-Documentație-

Tehnici de Programare

Sistem de procesare a polinoamelor

Nume:Kovacs Julien-Narcis

Grupa:30227

Data:4/4/2018

-Cuprins-

1.Obiectivul temei.........................................................3

2.Analiza problemei.......................................................3

3.Proiectare....................................................................

4.Implementare..............................................................

5.Rezultate......................................................................

6.Concluzii.......................................................................

7.Bibliografie...................................................................

**1.Obiectivul temei**

Obiectivul temei, cum spune și cerința, este acela de a coda si implementa un sistem de procesare a polinoamelor("calculator de polinoame").Sistemul trebuie să fie capabil de a efectua operații simple asupra unuia sau asupra a doua polinoame si anume:

-adunare

-scădere

-înmulțire

-împărțire

-derivare

-integrare

**2.Analiza problemei**

Pentru a realiza sistemul de procesare al polinoamelor, problema trebuie abstractizată în așa fel încât produsul final să fie cat mai user-friendly atât pentru client, utilizatorul de zi cu zi, cât si de cel ce se ocupă de mentenanța programului cu pricina.În acest caz, cum scrie și mai sus, problema a fost "ruptă" in mai multe bucăți, în 2 categorii: 1.Interfața grafică, care, într-un final, a ieșit cât de simplă si user-friendly se poate; 2.Partea funcțională, operațiile și logica din spatele lor.

Datele de intrare sunt introduse de către utilizator în interfața grafică. Din păcate, există anumite restricții de introducere a datelor de care utilizatorul trebuie să țină cont precum:

Toți termenii polinomului, toate monoamele trebuie introduse împreună cu puterea și coeficienții lor, indiferent daca termenii sunt liberi.

Aplicația a fost realizată si dezvoltată în mediul de dezvoltare Eclipse, în limbajul Java.

**3.Proiectare**

**3.1 - Diagrame UML**



UML (Unified Modelling Language) este, aşa cum îi spune şi numele, un limbaj. El este un limbaj pentru modelare folositor în domeniul software, la realizarea documentelor de specificaţii şi, în general, bun pentru comunicarea între ingineri.UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

3.2 - Decizii de proiectare

Am decis de la bun început să folosesc clase separate pentru Monoame si Polinoame, asta pentru a nu incărca programul inutil.Nu a fost proiectată o clasă separat pentru operații, acestea au fost implementate in clasele Monom, respectiv Polinom.Interfața a fost proiectată separat de cele două clase.

3.3 - Structuri de date

Structura de date folosită a fost Array List-ul, care are ca și elemente Monoame(la rândul lor formate din coeficient si putere).A fost de dorit utilizarea acestei structuri de date deoarece este o structură cu o lungime variabilă, putând sa crească sau să scadă dinamic în dimensiuni.

3.4 - Interfața utilizator

După cum scrie si mai sus, interfața utilizator este una super simpla si user-friendly. Aceasta este compusă din Frame(planul principal unde își au locul widget-urile), cele 6 butoane cu ajutorul cărora utilizatorul se poate folosi de program pentru a realiza operațiile și anume:

-Butonul + ce efectuează adunarea celor două polinoame;

-Butonul – ce efectuează scăderea celor două polinoame;

-Butonul \* ce efectuează înmulțirea celor două polinoame;

-Butonul / ce nu face nimic ;

-Butonul Derivare ce derivează primul polinom;

-Butonul Integrare ce integrează primul polinom;

Nu în ultimul rând este introdus butonul CLEAR, care, după cum îi spune și numele, șterge informația din cele trei field-uri de date, primele două pentru date de intrare și ultimul pentru date de ieșire.

A fost evitată folosirea Label-urilor, informațiile necesare utilizatorului fiind prezente in Field-uri la deschiderea programului.



3.5 - Modul de tratare al erorilor

În ciuda faptului că timpul alocat rezolvării erorilor a fost mult, mult mai mare decât timpul alocat proiectării, programul mai dispune de câteva erori si anume restricțiile de input ale utilizatorului. Aceste restricții apar în urma proastei parsări ale String-ului în Polinom. S-a incercat utlilizarea regex-urilor, dar fiecare încercare s-a terminat într-un eșec. Citirea a fost facută superficial, prin parsarea cat de mult posibilă a String-ului din input.

3.6 - Algoritmii implementați

-Monom(int coeficient, int grad) ->constructorul Monom-ului, care construiește un Monom cu coeficientul și puterea ce-i corespund;

-getCoeficient(),getPutere() -> getter-urile clasei Monom, care returnează puterea si coeficientul Monom-ului;

-toStringM() ->parsează Monom-ul într-un string pentru a putea fi afișat;

-adunareMonom(Monom m)->face adunarea dintre Monom-ul curent si Monom-ul m;în cazul in care puterile celor două Monoame corespund, un al treilea Monom, Monom-ul rezultat primește suma coeficienților celor două Monoame, păstrând puterea unuia dintre ele, indiferent care;în cazul în care puterile celor două Monoame nu sunt egale, este returnat Monom-ul cu gradul mai mare;

-scadereMonom(Monom m) ->face scaderea dintre Monom-ul curent si Monom-ul, analog cu adunarea;

-inmultireMonom(Monom m)->face înmulțirea celor două Monoame, înmulțindu-se coeficienții si adunându-se puterile;

-derivareMonom-> derivează Monom-ul, coeficientul acestuia înmulțindu-se cu puterea sa, puterea sa fiind decrementată cu 1;

-integrareMonom->integrează Monom-ul după formula de integrare a unui Monom;

-Polinom () -> constructorul care transformă polinomul în ArrayList ;

-Polinom ( ArrayList<Monom> Polinom ) -> constructorul care atribuie polinomului ArrayList-ul primit ;

-toStringP()->metoda care care se ocupă cu parsarea din Polinom în String a Polinom-ului;

-isInteger(String s)->metodă ce verifică dacă un String este sau nu numeral;

-parse(String s)->metodă care are ca rol parsarea unui String s într-un Polinom;s-a încercat, așa cum scrie și mai sus, parsarea cu ajutorul Regex-urilor.

-adunarePolinom(Polinom p)->metodă ce sa ocupă cu adunarea la this.polinom Polinomul p cu ajutorul a trei iteratori si a metodei adunareMonom(Monom m);

-scaderePolinom(Polinom p)->analog metodei de mai sus;

-derivarePolinom(Polinom p)->metoda ia fiecare Monom din Polinomul p si aplică pe urmă metoda derivareMonom();

-integrarePolinom(Polinom p)->analog metodei de mai sus;

4 - Implementare

4.1 - Clase

Pentru realizarea acestui proiect s-au folosit un total de patru clase:

-clasa Monom -> este cea mai importantă clasă, stocheaza informația cu ajutorul căreia se pot realiza polinoamele.Intră in compoziția ArrayList-ului din clasa de Polinoame.Clasa primește 2 parametri, coeficientul si puterea;clasa are setter-e si getter-e pentru putere și pentru coeficienți.Clasa are 5 metode care reprezintă operații și anume adunareMonom(m),scadereMonom(m),inmultireMonom(m),derivareMonom(m),integrareMonom(m), rolul carora este explicat mai sus;în plus, mai este prezentă si metoda toStringM(), cu rolul de parsare;

-clasa Polinom ->este clasa care primește ca si argument unul sau mai multe Monoame;clasa are un singur setter;metodele prezente în această clasă sunt metodele cu ajutorul cărora se pot reazila operațiile elementare pe polinoame, adunarePolinom(p),scaderePolinom(p),inmultirePolinom(p),derivarePolinom() si nu în ultimul rând integrarePolinom();funcționarea acestora este descrisă în mare mai sus;nu în ultimul rând, sunt prezente cele două metode ce se ocupă cu parsarea si "deparsarea" Polinoamelor, adică parse(String s), metodă ce parsează un String s într-un Polinom rezultat si toStringP(), metodă ce transformă polinomul p într-un String s;

-clasa GUI->clasa ce are ca scop crearea și implementarea interfeței utilizator;componentele acestei clase, în ordinea numărului de pe tricou interfeței sunt:

-Frame ->Frame-ul;

-Polinom1(JTextField)->cu rolul de a primi datele de intrare ale primului polinom(presetat cu textul :"Introduceți primul polinom:");

-Polinom2(JTextField)->cu rolul de a primi datele de intrare ale celui de-al doilea polinom(presetat cu textul:"Introduceți cel de-al doilea polinom:");

-BAdunare(JButton)-> cu rolul de facilita operația de adunare;în momentul în care este apăsat, iar în primele două JTextField-uri sunt două polinoame valide, cele două String-uri din JTextField-uri vor fi parsate in Polinoame, ca pe urmă, să se execute operația de adunare a polinoamelor, a cărei rezultat va fi afișat în JTextField-ul rezultat;

-BScadere(JButton) ->cu rolul de facilita operația de scădere;în momentul în care este apăsat, iar în primele două JTextField-uri sunt două polinoame valide, cele două String-uri din JTextField-uri vor fi parsate in Polinoame, ca pe urmă, să se execute operația de scădere a polinoamelor, a cărei rezultat va fi afișat în JTextField-ul rezultat;

-BÎnmultire(JButton)->cu rolul de facilita operația de înmulțire;în momentul în care este apăsat, iar în primele două JTextField-uri sunt două polinoame valide, cele două String-uri din JTextField-uri vor fi parsate in Polinoame, ca pe urmă, să se execute operația de înmulțire a polinoamelor, a cărei rezultat va fi afișat în JTextField-ul rezultat;

-BDerivare(JButton)->cu rolul de a facilita operația de derivare;în momentul în care este apăsat, iar în primul JTextField este un polinom valid, String-ul din Field va fi parsat într-un Polinom, ca pe urmă, să se execute operația de derivare a polinomului, a cărei rezultat va fi afișat în JTextField-ul rezultat;

-BIntegrare(JButton)->cu rolul de a facilita operația de integrare;în momentul în care este apăsat, iar în primul JTextField este un polinom valid, String-ul din Field va fi parsat într-un Polinom, ca pe urmă, să se execute operația de integrare a polinomului, a cărei rezultat va fi afișat în JTextField-ul rezultat;

-CLEAR(JButton)->cu rolul de a șterge input-urile din cele 3 JTextField-uri;

-rezultat(JTextField)->cu rolul de a arăta utilizatorului rezultatul operației;

5 - Rezultate

Aplicația de simulare a unui calculator polinomial poate face următoarele operații:

-Adunare;

-Scădere;

-Înmulțire;

-Derivare;

-Integrare;

6 - Concluzie

În concluzie, acest proiect m-a ajutat să-mi aprofundez cunostințele despre POO și despre calculul cu polinoame;a fost un proiect care, spre surprinderea mea, nu a dat chiar așa bătăi de cap;implementarea proiectului a mers în mare parte cursiv, cu mici piedici pe drum;

În orice caz, există o sumedenie de îmbunătățiri care pot fi aduse programului.

7 - Bibliografie

[www.youtube.com](http://www.youtube.com/)

[www.stack-overflow.com](http://www.stack-overflow.com/)

[www.google.com(scriu](http://www.google.com(scriu/) asta deoarece au fost o grămadă de site-uri ce m-au ajutat pe care le-am găsit datorită acestui motor de căutare)