MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

CALCULATOR POLINOAME

Documentatie Tema 1

Elev: Ţoc Roxana-Ştefania
Grupa: 30224
Profesor indrumător: Antal Marcel

Cuprins:

1.Obiectivul temei

1.1 Obiectivul principal

Obiectivul acestei teme este de a proiecta si implementa un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi. Operatiile continute de acest proiect sunt:

- Adunarea a doua polinoame: P(x)+Q(x)
- Scaderea a doua polinoame: P(x)-Q(x)
- Inmultirea a doua polinoame: P(x)*Q(x)
- Impartirea a doua polinoame: P(x)/Q(x)
- Derivarea unui polinom: P(x)'
- Integrarea unui polinom: ∫ P(x).

Polinoamele sunt construite din monoame, care sunt alcatuite dintr-un coeficient de tip double inmultit cu una din variabile care au un exponent intreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile in acel monom.
Forma generala a unui polinom este:
P(x)=an*x^n+an-1*x^n-1+…+a1*x+a0
Toate operatiile genereaza un alt polinom.

1.2 Obiectivul secundar

-Dezvoltarea de use case-uri si scenarii: Realizarea de operatii pe polinoame. Se introduc polinoamele si se selecteaza operatia dorita, programul generand rezultatul.
-Alegerea structurilor de date: Folosirea ArrayList in loc de vector pentru usurinta de utilizare
-Impartirea pe clase: Am ales clasele Monom si Polinom pentru constructia polinoamelor , clasa Operatii pentru implementarea operatiilor cerute, clasa View+Controller unde avem interfata cu utilizatorul si clasa testCalculator pentru JUnit test.
-Dezvoltarea algoritmilor: Algoritmii dezvoltati sunt cei de operare cu polinoame
-Implementarea solutiei: Prezentarea claselor, metodelor, descrierea interfetei utilizator si a ideilor aplicate in solutionarea problemei
-Testare: Crearea unui JUnit pentru testarea operatiilor pe polinoame

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

2.1 Analiza problemei

Abstractizarea transpune structura unei probleme din realitate in entitate. Programul trebuie sa realizeze operatiile de adunare, scadere, inmultire, impartire, derivare si integrare dupa ce am introdus doua, respectiv un polinom. Abstractizarea este nivelul cel mai inalt unde privim obiectul program ca o comunitate de obiecte care interactioneaza. Polinomul are ca si caracteristici o lista de monoame, iar monomul are ca si caracteristici puterea si coeficientul.

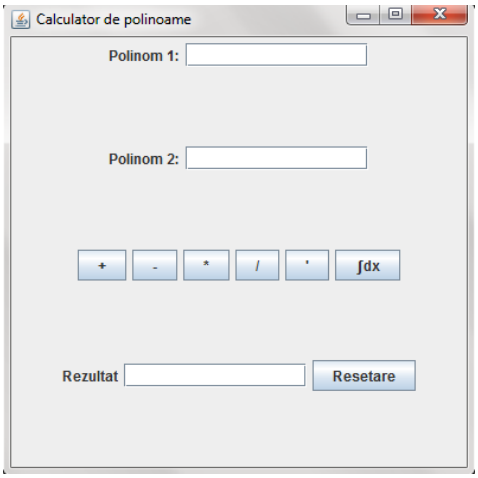
2.2 Asumptii

Datele se introduc sub forma "x^5+5x^3+x^1+x^0+1" sau "x^5+5x^3+x+1", cele doua fiind echivalente. Introducerea sub alta forma duce la eroare.

2.3 Cazuri de utilizare

-Adunarea: Se introduc polinoamele, apoi se selecteaza operatia de "+", apoi rezultatul va fi afisat in JTextField-ul corespunzator JLabel-ului "Rezultat".
-Scaderea: Este similara cu utilizarea operatiei de adunare, cu exceptia apasarii butonului pentru scadere "-".
-Inmultirea: Analog cu operatiile precedente, apasand butonul pentru inmultire "*".
-Impartirea: Analog cu operatiile de mai sus, cu exceptia apasarii butonului "/", pentru impartirea celor doua polinoame.

-Derivarea: Aceasta operatie necesita doar un singur polinom, campul al doilea va deveni inactiv, iar la apasarea butonului “ ‘ ”, rezultatul va fi afisat.
-Integrarea: Este asemenea derivarii, doar ca afisarea coeficientilor se va face sub forma reala.



2.4 Erori

Programul trateaza aparitiile erorilor care ar putea aparea. Se trateaza erorile de introducere a polinoamelor si de a realiza operatii fara a introduce polinoame.

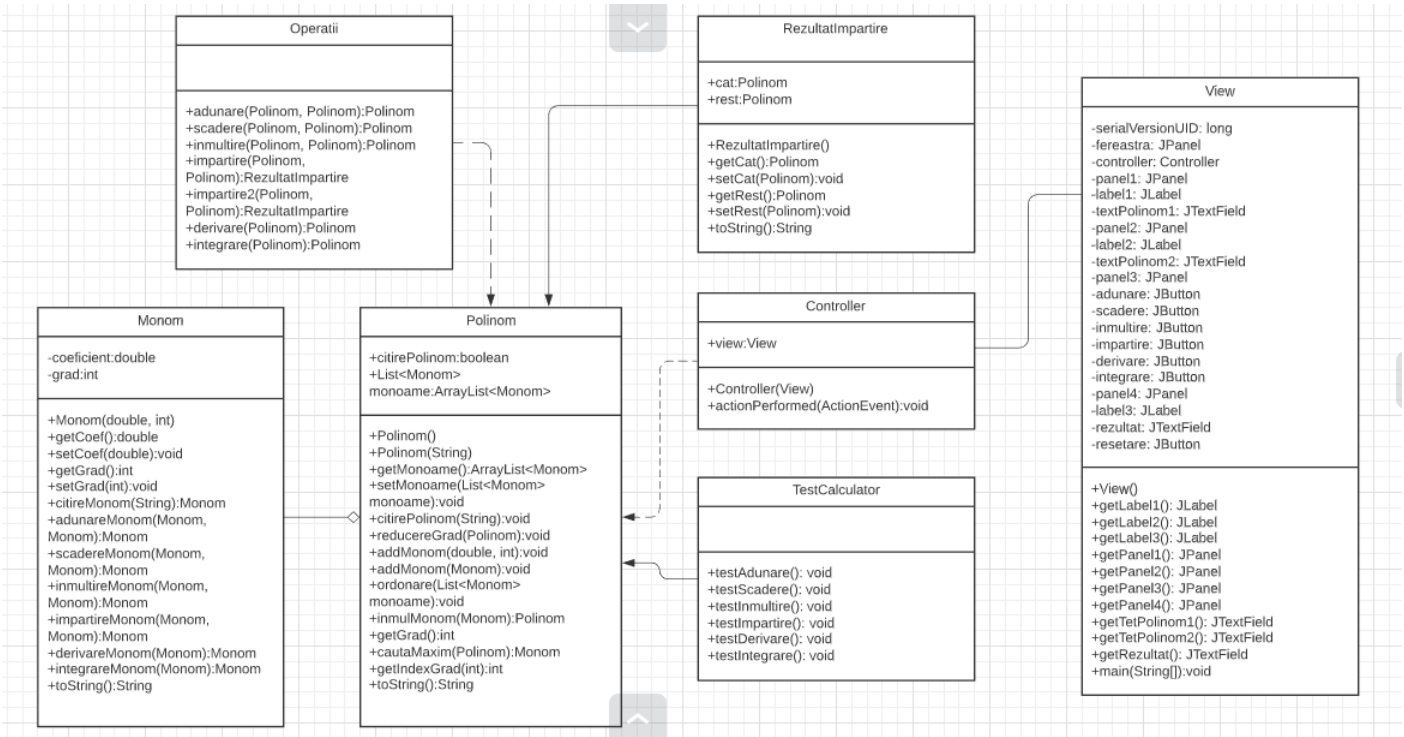
3.Proiectare

3.1 Structuri de date

Ca structura de date, am utilizat un ArrayList de monoame, reprezentand polinomul. Am lucrat cu aceasta structura, folosind functii definite in biblioteca java.util.ArrayList, cum ar fi add, remove, fiind mult mai usor de operat decat cu un vector.

3.2 Diagrama de clase

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele si specificatii pentru software. Este folosita pentru reprezentarea vizuala a claselor si a interdependentelor, taxonomiei si a relatiilor de multiplicitate dintre ele. Diagramele de clasa sunt folosite si pentru reprezentarea concreta a unor instante de clasa, asadar obiecte, si a legaturilor dintre acestea.



3.3 Algoritmi

Algoritmi pentru adunare/scadere:

Algoritmii utilizati in realizarea acestui calculator de polinoame sunt cei ce opereaza cu polinoame, cum ar fi adunarea si scaderea polinoamelor, in care se aduna/scad coeficientii monoamelor cu aceeasi putere si restul se adauga la rezultat.

```
public Polinom adunare(Polinom p, Polinom q) {
		Polinom r = new Polinom();
		int i = 0, j = 0;
		int sizeP = p.getMonoame().size();
		int sizeQ = q.getMonoame().size();
		Monom m;
		while (i < sizeP && j < sizeQ) {
			if (p.getMonoame().get(i).getGrad() == q.getMonoame().get(j).getGrad()) {
				m = new Monom(p.getMonoame().get(i).getCoef() +
q.getMonoame().get(j).getCoef(),
							p.getMonoame().get(i).getGrad());
				i++;
				j++;
			} else if (p.getMonoame().get(i).getGrad() > q.getMonoame().get(j).getGrad()) {
				m = new Monom(p.getMonoame().get(i).getCoef(),
p.getMonoame().get(i).getGrad());
				i++;
			} else {
				m = new Monom(q.getMonoame().get(j).getCoef(),
q.getMonoame().get(j).getGrad());
				j++;
			}
			r.addMonom(m);
		}
		while (i < sizeP) {
			m = new Monom(p.getMonoame().get(i).getCoef(), p.getMonoame().get(i).getGrad());
			i++;
			r.addMonom(m);
		}
		while (j < sizeQ) {
			m = new Monom(q.getMonoame().get(j).getCoef(), q.getMonoame().get(j).getGrad());
			j++;
			r.addMonom(m);
		}
		;
		return r;
	}
```

Algoritm pentru inmultire:

Algoritmul pentru inmultirea a doua polinoame este acela de a inmulti fiecare monom cu fiecare monom din celalalt polinom inmultiind coeficientii si adunand exponentul.

```
public Polinom inmultire(Polinom p, Polinom q) {
		List<Monom> m = new ArrayList<Monom>();
		for (Monom i : p.getMonoame()) {
			for (Monom j : q.getMonoame()) {
				m.add(i.inmultireMonom(i,j));
			}
		}
		Polinom r = new Polinom(m);
		r.reducereGrad(r);
		return r;
	}
```

Algoritmi pentru integrare/derivare:

Alti algoritmi sunt cei pentru integrare si derivare a polinoamelor, facand modificari corespunzatoare asupra coeficientilor si puterilor monoamelor.

```
public Polinom derivare(Polinom p) {
		Polinom r = new Polinom();
		for (Monom m : p.getMonoame()) {
			r.getMonoame().add(m.derivareMonom(m));
		}
		return r;
	}

public Polinom integrare(Polinom p) {
		Polinom r = new Polinom();
		for (Monom m : p.getMonoame()) {
			r.getMonoame().add(m.integrareMonom(m));
		}
		return r;
	}
```

4.Implementare

Clasa **Monom** contine metodele getCoef(), getGrad(), setCoef(), setGrad() pentru settere si gettere, constructorii Monom(), Monom(double) si Monom(double,int). Mai contin si metodele care fac operatii pe monoame cum ar fi citireMonom(string), adunareMonom(monom, monom), scadereMonom(monom, monom), inmultireMonom(monom, monom), impartireMonom(monom, monom), derivareMonom(monom), integrareMonom(monom) si metoda toString() care returneaza monomul sub forma de string.

Clasa **Polinom** contine constructorii Polinom(), Polinom(List<Monom>) si Polinom(string). Citirea unui string se face prin metoda citirePolinom(string) prin care se citeste un string si cu ajutorul split-ului se delimiteaza monoamele. Avem metodele de adaugare a unui monom, addMonom(double,int) sau addMonom(string). Ordonarea monoamelor in polinoame se face prin metoda ordonareMonoame(List<Monom>) in functie de gradul monoamelor, iar daca avem mai multe monoame de acelasi grad acestea se vor reduce prin metoda reducerePolinom(). Avem ca si settere si gettere getMonoame() si setMonoame(List<Monom>).

Clasa **Operatii** contine urmatoarele metode: adunare(polinom, polinom), aceasta realizand adunarea monoamelor cu acelas grad si pe cele care nu au corespondent le copiaza, la final se apeleaza metoda de reducere si ordonare. Polinomul rezultat nu va avea polinoamele inserate in ordine descrescatoare dupa grad, iar din acest motiv este necesara o sortare dupa grad prin utilizarea metodei ordonare din clasa polinom. A doua metoda, cea de scadere(polinom, polinom) se face la fel cu adunarea cu diferenta ca monoamele din al doilea polinom vor fi negate. Metoda de inmultire(polinom, polinom) consta in parcurgerea polinoamelor si inmultirea fiecarui monom cu monoamele celuilalt. Pentru metoda de impartire(polinom, polinom) am avut nevoie de crearea clasei RezultatImpartire in care vom avea stocat catul si restul impartirii. Primul polinom va fi restul, iar un polinom auxiliar in care vom introduce succesiv monoamele necesare realizarii impartirii. Algoritmul va merge atata timp cat gradul primului polinom (impartitul) e mai mare sau egal cu al celui de-al doilea (impartitor). Se inmultest primul polinom al impartitorului cu un monom prin intermediul caruia sa se reduca primul monom al impartitului, iar acesta va fi inserat in cat. Primul polinom devine astfel diferenta lui cu produsul monomului cu impartitorul.
Metoda derivare(polinom) efectueaza operatia de derivare asupra unui polinom, implementarea acestei metode se face relativ simplu, parcurgandu-se lista o singura data se adauga intr-un polinom nou, un monom dupa efectuarea operatiei de derivare din clasa Monom. Gradul va scadea cu 1, iar coeficientul va fi inmultit cu gradul initial, tinandu-se cont ca vor disparea constantele.
Metoda integrare(polinom) se efectueaza asemenea operatiei de derivare, doar ca gradul va greste, iar coeficientul va fi impartit cu noul grad. Trebuie tinut cont la afisare de adaugarea constantei C pentru o corectitudine matematica mai mare.

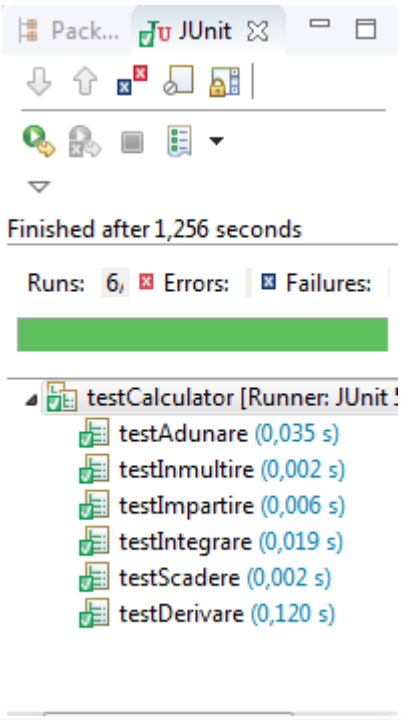
Clasa **View** este standard, aceasta continand de asemenea si main-ul programului. Prin ea se realizeaza interfata grafica cu utilizatorul, avand ca variabile: label-urile(3) cu scop de a ajuta utilizatorul sa utilizeze aplicatia, TextFieldurile pentru introducerea si afisarea polinoamelor si cele 7 butoane interactive. Pentru realizarea meniului, am utlizat 4 paneluri secundare care vor fi introduse orizontal in panelul principal “fereastra”. Primul va avea TextField-ul pentru primul polinom, respective label-ul orientativ pentru utilizator. Cel de-al doilea va fi analog primului, dar va fi utilizat pentru cel de-al doilea polinom. Al treilea panel va cuprinde 6 butoane interactive care vor fi reprezentate prin semne standard pentru operatiile pe care le executa, adica +, -, *, /,’(adunare, scadere, inmultire, impartire, derivare), iar in ceea ce priveste integrarea va fi utilizat un cod unic ce se foloseste pentru

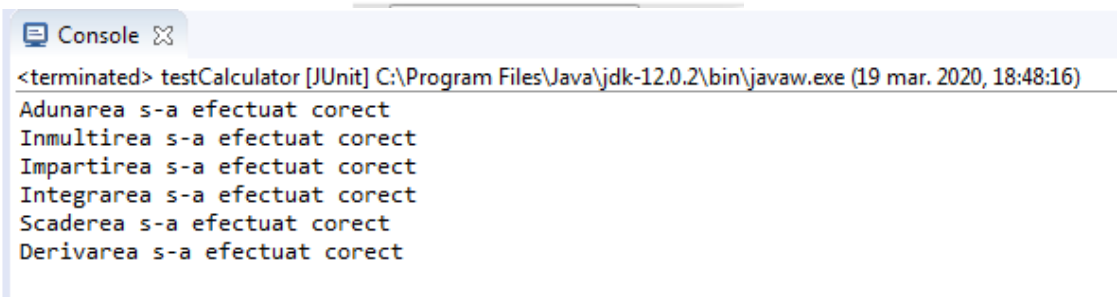
semnul de integrala, adica “u222B” plus notatia dx, pentru a evidentia asupra cui se face integrarea. In ultimul panel se va insera rezultatul reprezentat printr-un TextField ce are setata editarea pe false pentru a se evita modificarea de catre un utilizator, in stanga acestuia va fi adaugat label-ul “Rezultat”, iar la final va cuprinde un buton interactive “Resetare” care reinitializeaza TextField-ul rezultatului. In main am adaugat numele frameului “Calculator de polinoame”. Totodata, fiecare buton are adaugat un actionListener ce este implementat in clasa Controller.

Clasa **Controller** implementeaza ActionListener, deoarece in ea avem functionalitatile tuturor butoanelor din meniu. Astfel, am realizat Override la actionPerformed si initial avem de stabilit ce buton a fost selectat de catre utilizator. Pentru fiecare din acestea va fi folosita o metoda prezenta in interiorul clasei Operatii. Polinomul este transformat in monoame prin inserarea pe rand a coeficientilor si a gradelor, iar mai apoi acestea sunt introduse intr-un polinom si este returnat polinomul rezultat. In ceea ce priveste actionPerformed, trebuie stabilit care din cele 7 butoane a fost apasat. Primele 4 butoane au nevoie de 2 polinoame pentru realizarea operatiei. Citim continutul celor 2 TextFielduri editabile, iar apoi rezultatul returnat ca si polinom sau ca si RezultatImpartire in cazul impartirii, il afisam in TextFieldul needitabil fixat pentru rezultat. Analog procedam si cu butoanele de derivare si de integrare, cu exceptia ca vom citi un singur polinom de pe primul TextField.

5.Testare

Un pachet utlizat in acest proiect este pachetul JUnit. Acesta este, propriu-zis de tipul JUnit. Acest pachet este utilizat pentru a verifica daca operatiile de adunare, scadere, inmultire, impartire, derivare si integrare se efectueaza corect. Avem metodele testAdunare(), testScadere(), testInmultire(), testImpartire(), testDerivare(), testIntegrare() care sunt agregate in suite. Clasa suite este un tip specializat de test runner. Acesta permite construirea unui grup de teste din diferite clase.





6.Rezultate

Rezultatele se afiseaza in ultimul JTextField din interfata. Pe baza informatiilor primite in campurile de mai sus, se creeaza polinoamele, iar la apasarea unui buton se face operatia corespunzatoare.
Rezultatul se va afisa sub forma: an*xn+an-1*xn-1+…+a1*x+a0, an…a0 sunt coeficientii, iar n, n-1, …1,0 sunt gradele.

7.Concluzii, dezvoltari ulterioare

7.1 Concluzii

Concluzia principala pe care am dedus-o in urma realizarii acestei teme este faptul ca pentru o mai buna gestionare a memoriei calculatorului este indicat sa se foloseasca structuri de date de memorare in liste in loc de vectori. Astfel consumul memoriei este mult mai mic, desi timpul de acces este mai ridicat.
De asemenea, am realizat faptul ca testarea metodelor are un rol deosebit de important in procesul de dezvoltare al unui program, deoarece prin utilizarea metodelor de testare ne putem da seama daca codul pe care l-am scris la un anumit punct este unul functional, detectarea si corectarea erorilor fiind mult mai eficiente. In concluzie, implementarea metodelor de testare nu trebuie neglijata.

7.2 Dezvoltari ulterioare

Exista o serie de imbunatatiri ce ar putea fi adaugate acestui program. O prima imbunatatire ar fi aflarea radacinilor polinoamelor, generarea graficicului unui polinom in functie de radacini si cresterea numarului de polinoame, eventual a polinoamelor cu mai multe variabile. Cel mai important lucru a unui program este functionalitatea, utilizatorul in zilele noastre nu este interesat de modul in care aplicatia functioneaza si de modul in care aceasta este implementata. El vede doar interfata si asadar aceasta trebuie sa fie foarte usor de accesat. Este evident ca nimeni nu o sa o foloseasca un program care nu functioneaza, dar foarte putini o sa foloseasca un program care este prea greu de utilizat. Asadar este extrem de important eficienta programului, dar si usurinta utilizarii aplicatiei.

8.Bibliografie:

-https://www.geeksforgeeks.org/split-string-java-examples/
-http://stackoverflow.com/
-http://wikipedia.com/polynom
-http://java2novice.com/java-collections-and-util