MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Facultatea de Automatică și Calculatoare

Specializarea Calculatoare

Tema 1:Calculator polinomial

-documentație-

Strujan Florentina

Gr. 302210

An 2, semestrul 2

Cuprins:

1. Obiectivul temei

Obiectivul acestei teme de laborator a fost să proiectăm și să implementăm un calculator polinomial, cu o interfață grafică ce poate fi ușor folosită de orice utilizator, cu doua campuri de introducere a polinoamelor de o singură variabilă cu coeficienți întregi, butoane pentru alegerea operației și un camp de afișare a rezultatului.

Obiectivele secundare sunt:

-folosirea programării orientate pe obiect (definirea de clase, metode, folosirea încapsulării etc.)

-folosirea Model View Controller

-folosirea limbajului de programare Java

-folosirea Java Swing pentru implementarea interfeței grafice

-folosirea Regex pentru verificarea validiății polinoamelor

-folosirea listelor in loc de vectori

-folosirea convențiilor de denumire Java

-folosirea Junit pentru testarea aplicației

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

Un polinom reprezintă o expresie formată din mai multe monoame, care sunt la rândul lor construite cu ajuor unor coeficienți și exponenți. Un calculator de polinoame poate fi realizat prin mai multe metode, diferența făcându-se în principal la modul de introducere a datelor. Eu am optat pentru introducerea a două șiruri în două textField-uri, cu asignarea valorii 0 pentu coeficientul termenilor inexistenți , pe care le-am prelucrat ulterior, însă exista și posibilitatea introducerii pe rând a coeficienților și exponenților sau a introducerii a două șiruri fără construirea termenilor inexistenți. Diferențe pot exista și la nivelul prelucrării datelor , așezarea și sortarea coeficienților și exponenților rezultați, aspectul final al interfeței grafice.

Metoda aleasă de mine este simplă, eficientă, însă are și neajunsuri.

Calculatorul este folosit pentru operații cu polinoame. Utilizatorul va introduce cele doua polinoame, dupa care va alege operatia dorita din cele disponibile: adunare, scădere, înmulțire. În cazul în care se optează pentru operația de derivare sau integrare, nu este necesară introducerea celui de-al doilea polinom deoarece nu se va ține cont de acesta. După apăsarea butonului respectiv operației, rezultatul va fi afișat în campul Rezultat, cu posibilitatea ștergerii acestuia prin click pe butonul clear, pentru realizarea altor operații.

3. Proiectare

3.1 Diagrama UML

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. 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.

3.2 Proiectare clase

Clasa CalcController este granița dintre interfața grafica și programul propriu-zis, unde se petrece interacțiunea utilizator-sistem de calcul.

Clasa Operatii este clasa în care au loc toate calculele. Operatii de adunare, scadere, inmultire, integrare si derivare.

Clasa CalcModel este clasa in care spunem programului ce sa execute. In aceasta clasa se realizeaza deschiderea interfetei grafice.

Clasa StringToPolinom este clasa unde convertim String ul introdus în polinom.

În clasele Monom și Polinom definim obiecteve respective, un polinom fiind format dintr-o listă de monoame.

Clasa CalcView nu are acțiuni utilizator, folosește la prezentare.

3.3 Metode

Metode utilizate in clasa Monom:

Structura unui monom: coefient și exponent; pentru fiecare dintre aceste doua valori există câte doua metode, o metodă de setare și de preluare a coeficientului și setare și preluare a exponentului.

Am mai folosit și metoda toString cu ajutorul căreia se realizează afișarea cât mai corectă a monoamelor, pe toate cazurile.

```
public double getCoeficient() {
	return coeficient;
}

public void setCoeficient(double coeficient) {
	this.coeficient = coeficient;
}

public int getExponent() {
	return exponent;
}

public void setExponent(int exponent) {
	this.exponent = exponent;
}
@Override
public String toString() {
	if (coeficient == 0)
		return "";
	if (coeficient > 0 && exponent == 0)
		return "+" + coeficient;
	if (coeficient < 0 && exponent == 0)
		return "" + coeficient;
	if (coeficient == 1 && exponent == 1)
		return "+x";
	if (coeficient == -1 && exponent == 1)
```

```
                return "-x";
        if (coeficient == 1)
                return "+x^" + exponent;
        if (coeficient == -1)
                return "-x^" + exponent;
        if (coeficient > 0 && exponent == 1)
                return "+" + coeficient + "x";
        if (coeficient > 0 && exponent == 1)
                return coeficient + "x";
        if (coeficient > 0 && exponent > 0)
                return "+" + coeficient + "x^" + exponent;
        else
                return coeficient + "x^" + exponent;
    }

}
```

Metode utilizate in clasa Polinom:

In aceasta clasa folosim o lista pentru a creea un polinom. Am creat metoda adaugăMonom pentru ușurarea calculelor și crearea polinoamelor, metoda getPolinom de tip ArrayList<Monom>, metoda toString pentru afișare și metoda size pentru determinarea numărului de monoame conținute în monom pentru operația de adunare.

```
public void adaugaMonom(Monom m) {
        polinom.add(m);
}

public List<Monom> getPolinom() {
        return polinom;
}

@Override
public String toString() {
        String p = "";
        for (Monom i : polinom)
                p = p + i;
        return p;
}

public int size() {
        return polinom.size();
        }
```

Metode utilizate in clasa Operatii

Adunare: pentru inceput, initializam un polinom și determinăm polinomul cu numărul maxim de monoame. Avem următoarele argumente pentru aceasta metoda :(Polinom p1,Polinom p2).

În funcție de size-ul maxim, parcurgem element cu element, adică fiecare monom din fiecare polinom pentru a căuta monoame cu aceeași exponenți. Odată gasite efectuăm adunarea creând un nou monom cu coeficientul reprezentat de suma celor 2 coeficienți , și același exponent. Monomul rezultat este adăugat in polinomul anterior inițializat. În momenul în care se depășește size-ul polinomului mai scurt, reiese că celelalte monoame ale polinomului lung nu au cu cine să mai fie adunate, așa că se adaugă pe rând la polinomul ce va fi returnat ca rezultat al sumei polinoamelor primate ca parametru.

```
public static Polinom adunare(Polinom p1, Polinom p2) {
		Polinom polFin = new Polinom();
		int max = maxim(p1.size(), p2.size());

		if (max == p1.size()) {
			for (Monom m1 : p1.getPolinom()) {
				for (Monom m2 : p2.getPolinom())
					if (m1.getExponent() == m2.getExponent())
						polFin.adaugaMonom(new
Monom(m1.getCoeficient() + m2.getCoeficient(), m1.getExponent()));
				if (m1.getExponent() >= p2.size())
					polFin.adaugaMonom(m1);
			}
			return polFin;
		} else {
			for (Monom m2 : p2.getPolinom()) {
				for (Monom m1 : p1.getPolinom())
					if (m1.getExponent() == m2.getExponent())
						polFin.adaugaMonom(new
Monom(m1.getCoeficient() + m2.getCoeficient(), m1.getExponent()));
```

```
                        if (m2.getExponent() >= p1.size())
                                polFin.adaugaMonom(m2);
                }
                return polFin;
        }
        }
```

Scadere: pentru inceput, initializam un polinom și determinăm polinomul cu numărul maxim de monoame. Avem următoarele argumente pentru aceasta metoda :(Polinom p1,Polinom p2).

În funcție de size-ul maxim, parcurgem element cu element, adică fiecare monom din fiecare polinom pentru a căuta monoame cu aceeași exponenți. Odată gasite efectuăm scăderea creând un nou monom cu coeficientul reprezentat de diferența celor 2 coeficienți , și același exponent. Monomul rezultat este adăugat in polinomul anterior inițializat. În momenul în care se depășește size-ul polinomului mai scurt, reiese că celelalte monoame ale polinomului lung nu au cu cine să se mai scadă, așa că, în funcție de al câtelea polinom e cel lung, se adaugă pe rând la polinomul ce va fi returnat ca rezultat al diferenței polinoamelor primate ca parametru ori vor primi coeficientul înmulțit cu -1 după care vor fi adăugate.

```
        public static Polinom scadere(Polinom p1, Polinom p2) {
                Polinom polFin = new Polinom();
                int max = maxim(p1.size(), p2.size());

                if (max == p1.size()) {
                        for (Monom m1 : p1.getPolinom()) {
                                for (Monom m2 : p2.getPolinom())
                                        if (m1.getExponent() == m2.getExponent())
                                                polFin.adaugaMonom(new
Monom(m1.getCoeficient() - m2.getCoeficient(), m1.getExponent()));
                                if (m1.getExponent() >= p2.size())
                                        polFin.adaugaMonom(m1);
```

```
                }
                return polFin;
            } else {
                for (Monom m2 : p2.getPolinom()) {
                    for (Monom m1 : p1.getPolinom())
                        if (m1.getExponent() == m2.getExponent())
                            polFin.adaugaMonom(new
Monom(m1.getCoeficient() - m2.getCoeficient(), m1.getExponent()));
                    if (m2.getExponent() >= p1.size()) {
                        m2.setCoeficient(-1 * m2.getCoeficient());
                        polFin.adaugaMonom(m2);
                    }
                }
                return polFin;
            }
        }
```

Inmultire: pentru început, inițializăm un polinom. Avem următoarele argumente pentru aceasta metoda :(Polinom p1,Polinom p2).

Pentru fiecare polinom parcurgem element cu element, adică fiecare monom din fiecare și le înmulțim, creând un nou monom cu coeficientul reprezentat de produsul celor 2 coeficienți , și exponentul reprezentat de suma celor 2 exponenți. Monomul rezultat este adăugat in polinomul anterior inițializat.

```
    public static Polinom inmultire(Polinom p1, Polinom p2) {
            Polinom polFin = new Polinom();
            Monom mon = new Monom();
            for (Monom m1 : p1.getPolinom())
                for (Monom m2 : p2.getPolinom())
                    polFin.adaugaMonom(
                                    new Monom(m1.getCoeficient() *
    m2.getCoeficient(), m1.getExponent() + m2.getExponent()));
            return polFin;
        }
```

Derivare: pentru început, inițializăm un polinom.Avem argument pentru aceasta metoda : (Polinom pol1) deoarece derivarea se realizează asupra unui singur polinom, în cazul nostru polinomul 1.

Se parcurge acest polinom element cu element și creăm un monom cu coeficientul egal cu produsul coeficientului monomului din polinom parcurs și exponentul acestuia și cu exponentul egal cu exponentului monomului adunat cu -1. Monomul rezultat este adăugat in polinomul anterior inițializat.

```
	public static Polinom derivare(Polinom p) {
		Polinom polFin = new Polinom();
		for (Monom m : p.getPolinom())
			polFin.adaugaMonom(new Monom(m.getCoeficient() * m.getExponent(), 
m.getExponent() - 1));
		return polFin;
		}
```

Integrare: pentru început, inițializăm un polinom.Avem argument pentru aceasta metoda : (Polinom pol1) deoarece derivarea se realizează asupra unui singur polinom, în cazul nostru polinomul 1.

Se parcurge acest polinom element cu element și creăm un monom cu coeficientul egal cu câtul rezultatului coeficientului monomului din polinomul parcurs împărțit cu exponentul acestuia +1 și cu exponentul egal cu exponentului monomului adunat cu 1. Monomul rezultat este adăugat in polinomul anterior inițializat.

```
		public static Polinom integrare(Polinom p) {
		Polinom polFin = new Polinom();
		for (Monom m : p.getPolinom())
			polFin.adaugaMonom(new Monom(m.getCoeficient() / (m.getExponent() 
+ 1), m.getExponent() + 1));
		return polFin;}
```

3.4 GUI

Interfața grafică sau Graphical User Interface este o interfață cu utilizatorul bazată pe un sistem de afișaj ce utilizează elemente grafice. Interfața grafică este numit sistemul de afișaj grafic-vizual pe un ecran, situat funcțional între utilizator și dispozitive electronice. Folosim o interfata grafica User-Friendly cu scopul de a putea fi folosit acest calculator de polinoame si de persoane non-specializate.

Interfata grafica cuprinde urmatoarele elemente :

Frame = “rama” in care se adauga toate elementele de care avem nevoie pentru buna functionare a programului. E practice o fereastra care, dupa utlizare, apasand butonul X din dreapta sus se poate inchide (Exit_on_close).

Butoane = sunt in numar de 6 , fiecare cu alta funcționalitate la apăsare:

Aduna face adunarea dintre cele 2 polinoame după introducerea acestora

Scade face scăderea dintre cele 2 polinoame după introducerea acestora

Inmulteste face înmulțirea dintre cele 2 polinoame după introducerea acestora

Derivеaza derivează primul polinom

Integreaza integrează primul polinom

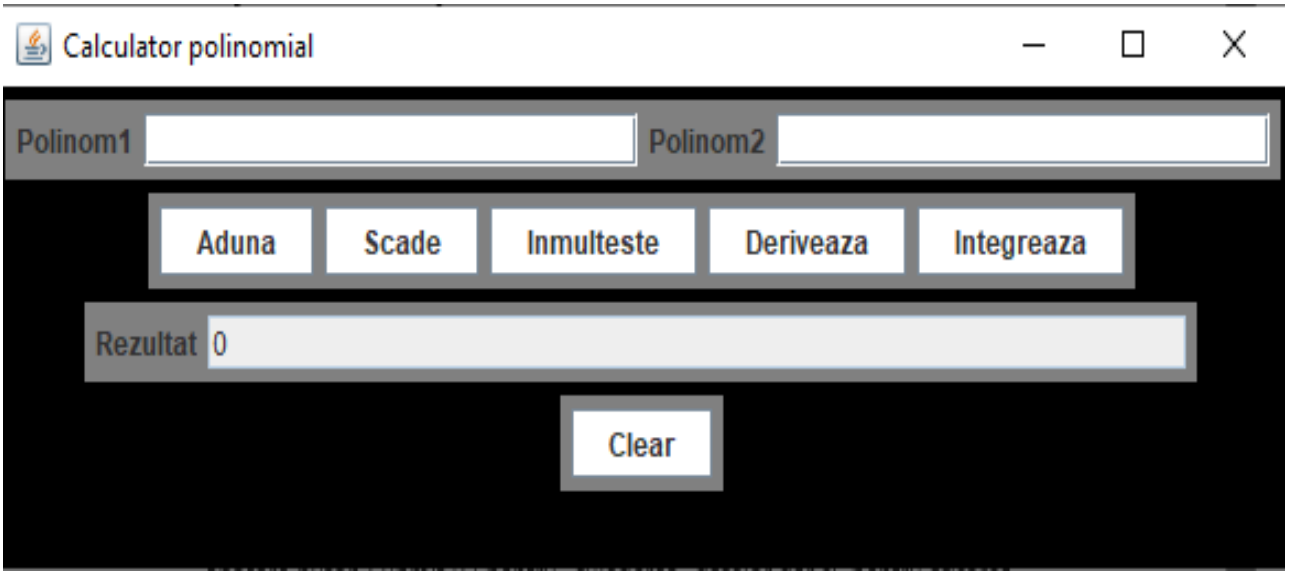
Clear face ca toate textField-ul rezultat sa devina 0.

TextField = spații dreptunghiulare în care se pot introduce date de la tastatura. Dar pe langa asta pot fi folosite si pentru a afișa rezultatul fără a se putea introduce date de la tastatură.

Label = este efectiv o etichetă, care poate fi titlul, informații, indicații etc. care pot ajuta utilizatorul să folosească progamul. În cazul nostrum e folosit doar pentru titlul sugestiv “Calculator polinomial”.

Pentru fiecare button, s-a făcut o metodă nouă care implică ActionEvent. Astfel, de fiecare dată când are loc o acțiune la un buton, de exemplu a fost apăsat, ActionEvent-ul denumit e, transmite informația la ActionListener care așteaptă astfel de informații.

4.Implementare



Etapele necesare efectuarii operatiei de **adunare** sunt:

-introducerea String ului corespunzător primului polinom în TextField ul cu eticheta Polinom1

-introducerea String ului corespunzător celui de-al doilea polinom în TextField ul cu eticheta Polinom2

-apăsarea butonului Adauga pentru efectuarea operației de adunare dintre cele 2 polinoame. Rezultatul va fi afișat in textField-ul rezultat.

Etapele necesare efectuării operației de **scădere** sunt:

-introducerea String ului corespunzător primului polinom în TextField ul cu eticheta Polinom1

-introducerea String ului corespunzător celui de-al doilea polinom în TextField ul cu eticheta Polinom2

-apăsarea butonului Scade pentru efectuarea operației de scădere dintre cele 2 polinoame. Rezultatul va fi afișat in textField-ul rezultat.

Etapele necesare efectuării operației de **înmulțire** sunt:

-introducerea String ului corespunzător primului polinom în TextField ul cu eticheta Polinom1

-introducerea String ului corespunzător celui de-al doilea polinom în TextField ul cu eticheta Polinom2

-apăsarea butonului Inmulteste pentru efectuarea operației de înmulțire dintre cele 2 polinoame. Rezultatul va fi afișat in textField-ul rezultat.

Etapele necesare efectuarii operatiei de **derivare** sunt:

-introducerea String ului corespunzător polinomului în TextField ul cu eticheta Polinom1

-apăsarea butonului Deriveaza pentru efectuarea operației de derivare a polinomului. Rezultatul va fi afișat in textField-ul rezultat.

Etapele necesare efectuarii operatiei de **integrare** sunt:

-introducerea String ului corespunzător polinomului în TextField ul cu eticheta Polinom1

-apăsarea butonului Integreaza pentru efectuarea operației de integrare a polinomului. Rezultatul va fi afișat in textField-ul rezultat.

5.Rezultate

```
O singura data inaintea executiei setului de teste din clasa!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
Constructor inaintea fiecarui test!
Incepe un nou test!
S-a terminat testul curent!
O singura data dupa terminarea executiei setului de teste din clasa!
S-au executat 8 teste din care 8 au avut succes!
```

Folosind Junit am testat fiecare operație implemenată, testând cu assertEquals egalitatea dintre rezultatul aștetat și cel rezultat.

6.Concluzii

Sunt de părere că în urma realizării acestei teme am reușit să-mi reamintesc și să aprofundez materia de semestrul trecut, precum și să-mi îmbunătățesc tehnicile de programare în acest limbaj.

În continuare va trebui să lucrez la introducerea polinoamelor fără a introduce coeficientul 0 pentru monoamele inexistente și fără putere în cazul puterilor 0 și 1.

7.Bibliografie

http://users.utcluj.ro/~igiosan/teaching_poo.html

https://www.geeksforgeeks.org/program-add-two-polynomials/

https://stackoverflow.com/questions/28859919/java-regex-separate-degree-coeff-of-polynomial