Tehnici de programare

Documentație pentru tema 1

Miklós Balázs

Grupa 30224

UTCN, Calculatoare

Sistem de procesare a polinoamelor

1. Obiectivul temei

Programul este făcut pentru a face operații cu polinoame: adunare, scădere, înmulțire, împărțire, derivare și integrare. În plus mai este implementat o interfață grafică pentru utilizatori. Programul este făcut așa ca să fie ușor de modificat sau îmbunătățit și cu alte operații dacă este cazul.

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

Polinomii sunt foarte impotanți ca obiecte în matematică. Un polinom este o algebrică constituită din mai multe monoame, legate între ele prin semnul plus sau minus, suma algebrică a mai multor monoame

Programul poate să realizeze orice operație enumerate anterior între polinoame, deci este folositor, poate fi numit și ca un *"Polinom Calculator".*

La adunarea și scăderea polinoamelor algoritmul este următorul: polinomul rezultat va avea gradul egal cu maximul gradurilor dintre cele două polinoame dată de utilizator. După care polinomul rezultat este încărcată cu coeficienți 0 și cu graduri descrescător de la gradul maxim până la 0. Parcug polinomul rezultat și cele două polinoame cu câte un ciclu for și adaug sau scad monoamele la polinomul rezultat. În final parcurg noul polinomul și dacă găsesc vre-un monom în el cu coeficient 0, șterg monomul respectiv și returnez un nou polinom ca rezultat.

La înmulțire gradul noului polinom este egală cu suma gradelor polinoamelor înmulțite. Și aici creez un nou polinom cu monoame de coeficienți zero, și încep să încarc cu monoame primite prin înmulțirea fiecărui monom din primul polinom cu fiecare monom din al doilea polinom. Daca cumva rămân niște monoame cu coeficienți 0 în polinomul rezultat, iarăși le șterg ele. În final rezurnez un nou polinom rezultat.

La operația de împărțire un polinom cu altul, prima dată verific daca pote fi împărțită, adică dacă gradul primului polinom (care va fi împărțită) este mai mare sau egal cu gradul celui de al doilea polinom (împărțitorul). Dacă da, atunci cu ajutorul operațiilor făcute anterior (adunare, scădere, înmulțire) efectuez operația. Împart pimul monom din primul polinom cu monomul cu cel mai mare grad din al doilea polinom, restul primesc prin înmulțirea rezultatului temporal cu al doilea polinom. După ce scad din primul polinom restul, și în continuare lucrez cu restul. Algoritmul continuă până când gradul restului este mai mic decât gradul împărțitorului, sau 0. În final returnez rezultatul într-un nou polinom.

La derivarea întotdeauna derivez primul polinom în modul următor. Parcurg fiecare monoame din polinom, și înmulțesc gradul monoamelor cu coeficienți si scad 1 din grraduri. Am grijă ca în cazul în care monomul este o costantă, derivata să fie 0. Rezultatul returnez într-un nou polinom.

La integrarea tot așa operez ca și la derivarea, doar aici am grijă cu împărțirea cu 0 în cazul monomului x^-1. În cazul aceasta nu efectuez împărțirea ci afișez lnx la reuzltat. Polinomul rezultat tot returnez într-un nou polinom.

Mai este făcuă o funcție care convertește un String dată de utilizator în forma corectă la un polinom, care poate fi ultilizat la operațiile.

Aplicația poate fi folosită foarte ușor de către un utilizator prin GUI. El conține doar două fied-uri de text ca imput și un output textfield. Lângă acestea mai conține niște bunoate cu semnul operațiilor pe care dorim sa realizăm cu polinoame.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

Proiectul conține trei pachete: un pachet pentru interfață, un pachet pentru operațiile, și testele individuale și un pachet cu o clasă pentru a testa corectitudinea tuturor operațiilor făcute (adunare, ..., integrare) cu ajutorul unui test Junit Test Suite, unde sunt grupate testele mici de la pachetul anterior.

Pachetul interfață conține o singură clasă numită Window.java. Cu ajutorul acestei clasă creez fereastra necesară pentru interfața grafică de utilizator. De fapt aici este definită un Jframe, niște Jpanel-uri și JtextField-uri și Jbutton-le pe care adaug în ele. Constructorul este făcut așa ca scriind new Window(), apare fereastra. În plus mai este făcut o metodă numit AddButtonListener care adaugă butoniilor niște acțiune ce trebuie să facă ei după semn. Butonii și TextField-urile sunt variabilele de instanță, ele are niște gettere.

Pachetul model conține 4 clase principale: Monom, Polinom, Operații și Test.

În clasa Monom este descris structura unui monom prin coeficient și grad, și sunt create niște gettere și settere pentru acestea.

Clasa Polinom definește o listă de monoame (ArrayList), un getter și un setter și o metodă care afișează conținutul unui polinom.

Clasa operații este făcuă pentru a descrie operațiile enumerate anterior între polinoame și niște metode ajutătoare pentru a ușura operațiile de genul sortare_polinom, simplificare_polinom etc. Clasa aceasta nu conține nici-o variabilă de instanță, toate metodele sunt statice, folosesc ele incluzând clasa Operații în clasele unde folosesc operațiile.

Clasa Test are o singură variabilă de instanță (Window w). În clasa aceasta sunt definite niște clase care implementeze ActionListener. Cu metoda ButtonListener din clasa Window, instalez fiecărui button la constructorul Clasei câte o acțiune. În metoda main, când scriu new Test() apare fereastra de interfață grafică.

În pachetul aceasta mai sunt niște clase Junit Test Case (AddTest, ScadTest, MulTest, DivTest, DerivTest, IntegTest). De aceea am pus în această clasă pentru că metodele de operații sunt

definite ca statice, și daca includ clasa operații într-o altă clasă care nu este în pachetul model, atunci clasa respectivă nu o sa cunoască metodele. În clasele acestea testez corectitudinea adunării, scăderii, înmulțirii, împărțirii, derivării și integrării cu Junit.

În pachetul testing am o clasă numită AllTests, unde grupezi testele prezentate anterior, și rulez toată testele odată (JUnit Test Suite).

De fapt toată structura a proiectului seamănă cu structura MVC (Model, View, Controller) doar ca la mine Modelul este clasa Operații care este inclus în clasa Controler (Test). Proiectul este în format Maven, testele sunt făcute cu JUnit.

4. Implementare

Pachetul interfața

Clasa *Window* conține câmpurile f(frame), mainPanel alcătuit din mainPanel1 și mainPanel2. MainPanel1 este alcătuit din câmpurile panel1, panel2 și panel 3. Panel1 conține primul JtextField f1, panel2 conține al doilea JtextField f2 și panel3 conține f3. MainPanel2 conține cele 6 butoane (JButton): plus, minus, mul, div, deriv, integ.

Constructorul: public Window()

Aici este construit fereastra și sunt apelate metodele ajutătoare pentru construirea interfeței:

-public void createPanels()- crează și separă interfața în diferite părți grafice

-public void createAndAddTextFields()- adaugă și crează Field-urile de texte.

-public void addMainPanel1()- adaugă panoul care conține textfield-urile.

-public void createAndAddButtons()- Crează și adună butonii la al doilea panel principal.

Aceste metode sunt apelate în constructorul. Clasa mai conține gettere pentru variabilere enumerate anterior, și o metodă de a instala butoanele cu diferite acțiuni:

-public void addButtonListener(JButton a,ActionListener b).

Pachetul model:

Clasa *Monom* conține variabilele de instanță coeficient și grad. Clasa conține getters și setters pentru variabilele. Implementează interfața Comparable și conține metoda:

-public int compareTo(Monom o)- care compară două monoame după grad, și returnează un număr pozitiv, 0 sau negativ. Este folosit pentru sortarea polinoamelor.

Clasa *Polinom* are o singură variabilă de instanță ArrayList <Monom> polinom, care are un getter și un setter. Constructorul public Polinom() crează un nou Polinom de ArrayList de Monoame.

-public void afisarePolinom()- afișează polinomul la care este apelat.

Clasa *Operații* nu are variabile de instanță, conține următoarele metode:

-static int max(int a,int b)- returnează numărul mai mare a sau b

-static Polinom polinomSort(Polinom P)- sortează polinomul primit ca parametru după grad descrescător. Folosește metoda CompareTo(). Returnează un polinom.

-static Polinom removeZeroes(Polinom p)- șterge dintr-un Polinom monoamele care au coeficient 0 și returnează un polinom.

-static Polinom simplificare(Polinom P)- dacă cumva după rezultatul unei operație mai ales la înmulțire apare într-un polinom două sau mai multe monoame cu același grad, acestea vor fi adunate. Returnează un polinom.

-static Polinom addP(Polinom p1,Polinom p2)- adună cele două polinoame primit ca parametru și returnează un nou polinom.

-static Polinom scadP(Polinom p1,Polinom p2)- scade cele două polinoame primit ca parametru și returnează un nou polinom.

- static Polinom polMul(Polinom p1,Polinom p2)- înmulțește cele două polinoame primit ca parametru și returnează un nou polinom.

- static String polDiv(Polinom p1,Polinom p2)- împarte polinomul p1 cu p2 și returneză un String ca rezultat cu câtul și restul

- static Polinom derivare(Polinom p)- derivează polinomul p1 și returnează un nou polinom

- static Polinom integrare(Polinom p)- integrează polinomul p1 și returnează un nou polinom

-static Polinom prelucrareString(String s)- convertește stringul s primit ca parametru scris în forma corectă într-un Polinom și returnează (Este folosită la interfața la textField-uri).

- static String convPolToString(Polinom p)- convertește un polinom la un String și returnează

Clasa *Test* are variabila de instanță w de tipul Window.

-Constrctorul public Window(): Crează interfața cu w=new Window() și instalează butoanele de la interfață cu niște acțiuni folosind metoda ButtonListener.

Conține următoarele clase micie:

- private class PlusListener implements ActionListener

-private class MinusListener implements ActionListener

-private class MulListener implements ActionListener

-private class DivListener implements ActionListener

-private class DerivListener implements ActionListener

-private class IntegListener implements ActionListener

Clasele acestea descriu comportamentul butoanelor la interfață sunt legate la butoane.

Pechetul mai conține clasele *AddTest, ScadTest, MulTest, DivTest, DerivTest, IntegTest* care sunt teste JUnit (JUnit Test Case) individuale pentru cele șase operații. Ele conțin metoda:

-public void test(): efectuează testul la o singură operație, culoarea verde arată dacă testul a fost cu succes, culoarea verde arată dacă testul a fost fără cucces.

Pachetul testing conține clasa *AllTests* care rulează testele JUnit enumerate anterior odată cu Suite.class. (JUnit Test Suite).

5. Rezultate

Testarea programului este făcută cu JUnit în felul următor: În fiecare operație am dat două sau dacă este cazul un polinom, și am dat și un string ca rezultat. Am luat un al treilea polinom unde țin rezultatul operației, după ce convert rezultatul într-un alt string cu funcția convPolToString. Stringul dat anterior și stringul primit ca rezulat compar în JUnit Test Case cu funcția AssertEquals.

Polinoamele care folosesc la testare:

P1: 4*x^5- 3*x^2+ 1*x^1

P2: -2*x^4+ 4*x^2+ 2*x^0

Testarea adunării: Rezultatul este 4.0*x^5+ (-2.0)*x^4+ 1.0*x^2+ 1.0*x^1+ 2.0*x^0 - rezultat correct.

Testarea scăderii din polinomul P1 polinomul P2: rezultatul este

4.0*x^5+2.0*x^4+ (-7.0)*x^2+1.0*x^1+ (-2.0)*x^0- rezultat corect.

Testarea înmulțirii: rezultatul este

(-8.0)*x^9+ 16.0*x^7+6.0*x^6+ 6.0*x^5+ (-12.0)*x^4+ 4.0*x^3+ (-6.0)*x^2+ 2.0*x^1- rezultat corect.

Testarea împărțirii polinomului P1 cu P2: rezultatul este

Cat:(-2.0)*x^1 Rest :8.0*x^3+ (-3.0)*x^2+ 5.0*x^1- rezultat corect.

Testarea derivării polinomului P1: Rezultatul este 20.0*x^4+ (-6.0)*x^1+ 1.0*x^0- rezultat corect.

Testarea integrării polinomului P1: rezultatul este 0.6666667*x^6+ (-1.0)*x^3+ 0.5*x^2- rezultat corect.

Deci toată testele au finalizat fără eroare, programul funcționează corect.

6. Concluzii:

Tema aceasta cu procesarea polinoamelor a fost folositor pentru mine ca să repet chestile de bază în limbajul de programare Java și ca și să fac un program care poate să opereze cu polinoame. Un program ca și aceasta poate fi folositor și în viață, mai ales la matematică când avem de lucrat cu polinoame. De exemplu dacă este vorba de împărțirea polinoamelor unde pe foaie utilizatorul are mai mult de lucrat, aici pur și simplu scrie în interfața grafică polinoamele și rezultatul apare în milisecunde. Deci interfața grafică de utilizator ușurează foarte mult lucrul cu polinoamele. Polinoamele folosesc și la aproximarea altor funcții, cum ar fi sinus, cosinus, și exponențiala. Sau și în alte domennii ale matematicii sau programării, proiectul aceasta poate fi folosită și poate să ajute.

Limbajul de programare Java pote fi beneficial și folositor mai ales când utilizatorul lucrează cu obiectele, și trebuie să descrie obiectele prin trăsăturile sale (variabile de instanță)sau prin comportamentul lor (metode ajutătoare ale obiectelor). Clasele și obiectele pot fi legate foarte ușor fiind creat astfel o ierarhie dintre clase. Java este un libaj ușor de învățat chiar și pentru cei care nu au experiență în domeniul programării. Prin învățarea Java avem acces la reale oprtunități de carieră.

7. Bibliografie:

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

https://www.mathsisfun.com/algebra/polynomials-division-long.html

8. Diagrama UML