Calculator de polinoame

Tema Numărul

1

Nume: Șerban

Prenume: Sebastian – Mihai

Grupa:30223

Specializare: Calculatoare și Tehnologia Informației

Contents

1. Obiectivul temei 3
2. Analiza problemei 4
3. Modelarea, scenarii și cazuri de utiliare 6
4. Proiectarea 8
5. Implementare 9
6. Concluzii 11
7. Bibliografie 12

1. Obiectivul temei

Principalul obiectiv al temei este acela de a implementa un calculator de polinoame coare reușește să realizeze operații minimale pe polinoame, anume: adunare, scădere, înmulțire, împărțire, derivare și integrare.

Pentru a putea îndeplini acest obiectiv a fost necesară împărțirea lui în obiective mai mici precum:

- Abstractizarea și transpunerea polinoamelor și implicit a monoamelor în paradigma programării orientate pe obiecte.
- Crearea și identificarea unor expresii regulate și a unor șabloane ce ne ajută la generarea unui polinom primit dintr-un șir de caractere.
- Implementarea operației de adunare.
- Implementarea operației de scădere.
- Implementarea operației de înmulțire.
- Implementarea operației de împărțire.
- Implementarea operației de derivare.
- Implementarea operației de integrare.
- Crearea unei interfețe grafice cât mai intuitivă pentru utilizatorul aplicației.

Pentru a ne ușura munca, vom presupune pe tot parcursul acestei documentații că valorile coeficienților pot fi doar valori întregi.

2. Analiza problemei

Pentru a putea continua, vom introduce câteva noțiuni teoretice referitoare la polinoame pe care le cunoaștem de la algebră.

Noțiuni teoretice:

Fie *K* una dintre mulțimile $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$.

Forma algebrică a unui polinom: $f = \sum_{i=1}^{n} a_i X^i$.

- Numerele $a_0,\ a_1, \dots, a_n \in K$ se numesc **coeficienți**.
- Litera X reprezintă **nedeterminata**.
- Polinomul $f = 0$ se numește **polinoul nul**.

Operații cu polinoame

Adunarea polinoamelor

- Operație asociativă, comutativă și admite element neutru polinomul nul.
- grad($f + g$) = max(grad f, grad g).

Înmulțirea polinoamelor

- Operație asociativă, comutativă și districutivă în raport cu adunarea.
- grad(fg) = grad f + grad g.

Împărțirea polinoamelor

Schema Horner:

- Se aplică pentru determinarea câtului și restului împărțirii polinomului f la polinomul g.
- Se utilizează relațiile:

$$\boldsymbol{b_n = a_n, b_{n-1} = \alpha b_n + a_{n-1}, b_{n-2} = \alpha b_{n-1} + a_{n-2}, \dots, b_0 = \alpha b_1 + a_0}$$

- Atunci câtul este polinomul $g = \sum_{i=1}^{n} b_i X^{i-1}$, iar $r = b_0$.

Derivarea polinoamelor

- Reprezintă obținerea unui polinom cu grad = grad $f - 1$.
- $\frac{d}{dx} f = n a_n X^{n-1} + \dots + a_2 X^1 + a_1$.

Integrarea polinoamelor

- Se obține un polinom cu grad = grad $f + 1$.
- $\int f \, dX = \frac{a_n}{n+1} X^{n+1} + \cdots + \frac{a_1}{2} X^2 + a_0 X$.

Scăderea polinoamelor

- Este de fapt o adunare cu semn inversat la cel de al doilea polinom.

3. Modelarea, scenarii și cazuri de utiliare

Pentru început a fost nevoie să ne reîmprospătăm cunoștințele matematice în legătură cu polinoamele, iar apoi am trecut la stadiu de abstractizare, de modelare a problemei. Am observat că polinoamele de fapt sunt o colecție de monoame, astfel am simțit nevoia să împărțim polinoamele în seturi de monoame ordonate crescător după putere.

Deoarece am ales să împărțim polionamele în monoame a fost nevoie de identificarea și separarea lor în monoame deci de identificarea unor pattern-uri ce pot fi recunoscute cu ajutorul expresiilor regulate. Însă pentru a reuși asta este mai întâi nevoie să identificăm totalitatea posibilităților de a introduce greșit un polinom.

În urma unei analize mai atente am descoperit următoarele probleme la introducerea polinoamelor de la tastatură:

- Introducerea multiplă a nedeterminantei pentru același grad:
  3xx^2 sau 3xX^2 sau 3XX^2
- Introducerea altor caractere decât cele pe care le consideram permise:
  3@^2 sau 3a^2
- Introducerea de semne multiple:
  --3x^2 sau ++3x^2
- Introducerea de polinom nou fără semn:
  3x^2x^3
- Introducerea de semn între nedeterminantă și exponent:
  3x-^2 sau 3x+^2

De asemenea, atunci când unul dintre polinoame nu este introdus corect putem trebuie să ne asigurăm că utilizatorul știe acest lucru, iar operația nu se execută. De aceea am determinat cazurile pentru care operațiile nu se pot executa în mod normal și au nevoie de atenție speciala.

Am descoperit următoarele cazuri:

- Integrare și derivare a monomului 0 care necesită un tratament special.
- Împărțirea necesită stabilirea polinomului de grad superior.

Pentru a utiliza aplicația, a fost nevoie de construirea unui GUI care să știe să opereze cu toate cele 6 operații definite. Astfel, GUI-ul ne permite să folosim aplicația în felul următor:

La intrarea în aplicație suntem întâmpinați de o interfață prietenoasă ce conține 3 câmpuri text, 6 butoane cu simboluri sugestive pe ele(+, - etc.) și 3 etichete ce indică zona unde se pot introduce polinoamele.

Pentru a putea efectua orice operație, trebuie să introducem unul sau două polinoame, urmărind următorul model:

coeficientxX^exponent+-coeficientxX^exponent+-...

!!!ATENȚIE monoamele se vor introduce fără spațiu între ele!!!

Pentru operațiile de adunare, scădere, înmulțire și împărțire, aplicația necesită 2 polinoame, iar pentru celelalte două, aplicația necesită ca introducerea să se facă pentru primul polinom.

La apăsarea butonului corespunzător operației pe care utilizator dorește să o efectueze execuția este lansată, iar dacă polinoamele nu sunt introduse corespunzător se va afișa un mesaj de eroare ce ne va indica faptul că nu a fost introdus confrom tiparului pe care l-am descris mai sus.

Aplicația este prevăzută și cu un buton de ieșire, în colțul din dreapta sus, care oprește execuția programului și închide fereastra GUI-ului.

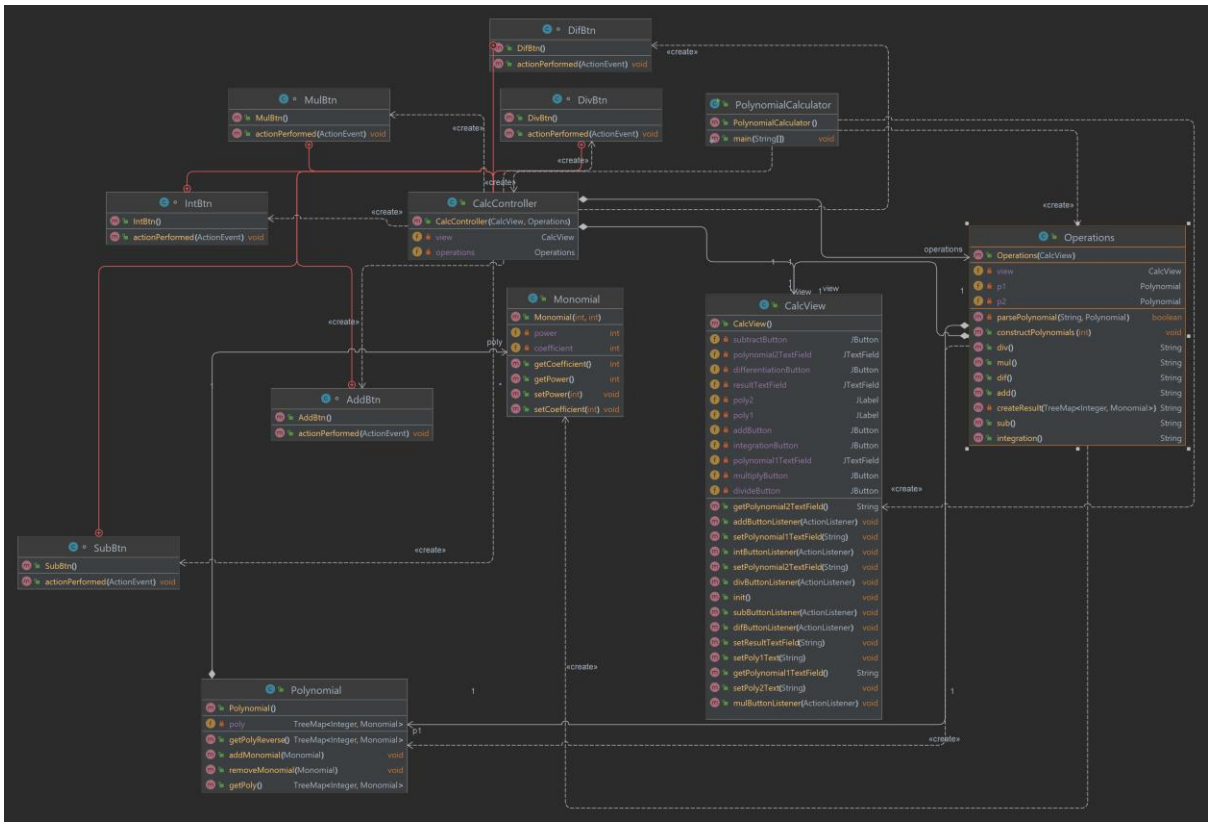
4. Proiectarea

Pentru proiectarea aplicației am ales tipologia Model-View-Controller, un tipar ce implică separarea după funcționalități a claselor, astfel obținând trei mari categorii:

- Business Logic – se ocupă cu partea de conectare a interfeței grafice cu partea de algoritmică.
- GUI – se ocupă cu elementele vizuale și interacțiunea utilizator-program.
- Data Models – se ocupă cu stocarea datelor și încapsularea lor.

În completarea design-uli MVC am ales să implementez și o categorie de test care se ocupă cu testarea metodelor și asigurarea bunei funcționări a aplicației. Testarea a fost realizată cu ajutorul testelor unitare ce au fost realizate cu JUnit.

Mai jos putem observa și diagrama UML a aplicației.



5. Implementare

Pachetul GUI

Conține clasa CalcView care reprezintă interfața grafică a aplicației, unde sunt descrise toate componentele cu care utilizatorul poate să interacționeze în timpul rulării aplicației. Această clasa moștenește clasa JFrame pentru a dobândi toate funcționalitățile necesare pentru crearea de ferestre în sistemul de operare.

```
public CalcView();
```

- Constructorul are drept scop inițializarea tututror componentelor și așezarea lor corespunzător în geografia main Frame-ului.

```
public void init();
```

- Are rolul de a reinstanția mesajele în cazul în care se întâmpină vreo eroare.

Pachetul dataModels

Conține clasele Monomial și Polynomial, două clase ce se ocupă cu stocare și încapsularea datelor pe tot parcursul execuției programului și conține get-erele și set-erele aferente acestui tip de dată.

Pachetul businessLogic

Conține clasele CalcController și Operations, două clase ce reprezintă main driver-ul aplicației.

Clasa CalcController se ocupă cu maparea și adăugarea de action listeners pentru fiecare dintre butoanele declarate în intrefața grafică. Fiecare clasă internă implementează interfața ActionListener pentru a suprascrie metoda actionPerformed pentru fiecare din cele 6 operații descrise în clasa Operations.

Clasa Operations este cea mai amplă calsă din întregul proiect. Ea deține definițiile tuturor operațiilor și partea de RegEx care parsează polinoamele introduse de la tastatură. Câteva dintre cele mai importante funcții sunt:

```
private boolean parsePolynomial(String polynomial, Polynomial
p);
```

- Această funcție se ocupă cu identificarea șabloanelor și face acest lucru folosind expresii regulate.

- Pentru început am descris toate posibilitățile de introducere greșită a unui polinom de la tastatură, lucru ce ne-a ajutat să realizăm o identificare a polinoamelor corecte cât mai aproape de ideal.
- Am obținut următoarele două pattern-uri – primul pentru descoperirea introducerii greșite, iar al doilea pentru determinarea monoamelor

```
1: ([x]{2,}|[X]{2,}|[\^]{2,}|[\+]{2,}[\-]{2,})|([xX][\+\-
]\^)|(\^\d*[xX])|(x\d+)|([^xX0-9\+\-\^])
```

```
2: ([+-]?[^-+]+)
```

- În continuare vom explica mai în detaliu ce face fiecare bucățică din expresiile descrie mai sus:
- [x]{2,} verifică dacă x apare de cel puțin 2 ori consecutiv – duplicarea nedeterminatei.
- [X]{2,} asemenea ca pentru x, doar că de data aceasta este majusculă.
- [\^]{2,} asemenea ca pentru x, doar că de data aceasta ^.
- [\+]{2,} asemenea ca pentru x, doar că de data aceasta +.
- [\-]{2,} asemenea ca pentru x doar că de data aceasta -.
- [xX][\+\-]\^ verifică dacă între nedeterminată și exponent există adăugat vreun semn.
- \^\d*[xX] verifică dacă după operația de ridicare la putere, cu sau fără exponent, se încearcă introducerea unui nou monom.
- x\d+ verifică dacă după nedeterminată există un exponent ce nu este însoțit de operatorul de ridicare la putere.
- [^xX0-9\+\-\^] verifică dacă se folosesc alte caractere decât cifre, x, X, +, - sau ^.
- Toate mențiunile de mai sus vor determina afișarea pe ecran a mesajului „Incorect polynomial”.

- [+-]? separă, dacă există, coeficientul de nedeterminată.
- [^-+]+ separă, unul sau mai multe caractere ce determină exponentul.
- Toate mențiunile de mai us ajută la determinarea corectă a monoamelor.

```
private String createResult(TreeMap<Integer, Monomial> p);
```

- Această funcție primește un polinom drept parametru și îl construiește în șir de caractere pentru a fi trimis la interfața grafică și afișat drept rezultat.

6. Concluzii

Consider că în urma acestui proiect am reușit să înțeleg mecanismul din spatele expresiilor regulate și să îmi aprofundez cunoștințele atât despre colecțiile specfice Java cât și cunoștințele matematice despre operațiile elementare pe polinoame.

Ca și îmbunătățiri, aș putea lucra mai mult la design pentru a realiza o interfață și mai intuitivă și mai rafinată, aș implementa coeficienții de tip fracționar, nu întreg, pentru a putea realiza împărțirea polinoamelor sub o formă mult mai corectă și mai aproape de valoarea reală.

De asemenea, o structură de date care să rețină un număr finit de operații ar fi de adăugat pentru ca utilizator să poate consulta operațiile anterioare și de asemenea conectarea acestei aplicații la un site web pentru a o pune la dispoziția doritoril de calculatoare de polinoame deoarece sunt un număr infim de calculatoare de polinoame care lucrează corect și rapid pe numere mari și care să fie și gratuite.

7. Bibliografie

- Java Regex | Regular Expression - javatpoint
- Use a regular expression (regex) to match polynomials (polynomial), regexpolynomial (alibabacloud.com)
- String.prototype.split() - JavaScript | MDN (mozilla.org)
- How to Replace a Value for the Given Key in the TreeMap? - GeeksforGeeks
- TreeMap get() Method in Java - GeeksforGeeks
- Polynomial Division Calculator (mathway.com)
- Remove all values from TreeMap in Java (tutorialspoint.com)
- How to get first key element from TreeMap (Sorted Map)? - Java TreeMap Programs (java2novice.com)
- UML class: Lucidchart
- Java Regular Expressions (w3schools.com)