Documentatie

Tema numarul 1

Calculator de polinoame

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei este crearea unei aplicatii pentru modelarea unui set de date introdus sub forma unui polinom si care sa poata realiza operatii de baza asupra acestora: adunare, scadere, inmultire, impartire, derivare si integrare.

Printre obiectivele secundare se numara:

- Crearea unui model prin intermediul claselor Monome si Polynomial care vor fi folosite pentru transformarea textului introdus de utilizator intr-o structura cu care putem lucra si pentru implementarea metodelor folosite la calculul rezultatelor;
- Crearea unei interfete grafice prietenoase;
- Legarea modelului la interfata grafica printr-un controller;

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

Prin analiza problemei, ne referim la un prim set de operatii si proprietati prin care incercam sa gasim diferite însușiri și comportamente ale proceselor necunoscute. Programarea orientata pe obiect ne ofera un avantaj clar, componentele constituente putand fi gasire relativ usor, deoarece au o legatura directa cu lumea reala (actiuni, obiecte).

Pornind de la specificatia proiectului, cautam:

- substantive, care vor deveni clase;
- verbe, care vor deveni metode;

Dupa acest pas, avem o idee generala asupra problemei. Pasul următor constă în descrierea funcțională a acesteia. Ce trebuie să facă aplicația?

Deoarece programul va putea fi accesat de un număr ridicat de persoane, interfața cu utilizatorul devine punctul de pornire al proiectului. Ea trebuie să permită comunicarea utilizatorului cu aplicația.

În cazul Calculatorului de Polinoame, se cunoaște că aplicația trebuie să implementeze operațiile algebrice elementare, deci trebuie să existe minim două câmpuri de intrare in care se vor introduce polinoamele. Acestea vor fi introduse sub forma te text, fiind preluate in string-uri de catre aplicatie, ca mai apoi sa fie transformate in structuri de tip polinom.

Scenarii

Pentru utilizarea normala a programului sunt necesari urmatorii pasi:

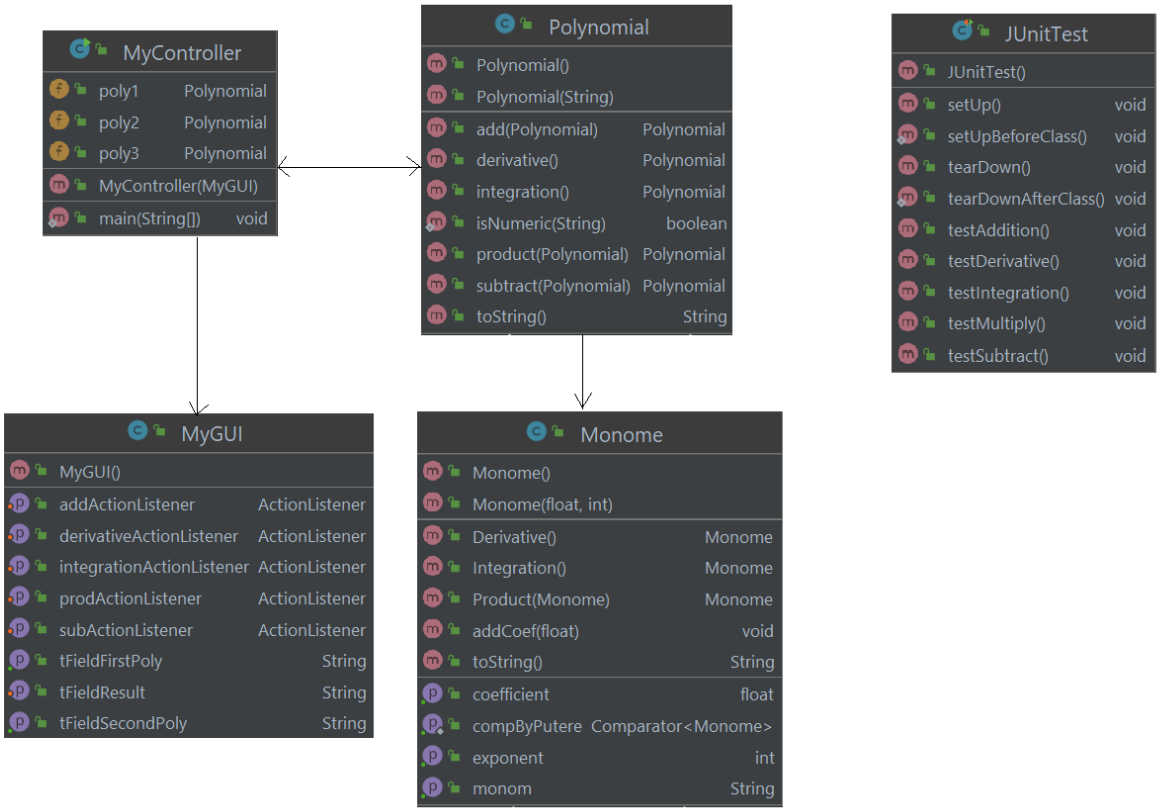
- Introducerea polinoamelor in format clasic, in ordine descrescatoare a puterilor.
- Aplicarea operatiilor prin apasarea butoanelor
- Rezultatul va fi vizibil intr-un TextField.

Cazuri de utilizare

Dupa ce stabilim modul de intrare a datelor, vorbim de urmatoarele:

- Suma, diferenta, produsul si impartirea a doua polinoame
- Derivarea si integrarea unui polinom
- Calcularea rezultatelor numerice

3. Proiectare



Calculatorul de polinoame contine 5 clase:

1. Clasa MyGUI - cu ajutorul ei se creează interfața grafică folosită în proiect. Tot aici se instanțiază și asculatătorii, care au rolul de a surprinde eventualele acțiuni ale utilizatorilor prin intermediul evenimentelor.
2. Clasa MyController – prin intermediul acestei clase sunt luate datele de catre ascultatori si se activeaza modulele corespunzatoare in functie de operatia aleasa.
3. Clasa Polynomial – permite convertirea string-ului introdus intr-un polinom, sub forma de lista de monoame. Tot aici se afla si operatiile fundamentale: adunare, scadere, inmultire, derivare si integrare.

4. Clasa Monome – aceasta clasa este fundamentala proiectului, deoarece defineste structura de monom, ce contine un coeficient real si o putere, numar intreg pozitiv.
5. Clasa JUnitTest – care este folosita pentru a testa buna functionare a operatiilor.

Pe lângă tipurile de date primitive existente în clasele de mai sus, se folosesc și colecții. Una din cele mai importante colecții se află în clasa Polinom, ce conține o listă de monoame.

Modul de proiectare a claselor este in stilul MVC pattern:

Polinom si Monom reprezintă clase model, incapsuland datele si functionalitatile

MyGUI joacă rolul unui View, afisand informatia pe ecran

MyController este Controller-ul , facand legatura intre view si clasele model

4. Implementare

1. Clasa MyGUI

Contine un Jframe pe care se construieste intreaga interfata.

Interfata cu utilizatorul este formata partea de control, reprezentată de câmpurile și butoanele prin care utilizatorul introduce datele si partea de afișare a rezultatului.

Am folosti doua Jlabel-uri

Exista cateva secvente de cod care se repeta cu mici diferente. De exemplu, cele sase butoane corezpunzatoare operatiilor si cele doua textField-uri in care utilizatorul introduce polinoamele.

Butoanele sunt declarate astfel:

```
62          buttonAdd = new JButton( text: "ADD");
63          buttonAdd.setBackground(new Color( r: 30,  g: 144,  b: 255));
64          buttonAdd.setFont(new Font( name: "Arial", Font.PLAIN,  size: 18));
65          buttonAdd.setBounds( x: 10,  y: 175,  width: 230,  height: 25);
66          MainPanel.add(buttonAdd);
```

Iar TextFieldu-rile:

```
55          tFieldSecondPoly = new JTextField();
56          tFieldSecondPoly.setBackground(new Color( r: 30,  g: 144,  b: 255));
57          tFieldSecondPoly.setFont(new Font( name: "Arial", Font.PLAIN,  size: 18));
58          tFieldSecondPoly.setBounds( x: 175,  y: 75,  width: 325,  height: 30);
59          MainPanel.add(tFieldSecondPoly);
60          tFieldSecondPoly.setColumns(10);
```

Pentru afisare, am folosit tot un TextField, dar care nu este editabil de catre utilizator.

Metodele setXActionListener contine ascultatorii ce vor fi utilizati in MyController.

```
117     public void setProdActionListener(final ActionListener e){
118         buttonMultiply.addActionListener(e);
119     }
```

2. Clasa MyController

Este clasa care gestioneaza toate actiunile venite din partea utilizatorilor si le coreleaza cu functionalitatile existente. In aceasta clasa avem o serie de sublcase care implementeaza interfata ActionListener si metoda actionPerformed(), fiecare dintre ele fiind responsabile cu efectuarea unei operatii selectate de utilizator.

In aceasta clasa vor exista tot timpul cele doua String-uri introduse de utilizator sub forma de text, in cele doua TextField-uri, informatia fiind neschimbata pana cand utilizatorul actualizeaza datele de intrare. Urmeaza doar sa selectam actiunea care se va efectua prin intermediul setXActionListener din clasa MyGUI.

La fiecare nou apel se intra intr-o subclasa in functie de operatia selectata. In subclasa se creeaza doua noi polinoame, deoarece textul introdus se poate schimba, apoi se calculeaza rezultatul prin intermediul metodelor existente si se trimite textul catre View.

```
21      class AdditionActionListener implements ActionListener{
22
23          @Override
24          public void actionPerformed(ActionEvent e) {
25              String firstPoly = myGUI.gettFieldFirstPoly();
26              String secondPoly = myGUI.gettFieldSecondPoly();
27              poly1 = new Polynomial(firstPoly);
28              poly2 = new Polynomial(secondPoly);
29              poly3 = poly1.add(poly2);
30              String result = poly3.toString();
31              myGUI.settFieldResult(result);
32          }
33      }
```

3. Clasa Polynomial

Clasa ce permite crearea de polinoame, sub forma de colectii de monoame. Are un singur constructor public, cel care primeste un String si un constructor privat, folosit in structura interna a clase, la creearea de polinoame intermediare necesare pentru efectuarea a diferite operatii.

La nivelul constructorului public se afla partea de parsare a String-urilor, care se foloseste de Regex pentru a genera un obiect de tipul Polynomial, ce contine un ArrayList<Monome> de monoame.

Adunarea a doua polinoame

Adunarea a doua polinoame se rezuma la adunarea coeficientilor monoamelor cu grad egal. Pentru a fi valabila pe caz general, folosim un algoritm de insumare, ce se reduce la trei cazuri:

1. Grad p1.monom < oricare grad p2.monom
   - Adaugam p2.monom la polinomul rezultat
2. grad p1.monom = oricare grad p2.monom
   - adaugam la rezultat p1.monom + p2.monom

- stergem p2.monom

3. daca exista grad p2.monom care n-au fost adaugate, adauga ce a ramas

```java
public Polynomial add(Polynomial pol) {
    Polynomial pol1 = new Polynomial();
    for (int i = 0; i < this.mList.size(); i++) {
        int p1 = this.mList.get(i).getExponent();
        float c1 = this.mList.get(i).getCoefficient();

        int i_pow = pol.getIndexPutere(p1);
        if (i_pow == -1) {
            // Nu am gasit deci adaugam un nou monom
            pol1.mList.add(new Monome(c1, p1));
        } else {
            // sumama deoarece am gasit ceva in polinomul 2
            int p3 = pol.mList.get(i_pow).getExponent();
            float c3 = pol.mList.get(i_pow).getCoefficient();
            pol1.mList.add(new Monome( coefficient: c1 + c3, p3));
            // Eliminam ce am adaugat deja
            pol.mList.remove(i_pow);
        }
    }
    //adaugam ce a ramas in pol 2
    for (int j = 0; j < pol.mList.size(); j++) {
        int p2 = pol.mList.get(j).getExponent();
        float c2 = pol.mList.get(j).getCoefficient();
        pol1.mList.add(new Monome(c2, p2));
    }
    pol1.sortM();
    return pol1;
}
```

Diferenta dintre doua polinoame

Diferenta dintre doua polinoame se poate trata ca un caz special de adunare.

1. Inmultim coeficientii polinomului al doilea cu -1
2. Adunam cele doua polinoame
3. Stergem monoamele care au coeficient 0

Inmultirea a doua polinoame

Inmultirea a doua polinoame am rezolvat-o printr-o metoda simpla.

1. Se inmulteste fiecare monom din primul polinom cu toate monoamele din cel de-al doilea si se adauga la un polinom rezultat
2. Ne folosim de functia de sortare si minificare pentru a ordona polinomul rezultat

```
116     public Polynomial product(Polynomial pol) {
117         Polynomial pol1 = new Polynomial();
118         for (Monome m1: this.mList) {
119             for (Monome m2 : pol.mList) {
120                 Monome m3 = m1.Product(m2);
121 
122                 pol1.mList.add(m3);
123             }
124         }
125         pol1.sortM();
126         return pol1;
127     }
```

Derivarea unui polinom

Derivarea se floseste de o metoda din clasa Monome, deoarece fiecare monom va fi derivat individual si adaugat la rezultat. In principiu, este parcursa lista de monoame si se apeleaza metoda prin care se deriveaza un monom.

```
141     public Polynomial derivative() {
142         Polynomial pol1 = new Polynomial();
143         for (Monome temp: this.mList) {
144             if(temp.exponent > 0)
145                 pol1.mList.add(temp.Derivative());
146         }
147         return pol1;
148     }
```

Integrarea unui polinom

Integrarea se face in aceeasi maniera cu derivarea. Este poarcursa lista de monoame si fiecare monom este integrat prin metoda din clasa Monome.

```
150     public Polynomial integration() {
151         Polynomial pol1 = new Polynomial();
152         for (Monome temp : this.mList) {
153             pol1.mList.add(temp.Integration());
154         }
155         return pol1;
156     }
157
```

Metoda sortM

In clasa Polynomial mai exista si o metoda de sortare si minificarea a unui polinom, folosita intern cu scopul de a reduce polinomul la o forma indicata. Dupa aplicarea acestei metode, lista de monoame va fi sortata in ordine descrescatoare a gradului, iar fiecare grad va aparea o singura data in cadrul unui polinom.

```
158     private void sortM() {
159         for (int i = 0; i < mList.size(); i++) {
160             int power = mList.get(i).getExponent();
161             for (int j = i + 1; j < mList.size(); j++) {
162                 int power_temp = mList.get(j).getExponent();
163                 if (power == power_temp) {
164                     mList.get(i).addCoef(mList.get(j).getCoefficient());
165                     this.mList.remove(j);
166                 }
167             }
168         }
169         Collections.sort(this.mList, Monome.getCompByPutere().reversed());
170     }
```

4. Clasa Monome

Clasa monome contine un constructor, o metoda folosita la compararea puterilor pentru sortarea din Polynomial, operatiile elemetare de inmultire a doua monoame, derivare si integrare a unui monom si o metoda de transformare a monomului intr-un string astfel incat sa fie afisat in formatul standard.

Inmultirea – metoda Product – se inmultesc coeficientii si se aduna exponentii

```
41  @   public Monome Product(Monome x) {
42          Monome p = new Monome();
43          p.coefficient = this.coefficient * x.coefficient;
44          p.exponent = this.exponent + x.exponent;
45          return p;
46      }
```

Derivarea – Derivative – coeficientul se inmulteste cu exponentul iar exponentul se scade cu o unitate

```
49      public Monome Derivative() {
50          Monome p = new Monome();
51          p.coefficient = this.coefficient * this.exponent;
52          p.exponent = this.exponent - 1;
53          return p;
54      }
```

Integrarea – Integration – coeficientul se imparte la (exponent + 1) iar la exponent se adauga 1.

```
56      public Monome Integration() {
57          Monome p = new Monome();
58          p.coefficient = this.coefficient / (this.exponent + 1);
59          p.exponent = this.exponent + 1;
60          return p;
61      }
```

5. Clasa JUnitTest

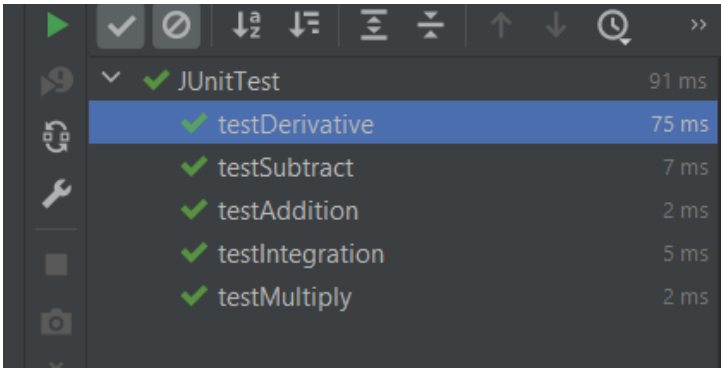
In aceasta clasa sunt executate testele pentru fiecare operatie. Codul pentru testarea operatiilor se repeta cu mici modificare. O sa luam ca exemplu testarea inmultirii:

```
60      @Test
61      public void testMultiply() {
62          poly1 = new Polynomial("2x^3+2x+1");
63          poly2 = new Polynomial("4x^3-3x^2");
64          result = poly1.product(poly2);
65          String res = result.toString();
66          assertNotNull(res);
67          assertEquals(res, actual: "8x^6-6x^5+8x^4-2x^3-3x^2");
68          testSuccess++;
69      }
```

Se creeaza doua noi polinoame, apoi se calculeaza rezultatul inmultirii dintre cele doua polinoame intr-un nou polinom rezultat, care este transformat in String. Apoi, se verica daca rezultatul oferit de aplicatie este egal cu rezultatul calculat manual dupa regulile matematice.

5. Rezultate

Programul ruleaza corespunzator, daca datele de intrare sunt corecte. Acest lucru este testat prin intermediul clasei JUnitTest care compara rezultatele obtinute in urma executiei operatiilor cu rezutatul adevarat.



Toate operatiile functioneaza corect, testele fiind executate cu succes.

6. Concluzii

In concluzie, tema Calculator de Polinoame pune accentul pe folosirea paradigmelor programarii orientate pe obiect in dezvoltarea aplicatiilor ceea ce m-a facut sa aprofundez cunostintele de programare orientata si sa invat rolul pe care il are separarea obiectivelor chiar de la inceputul problemei. Crearea unei imagini de ansablu asupra problemei duce la evitarea scrierii de cod inutil.