

Facultatea de Automatica si Calculatoare

Tema 1:

Sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi

Disciplina: Tehnici de programare

Student: Profesor coordonator:

Coman Vasile Antal Marcel

An II Profesor curs:

Grupa 30221 Ioan Salomie

UNIVERSITATEA TEHNICĂ _ DIN CLUJ-NAPOCA

Cuprins

1. Obiectivul temei	3
2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori	
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)	4
4. Implementare	
5. Testare7	7
6. Rezultate	9
7. Concluzii1	1
8. Bibliografie11	1



1. Obiectivul temei

Obiectivul principal al temei este proiectare si implementarea unui sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

Obiective secundare:

-construirea polinomului folosindu-se o lista de monoame;

- -implementarea operatiilor de adunare, scadere, inmultire si impartire pentru doua monoame, astfel se simplifica implementarea operatiilor pentru polinoame;
- -transformarea polinomului dat sub forma unui string de catre utilizator in monoame adica convertirea unui string in instanta unei clase, mai exact o lista de monoame;
- -transformarea inversa prin care lista de monoame este afisata sub forma unui string utilizatorului ca si rezultat in urma operatiilor pe monoame si polinoame.

Toate aceste obiective incapsulate in acelasi proiect duc la rezolvarea cerintei. Obiectivele vor fi detaliate in capitolele care descriu proiectarea si implementarea.

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

Forma generala a unui polinom:

 $P(X) = a0X^n+a1X^n(n-1)+...+(an-1)X + an$, unde a0!=0.

Proprietati:

- -suma/produsul/diferenta/ a doua polinoame este tot un polinom;
- -derivata unui polinom este tot un polinom.

Cerinte si functionalitati:

- -introducerea unui polinom;
- -afisarea unui polinom;
- -adunarea a doua polinoame;
- -diferenta a doua polinoame;
- -inmultirea a doua polinoame;
- -impartirea a doua polinoame;
- -integrarea unui polinom;
- -derivarea unui polinom.

Use-case

Avem un sistem in care un utilizator poate sa introduca doua polinoame si sa efectueze pe aceste doua polinoame operatiile amintite mai sus primind un rezultat specific operatiei selectate.

Asumptii facute:

- -se presupune ca utilizatorul introduce un polinom corect sub forma mentionata mai sus si nu introduce un polinom null care ar genera erori in cazul unor operatii;
- -tipul de data pentru coefcient este double pentru a facilita operatia de integrare.



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

Am ales sa impart proiectul in trei clase:

- -o clasa pentru a defini un monom prin coeficient si exponent(ax^b);
- -o clasa pentru a defini un polinom printr-o lista de monoame;
- -o clasa pentru implementarea interfetei grafice.

Ulterior am adaugat si o clasa de test in care am verificat unele operatii implementate pe polinoame.

Monom	Polinom	Gui
-coef: double	-poli:	-pane: Jpanel
-exp: int	ArrayList <monom></monom>	-11, 12 ,13 ,14: Jlabel
		-tf1 ,tf2 ,tf3 ,tf4: JtextField
		-btn1, btn2, btn3, btn4, btn5
		,btn6, btm7, btn8: Jbutton
		- c1, c2: JcheckBox
		- p,q: Polinom
+sumaMonom(Monom):	+sumaPolinom(Polinom):	+actionPerformed(ActionEv
Monom	Polinom	ent): void
+diferentaMonom(Mono	+diferentaPolinom(Polin	
m): Monom	om): Polinom	
+produsMonom(Monom	+produsPolinom(Polinom	
): Monom): Polinom	
+raportMonom(Monom)	+raportPolinom(Polinom)	
: Monom	: Polinom	
+getCoef(): double	<pre>-restrangePolinom(): void</pre>	
+getExp(): int	-grad(): int	
+compareTo(Monom):	+derivarePolinom():	
int	Polinom	
+toString(): String	+integrarePolinom():	
	Polinom	
	+parsarePolinom(String):	
	void	
	+afisarePolinom(): String	
	-getPoli():	
	ArrayList <monom></monom>	
	-	
	setPoli(ArrayList <mono< td=""><td></td></mono<>	
	m>): void	



4. Implementare

Clasa Monom defineste un monom de forma ax^b unde a este coeficientul si b este exponentul. Am implementat urmatoarele metode:

- -metode pentru suma, diferenta, raportul si produsul a doua monoame care returneaza un obiect de tip monom utilizand regulile din matematica(suma/diferenta a doua monoame de acelasi grad este egala cu un monom de acel grad avand coeficientul egal cu suma/diferenta coeficientilor, produsul a doua monoame returneaza un monom avand gradul egal cu suma gradelor celor doua monoame si coeficientul egal cu produsul coeficientilor iar raportul a doua monoame ne returneaza un monom cu gradul egal cu diferenta gradelor si coeficientul egal cu raportul coeficientilor); -am utilizat gettere pentru exponent si coeficient, variabilele instanta avand modul de vizibilitate private;
- -am suprascris metoda compareTo pentru a putea sorta lista de monoame descrescator, aceasta metoda returneaza -1 daca gradul primului monom este mai mare decat celui de-al doilea, 1 pentru conditia inversa si 0 in caz de egalitate;
- -am suprascris metoda toString pentru a putea lega intre ele monoamele din lista de monoame astfel inca sa aiba aspectul unui polinom, aceasta metoda returneaza un string de forma coef+"x^"+exp+"+";
- -avem doi constructori cel implicit care initializeaza variabile instanta la valorile specifice fiecarui tip de data si un constructor care primeste coeficientul de tip double si puterea de tip int.

Clasa Polinom este formata dintr-o variabila instanta de tip ArrayList<Monom> adica este compusa dintr-o lista de monoame.

Metode implementate:

- -pentru suma a doua obiecte de tip polinom, care sunt formate dintr-o lista de monoame, am parcurs cu foreach cele doua liste de monoame iar unde gasim egalitate intre grade efectuam suma intre cele doua monoame si adaugam rezultatul intr-o noua lista de monoame, apoi adaugam in acea noua lista monoamele din prima lista care au gradul diferit de toate monoamele din a doua lista si invers. Returnam un nou polinom folosind un constructor care primeste lista de monoame nou construita;
- -pentru diferenta ideea este acceasi cu exceptia ca monoamele din a doua lista care au gradul diferit de toate monoamele din prima lista le vom adauga cu semn schimbat in lista construita:
- pentru produsul a doua polinoame se parcurg cele doua liste de monoame si se inmulteste fiecare element din prima lista cu fiecare element din a doua lista utilizandu-se operatia de inmultire a doua monoame implementata in clasa Monom, monoamele rezultate se adauga intr-o lista care va fi transmisa ca si parametru pentru constructorul din clasa Polinom astfel se va construi un nou polinom fiind rezultatul inmultirii celor doua polinoame;



-pentru impartirea a doua monoame se utilizeaza algoritmul prezentat in cursul orelor de laborator, am adaugat cazul in care gradul deimpartitului este mai mic decat al impartitorului funcia sa returneze ca si cat 0 iar restul va fi impartitorul; -metoda de parsare a unui polinom primeste polinomul dat sub forma unui string de catre utilizator si cu ajutorul functie replaceAll se editeaza textul astfel incat sa poata fi despartit in stringuri asemanatoare unui monom. Despartirea(split) se face dupa caracterul "+" iar apoi cu foreach se parcurg stringurile formate in urma operatie split. Astfel ca din stringurile formate putem construi o lista de monoame facand cast la double la ce este in fata lui "x" si facand cast la int la ce este dupa "^". Astfel transmitem constructorului din clasa Monom un parametru de tip double care va reprezenta coeficientul si un parametru de tip int care va reprezenta exponentul. Constructorul va crea un obiect de tip monom care va si adaugat in lista de monoame. Astfel parcurgandu-se fiecare string se va crea lista de monoame corespunzatoare polinomului dat de catre utilizator;

-metoda de restrangere a unui polinom este conceputa pentru ca in cazul in care utilizatorul introduce un polinom care are mai multe monoame de acelasi grad sau daca in urma unei operatii apar mai multe monoame de acelasi grad acestea sa fie adunate sub forma unui singur monom. Metoda sorteaza lista de monoame, folosind functia sort si metoda suprascrisa compareTo, si cat timp doua monoame vecine au acelasi grad le aduna intr-un monom apoi acest monom este adaugat intr-o noua lista de monoame iar la final polinomul este setat cu noua lista de monoame adica cu lista restransa;

-metoda de afisare polinom poate fi considerata echivalentul invers al metodei de parsare, astfel ca parcurgem lista de monoame al polinomului si fiecarui monom apelam metoda toString suprascrisa in clasa Monom si adaugam intr-un string prin concatenare fiecare monom transformat. La final folosim functia replaceAll pentru a edita stringul creat cu scopul de a genera o varianta cat mai fidela de afisare a polinomului.

Clasa Gui extinde clasa JFrame si implementeaza interfata ActionListener.

In clasa Gui am construit interfata cu utilizatorul care permite o comunicare usoare intre utilizator si program. Astfel ca utilizatorul va avea posibilitatea de a introduce doua polinoame intr-un textfield. Acesta trebuie sa selecteze butonul Citeste pentru ca polinomul dat sa fie trimis spre parsare. Dupa ce a introdus cele doua polinoame utilizatorul poate sa selecteze operatiile de adunare, scadere, inmultire si impartire. La impartire catul se va afisa intr-un textfield iar restul in alt textfield. Pentru integrare si derivare acesta dispune de un checkbox care permite alegerea polinomului care sa fie trimis spre operatia de integrare, respectiv derivare.

Constructorul din clasa Gui primeste ca parametru un string care va reprezenta numele frame-ului(ferestrei). In constructor se face apel catre constructorul din clasa predefinita JFrame transmitandu-se numele frame-ului folosind super(); De asemena in constructor se pozitioneaza fiecare label, buton, textfield si checkbox si se adauga ascultatori unde este nevoie pentru a putea modela in metoda actionPerformed comportarea interfetei grafice.



Suprascrierea metodei actionPerformed ne permite sa controlam comportarea interfetei grafice cand utilizatorul interactioneaza cu aceasta.

Utilizatorul poate introduce polinoame care au mai multi termeni cu acelasi grad fara a afecta rezultatele operatiilor deoarece in momentul citirii polinomului programul va sorta descrescator in functie de grad si va restrange lista de monoame, la fel facand si inainte de a trimite spre afisare rezultatul;

Polinoame care au mai multe spatii intre termeni sau in loc de x au X nu afecteaza rezultatele. De asemenea au fost tratate cazurile cand utilizatorul introduce x sau -x sa se considere $+1x^1$ respectiv $-1x^1$.

Singura conditie care trebuie indeplinita este ca polinomul sa fie in functie de x sau X si fiecare termen sa aiba un numar inaite de x si un numar dupa ^ cu exceptia cazurilor cand avem x,-x sau termen liber.

Main-ul este implementat in clasa Gui unde este creat un obiect de tip JFrame care este instantiat ca si un obiect de tip Gui atribuindu-se astfel toate labelurile, butoanele si textfieldurile definite.

Librarii folosite:

- -java.awt.evemt, javax.swing pentru implementarea interfetei grafice;
- -java.util pentru utilizarea colectiilor;
- -org.junit.jupiter.api.Test folosita pentru testarea rezultatelor unor metode implementate.

5. Testare

In acest proiect pentru clasa Polinom am creat un Junit Test Case pentru a verifica operatiile de adunare, scadere, inmultire si impartire implementate in aceasta clasa.

Tip test	Date intrare	Rezultat asteptat	Rezultat obtinut	Pass/fa
				il
Adunar	Primul	2.0x^2+5.0x+4.0	2.0x^2+5.0x+4.0	Pass
e	polinom=2x^2+4			
	x+3			
	Al doilea			
	polinom=x+1			
Scader	Primul	$2.0x^2+3.0x+2.0$	$2.0x^2+3.0x+2.0$	Pass
e	polinom=2x^2+4			
	x+3			
	Al doilea			
	polinom=x+1			
Inmulti	Primul	$2.0x^3+6.0x^2+7.0x$	$2.0x^3+6.0x^2+7.0x$	Pass
re	polinom=2x^2+4	+3.0	+3.0	
	x+3			
	Al doilea			
	polinom=x+1			
Raport	Primul	2.0x+2.0	2.0x+2.0	Pass
	polinom=2x^2+4			
	x+3			
	Al doilea			
	polinom=x+1			



Printscreen-urile urmatoare arata ca testarea cu Junit a celor patru operatii s-a efectuat cu succes.

```
- -
🖺 Package ... 🗗 JUnit 🛭 🗀 📗 Polinom.java 📗 Monom.java 📗 Gui.java 🔛 *OperatiiPolinomTest.java 🗵
                              1⊕ import static org.junit.jupiter.api.Assertions.*;
 ↓ ↑ × □ □ □
                                 import org.junit.jupiter.api.Test;
 Q 🔒 🔳 🗒 🕶
                                 class OperatiiPolinomTest {
Finished after 0,106 seconds
                                     @Test
                                     void testSumaPolinom() {
 Runs: 4/4 

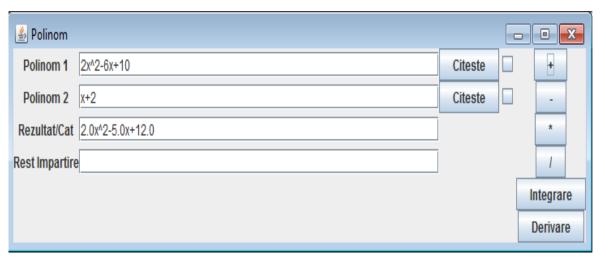
Errors: 0 

Failures: 0
                                        Polinom p=new Polinom():
                                        Polinom q=new Polinom();
                                        p.parsarePolinom("2x^2+4x+3");
                              10
                                        q.parsarePolinom("x+1");
 OperatiiPolinomTest [Runner: JUn]
                                        Polinom result=p.sumaPolinom(q);
     testDiferentaPolinom() (0,015 s
                                        assertEquals("2.0x^2+5.0x+4.0",result.afisarePolinom());
                              13
     testProdusPolinom() (0,001 s)
                              14
                              15⊖
     testRaportPolinom() (0,002 s)
                                     void testDiferentaPolinom() {
                              16
     testSumaPolinom() (0,008 s)
                              17
                                        Polinom p=new Polinom();
                                        Polinom q=new Polinom();
                              18
                              19
                                        p.parsarePolinom("2x^2+4x+3");
                                        q.parsarePolinom("x+1");
                              20
                                        Polinom result=p.diferentaPolinom(q);
                              21
                                        assertEquals("2.0x^2+3.0x+2.0",result.afisarePolinom());
                              22
                              23
                              249
                                     @Test
                              25
                                     void testProdusPolinom() {
              @Test
              void testProdusPolinom() {
                  Polinom p=new Polinom();
                  Polinom q=new Polinom();
                  p.parsarePolinom("2x^2+4x+3");
                  q.parsarePolinom("x+1");
                  Polinom result=p.produsPolinom(q);
                  assertEquals("2.0x^3+6.0x^2+7.0x+3.0",result.afisarePolinom());
              void testRaportPolinom() {
                  Polinom p=new Polinom();
                  Polinom q=new Polinom();
                  p.parsarePolinom("2x^2+4x+3");
                  q.parsarePolinom("x+1");
                  Polinom result=p.raportPolinom(q);
                  assertEquals("2.0x+2.0",result.afisarePolinom());
             }
         }
```

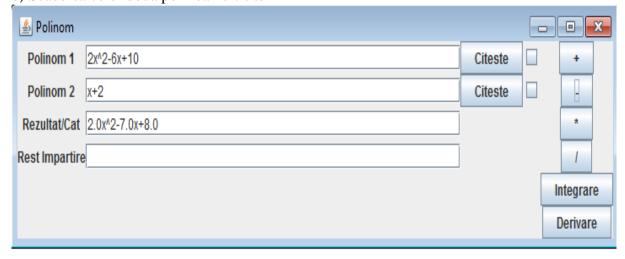
6. Rezultate

Dupa ce utilizatorul a introdus cele doua polinoame si a selectat butoanele de citire pentru a trimite textul spre parsare, el are posibilitatea de a efectua urmatoarele operatii cu polinoamele citite:

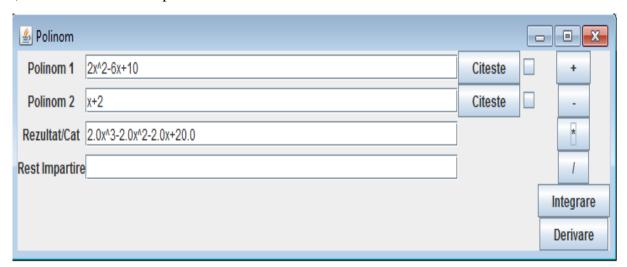
a) Adunarea celor doua polinoame citite



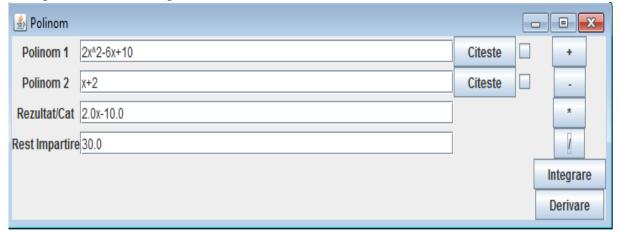
b) Scaderea celor doua polinoame citite



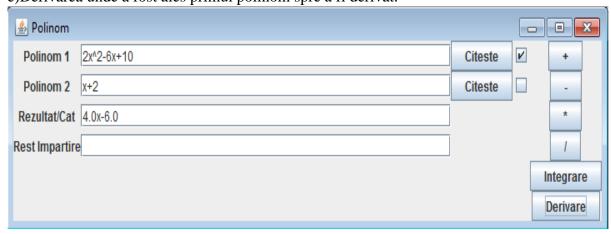
c)Inmultirea celor doua polinoame citite



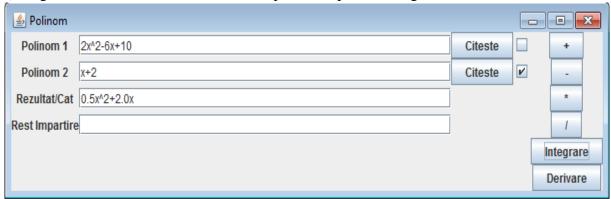
d) Impartirea celor doua polinoame citite si afisarea atat a catului cat si a restului



e)Derivarea unde a fost ales primul polinom spre a fi derivat.



f)Integrare aici a fost ales cel de-al doilea polinom spre a fi integrat.



7. Concluzii

In concluzie acest proiect implementeaza corect operatiile efectuate pe polinoame de orice ordin si ofera utilizatorului rezultatele dorite.

In urma rezolvarii acestei teme am aprofundat si am fixat cunostintele de programare orientata pe obiect invate semestrul trecut. Am invatat cum sa utilizez clasele de test pentru a-mi testa diferitele functii pe care trebuie sa le indeplineasca programul. Am invatat cum sa incarc, sa updatez, sa dau share unui proiect de tip Maven folosind bitbucket. Am invatat sa implementez o interfata grafica care poate fi utilizata usor de orice tip de utilizator. Am invatat sa scriu cod pentru un proiect functional pornind de la diagrama UML descrisa in cursul orelor de laborator.

De asemenea, am invatat cum sa imi gestionez timpul si resursele pentru a putea preda temele si proiectele la termenul de predare stabilit.

Acest proiect are o multime de posibilitati de dezvoltare ulterioara:

- -calcularea valorii polinomului intr-un punct x dat de catre utilizator;
- -calcularea derivatei de ordin n prin apelarea repetata a derivatei de ordin 1 deja implementata;
- -calcularea seriei Taylor intr-un punct dat avem nevoie doar de derivata de ordin n;
- -calcularea radacinilor pentru polinoame de grad 2;
- -calcularea radacinilor pentru polinoame de grad 3 folosind algoritmul lui Horner sau relatiile lui Viete;
- -ridicarea la putere a unui polinom.

8. Bibliografie

https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

https://stackoverflow.com/

http://www.mkyong.com/tutorials/junit-tutorials/

http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

http://www.coned.utcluj.ro/~marcel99/PT/

http://www.coned.utcluj.ro/~salomie/PT Lic/