Testarea operația de integrare**

Se va testa pe polinomul P1 = x^6 + 5x^4 + 8x^3 – 2x – 3

Se obține rezultatul -3x^1-1x^2+2x^4+1x^5+0,14x^7, rezultat corect.

Concluzii

În aplicație s-a realizat implementarea diverselor operații pe polinoame cu coeficienți întregi. Faptul că s-a folosit float pentru stocarea coeficienților, se permite operarea cu polinoame reale, de asemenea. Toate aceste operații sunt puse la dispoziție utilizatorului prin intermediul unei interfețe GUI ușor de utilizat. Aplicația are o structură simplă, fapt ce simplifică dezvoltarea ulterioară sau modificarea anumitor metode. Ca dezvoltare ulterioară, poate fi luată în considerare dezvoltarea unor algoritmi mai eficienți. În cazul de față, nu a fost necesar acest lucru, deoarece timpul în care se va efectua o anumită operație nu va fi sesizat din cauza dimensiunilor mici utilizate. De asemenea, aplicația poate fi extinsă pentru operarea pe mai mult de 2 polinoame, sau pentru operarea pe polinoame cu grad real.

Bibliografie

<https://www.lucidchart.com/>

<https://en.wikipedia.org/wiki/Long_division>

<https://stackoverflow.com/>

<https://smartdraw.com/>

https://ro.wikipedia.org/wiki/Polinom