**DOCUMENTATIE TEMA 1**

**CALCULATOR POLINOAME**

Cuprins:

1. Obiectivul temei
2. Analiza problemei
3. Proiectare
4. Implementare
5. Concluzii
6. Bibliografie

# Obiective

# Cerinte Functionale

( cerintele problemei )

Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

Operatiile pe care le suporta programul sunt:

* Citirea unui polinom de la tastatura sub forma de String cu formatul a0x^n+a1x^(n-1)+a2x^(n-2)+...+a0, unde si
* Adunarea a doua polinoame P1(x) + P2(x)
* Scaderea a doua polinoame P1(x) - P2(x)
* Inmultirea a doua plinoame P1(x) \* P2(x)
* Impartirea a doua polinoame P1(x) / P2(x)
* Derivarea unui polinom P1(x)’
* Integrarea unui polinom ∫ P1(x)

## Obiectiv Principal:

(obiectivul principal al proiectului)

Obiectivul principal al proiectului este de a creea o aplicatie care sa implementeze un sistem de procesare a polinoamelor ( acestea fiind construite din termeni numiti monoame, care sunt alcatuite dintr-o constanta (numita coeficient), inmultita cu una sau mai multe variabile; fiecare variabila poate avea un exponent constant intreg pozitiv). Incercam sa implemantam diferite operatii pentru aceste polinoame sis a avem grija de fiecare data dacaintalnim cazuri special (gradul polonomului P-primul introdus este mai mic decat al polinomului Q, introdus al doilea).

Polinoamele și operațiile cu acestea sunt unele dintre cele mai comune structuri matematice din Computer Science. Motivul principal este simplitatea algebrică a operațiilor – vorbim doar de adunări și înmulțiri (care la rândul lor sunt adunări repetate). Avantajul este că polinoamele pot aproxima oricât de bine orice funcție continuă și derivabilă, deci pot “interpola” funcții mult mai complexe.

Această strategie de conceptualizare, mai poartă numele și de bottom-up design. Este foarte avantajoasă din prisma găsirii componentelor constituente, deoarece pot fi găsite, relativ ușor, structuri cu o legătură directă în lumea reală( obiecte, acțiuni etc.). Din păcate această versatilitate vine cu prețul complexității, ea crescând pe măsură ce se avansează pe nivelele inferioare.

## Obective Secundare:

( pasii pe care trebuie sa-i urmam pentru a atinge obiectivul principal )

|  |  |
| --- | --- |
| **Obiectiv Secundar** | **Descriere** |
| Dezvoltarea de use case-uri si scenarii | Într-un sistem software un ‘use-case’ este o listă de acțiuni sau pași de eveniment care definesc în mod obișnuit interacțiunile dintre un rol ( cunoscut ca un actor în limbajul unificat de modelare ( UML ) ) și un sistem in atingerea unui obiectiv.  In computere, un scenariu este o naratiune a interactiunilor dintre roluri ( actori in UML ) si sistem. In cazul nostru rolul - un utilizator al aplicatiei. |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capat obiectivul principal |
| Impartirea pe clase | Folosirea unui MVC ( Model – View – Controller) pentru a putea forma un GUI ( Graphic User Interface ), a clasei Polinom, Monom etc. |
| Dezvoltarea algoritmilor | Vor fi descrise structurile de date necesare pentru atingerea obiectivului principal, schema UML ( diagrama de clase ) precum si algoritmii folositi pentru realizarea operatiilor. |
| Implementarea solutiei | Vor fi descrise fiecare clasa cu campurile si metodele importante precum si descrierea interfetei utilizator; |
| Testare | Vor fi descrise cateva scenarii de testare a operatiilor pe polinoame, folosind ca instrument un Junit Test. |

# Analiza problemei

**2.1. Use case-uri / Scenarii**

Utilizarea programului presupune introducerea de catre utilizator (in TextField-uri ) a celor doua polinoame ce urmeaza a fi folosite pentru efectuarea anumitor operatii ( adunare, scadere, inmultire, impartire, derivare, integrare ). Prin apasarea unor Butoane ce au semne specifice pentru operatii, intr-un alt TextField o sa apara rezultatul asteptat.

Deoarece e posibil ca unii utilizatori sa introduca polinoame gresite, de fiecare data cand un polinom introdus in TextField are un format gresit fata de cel normal al unui polinom, sau sunt introduse diferite Stringuri care nu formeza un polinom ( ex. Ana are mere ), la apasarea unui buton ce semnifica operatia va aparea o fereastra ce va avertiza utilizatorul ca polinomul introdus de el nu este valid ( nu respecta formatul ); utilizatorul trebuie sa fie atent sa scrie polinomul de la gradul cel mai mare spre cel mai mic.

# Proiectare

( etapele de proiectare )

## Structuri de date

(structurile de date folosite in program)

Ca si structura de data principala pe care am folosit-o este un ArrayList de Monoame (deoarece un polinom este format din mai multe monoame, un arraylist ar fi fost cel mai potrivit in rezolvarea acestei probleme ).

**private**List<Monomial> **argumente** = **new** ArrayList<Monomial>();

Un monom este format dintr-un coeficient, un grad si o variabila ( ex. 2X^3 , unde 2 – coeficientul, x – variabila, iar 3 – gradul ( puterea ) ). Monom-ul va fi in java o clasa separata care va contine ca variabile instanta un intreg, reprezentat de putere ( grad ) si un float care reprezinta coeficientul ( acesta va fi de tip float deoarece la anumite operatii, se poate obtine un numar care contine , : ex ½ =0.5 )

## Diagrama de clase

(schema UML a proiectului )



1. **Implemaentare**

Proiectul contine 4 clase(Polynomial, Monomial, Operations si clasa Main), fiecare implementand instructiuniile specific.

**Clasa Polynomial**: contine metodele de reprezentare a polinoameleor, in forma lor specificata si corecta.

**Clasa Monomial**: variabilele ceof (care reprezinta coeficientul polinomului) si variabila power (care reprezinta gradul polinomului)

**Clasa Operations**: aici sunt descries fiecare operatie in parte (adunare, scadere, inmultire, impartire, integrare, derivare). Exemlpu operatia de adunare si de scadere:

**public** Polynomial add(Polynomial P,Polynomial Q) {  
  
 Monomial m;  
 *//The returned polynomial* Polynomial PplusQ = **new** Polynomial();  
  
 **int** nrP = P.nrMonoms();  
 **int** nrQ = Q.nrMonoms();  
  
 **try**{  
 **while**(**i** < nrP && **j** < nrQ){  
 *//Coefficients and powers of P and Q* **double** coefP = P.getMonom(**i**).getCoef();  
 **double** coefQ = Q.getMonom(**j**).getCoef();  
 **double** powerP = P.getMonom(**i**).getPower();  
 **double** powerQ = Q.getMonom(**j**).getPower();  
  
 **if**(powerP == powerQ){  
 m = **new** Monomial(coefP + coefQ,powerP);  
 **i**++;**j**++;}  
  
 **else if**(powerP > powerQ){  
 m = **new** Monomial(coefP,powerP);  
 **i**++;  
 }  
 **else** {  
 m = **new** Monomial(coefQ,powerQ);  
 **j**++;  
 }  
 PplusQ.addMonom(m);  
 }  
 **while**(**i** < nrP) {  
 **double** coefP = P.getMonom(**i**).getCoef();  
 **double** powerP = P.getMonom(**i**).getPower();  
 m = **new** Monomial(coefP,powerP);  
 **i**++;  
 PplusQ.addMonom(m);  
 }  
 **while**(**j** < nrQ) {  
 **double** coefQ = Q.getMonom(**j**).getCoef();  
 **double** powerQ = Q.getMonom(**j**).getPower();  
 m = **new** Monomial(coefQ,powerQ);  
 **j**++;  
 PplusQ.addMonom(m);  
 }  
 }  
 **catch**(Exception E) {};  
 **return** PplusQ;  
}

**public** Polynomial sub(Polynomial P,Polynomial Q) {  
  
 **i** = 0;  
 **j** = 0;  
 Monomial m;  
 Polynomial PminusQ = **new** Polynomial();  
 **int** nrP = P.nrMonoms();  
 **int** nrQ = Q.nrMonoms();  
 **try**{  
 **while**(**i** < nrP && **j** < nrQ){  
  
 **double** coefP = P.getMonom(**i**).getCoef();  
 **double** coefQ = Q.getMonom(**j**).getCoef();  
 **double** powerP = P.getMonom(**i**).getPower();  
 **double** powerQ = Q.getMonom(**j**).getPower();  
  
 **if**(powerP == powerQ){  
 m = **new** Monomial(coefP - coefQ,powerP);  
 **i**++;**j**++;  
 }  
 **else if**(powerP > powerQ){  
 m = **new** Monomial(coefP,powerP);  
 **i**++;  
 }  
 **else** {  
 m = **new** Monomial(-coefQ,powerQ);  
 **j**++;  
 }  
  
 PminusQ.addMonom(m);  
 }  
  
 **while**(**i** < nrP) {  
 **double** coefP = P.getMonom(**i**).getCoef();  
 **double** powerP = P.getMonom(**i**).getPower();  
 m = **new** Monomial(coefP,powerP);  
 **i**++;  
 PminusQ.addMonom(m);  
 }  
 **while**(**j** < nrQ) {  
 **double** coefQ = Q.getMonom(**j**).getCoef();  
 **double** powerQ = Q.getMonom(**j**).getPower();  
 m = **new** Monomial(-coefQ,powerQ);  
 **j**++;  
 PminusQ.addMonom(m);  
 }  
 }  
 **catch**(Exception E) {};  
 **return** PminusQ;  
}

**Clasa Main**: contine efectiv crearea interfetei calculatorului, crearea fiecarui buton (butonul pentru operatie de adunare, scadere, inmultire, impartire) si de asemnea introducerea celor 2 polinoame si afisarea rezultatului dupa apasarea butonului pentru una dintre operatii. Mai exista si butonul de reset in cazul in care utilizatorul doreste sa isi schime decizia sis a sterga datele introduce si rezultatul.

Exemple de cod pentru creare interfetei

**Citirea polinoamelor P si Q:**

**if**(**readP**) {  
 textP = **editarePolinomP**.getText();*//get the text from the text field* **if**(textP.startsWith(**"-"**)) {hasCoef = **true**;  
 negative = **true**;}  
 **if**(textP.startsWith(**"x"**)) {hasCoef = **true**;  
 negative = **false**;}  
 **if**(hasCoef == **true**) {  
  
 **if**(negative == **true**) {cMonom = -1;textP = textP.substring(1);}  
 **if**(negative == **false**) cMonom = 1;  
  
 }  
 StringTokenizer st1 = **new** StringTokenizer(textP,**"x^"**);  
  
 **while** (st1.hasMoreElements()){  
 inter = (String) st1.nextElement();  
  
 *//Forward dividing into integers* StringTokenizer st2 = **new** StringTokenizer(inter,**"+"**);  
 StringTokenizer st3 = **new** StringTokenizer(inter,**"-"**);  
  
 **if**(inter.indexOf(**"-"**) != -1) *//if it has a - operator then we use* **while**(st3.hasMoreElements()){*//this part of code to break the string* **try**{  
 **if**(cMonom == 0) {  
 String copy = (String) st3.nextElement();  
 cMonom = Integer.*parseInt*(copy);  
 cMonom = - cMonom;  
 **continue**;  
 }  
  
 **else  
 if** (iMonom == 0) {  
 String copy = (String) st3.nextElement();  
 iMonom = Integer.*parseInt*(copy);  
 **continue**;  
 }  
 }  
 *//NumberFormatException could be thrown* **catch**(Exception e) {popUp(mesajFormat);}  
 *//We add the monom to the polynom* Monomial monom = **new** Monomial(cMonom,iMonom);  
 *polinomP*.addMonom(monom);  
 iMonom = 0;  
 cMonom = 0;  
 }  
 **else** *//it doesn't containt '-'* **while**(st2.hasMoreElements()){  
  
 **if**(cMonom == 0) { String value = (String) st2.nextElement();  
 **try**{cMonom = Integer.*parseInt*(value);  
 }  
 **catch**(Exception e) {popUp(mesajFormat);}  
 **continue**;  
 }  
  
 **else  
 if**(iMonom == 0) { String value = (String) st2.nextElement();  
 **try**{iMonom = Integer.*parseInt*(value);  
 }  
 **catch**(Exception e) {popUp(mesajFormat);}  
  
 **continue**;  
 }  
 *//Instantiate and add the monom to P* Monomial monom = **new** Monomial(cMonom,iMonom);  
 *polinomP*.addMonom(monom);  
 *//reseting the values* iMonom = 0;  
 cMonom = 0;  
 }  
  
 }  
 **if**(cMonom != 0) {  
  
 Monomial monom = **new** Monomial(cMonom,iMonom);  
 *polinomP*.addMonom(monom);  
  
 }  
 **readP** = **false**;*//We read P only once until reset button is pressed*}

**if**(**readQ**) {  
 textQ = **editarePolinomQ**.getText();  
 *//reset the values* iMonom = 0;  
 cMonom = 0;  
 hasCoef = **false**;  
 negative = **false**;  
  
 **if**(textQ.startsWith(**"-"**)) {hasCoef = **true**;  
 negative = **true**;}  
 **if**(textQ.startsWith(**"x"**)) {hasCoef = **true**;  
 negative = **false**;}  
  
 **if**(hasCoef == **true**) {  
  
 **if**(negative == **true**) {cMonom = -1;  
 textQ = textQ.substring(1);}  
 **if**(negative == **false**) cMonom = 1;  
  
 }  
  
 StringTokenizer stQ1 = **new** StringTokenizer(textQ,**"x^"**);  
  
 **while** (stQ1.hasMoreElements()){  
  
 interQ = (String) stQ1.nextElement();  
  
 StringTokenizer stQ2 = **new** StringTokenizer(interQ,**"+"**);  
 StringTokenizer stQ3 = **new** StringTokenizer(interQ,**"-"**);  
  
 **if**(interQ.indexOf(**"-"**) != -1) *//if it has a - operator then we use* **while**(stQ3.hasMoreElements()){*//this part of code to break the string* **try**{  
 **if**(cMonom == 0) {String copy = (String) stQ3.nextElement();  
 cMonom = Integer.*parseInt*(copy);  
 cMonom = -cMonom;  
 **continue**;  
 }  
  
 **else  
 if** (iMonom == 0) {String copy = (String) stQ3.nextElement();  
 iMonom = Integer.*parseInt*(copy);  
  
 **continue**;  
 }  
 }  
 **catch**(Exception e) {popUp(mesajFormat);}  
  
 *//Instantiate and add the monom to the polynomial* Monomial monom = **new** Monomial(cMonom,iMonom);  
 *polinomQ*.addMonom(monom);  
 *//reset the values* iMonom = 0;  
 cMonom = 0;  
 }  
 **else  
 while**(stQ2.hasMoreElements()){*//It has '+' as a token* **if**(cMonom == 0) { String value = (String) stQ2.nextElement();  
 **try**{cMonom = Integer.*parseInt*(value);}  
 **catch**(Exception e) {popUp(mesajFormat);}  
 **continue**;  
 }  
  
 **else  
 if**(iMonom == 0) { String value = (String) stQ2.nextElement();  
 **try**{iMonom = Integer.*parseInt*(value); }  
 **catch**(Exception e) {popUp(mesajFormat);}  
 **continue**;  
 }  
 *//Instantiate and add the monom to the polynomial* Monomial monom = **new** Monomial(cMonom,iMonom);  
 System.***out***.println(**"Creat monomul :"** + cMonom + **" , "** + iMonom);  
 *polinomQ*.addMonom(monom);  
 iMonom = 0;  
 cMonom = 0;  
 }  
 *//A test display to see if the reading is done properly* System.***out***.println(**"Polinomul P dupa citire "** + *polinomP*.toString());  
 System.***out***.println(**"Polinomul Q dupa citire "** + *polinomQ*.toString());  
  
 }

**public** Polynomial mul(Polynomial P,Polynomial Q) {  
 **int** i = 0, j = 0;  
 Monomial m;  
 Polynomial PmulQ = **new** Polynomial();  
  
 **try**{  
 **int** nrP = P.nrMonoms();  
 **int** nrQ = Q.nrMonoms();  
  
 **while**(i < nrP){  
 **double** coefP = P.getMonom(i).getCoef();  
 **double** powerP = P.getMonom(i).getPower();  
 i++;  
 j = 0;  
  
 **while**(j < nrQ){  
 **double** coefQ = Q.getMonom(j).getCoef();  
 **double** powerQ = Q.getMonom(j).getPower();  
 m = **new** Monomial(coefP \* coefQ, powerP + powerQ);  
 PmulQ.addMonom(m);  
 j++;  
 }  
 }  
  
 }**catch**(Exception E){}  
 **return** PmulQ;  
}

**Construirea butoanelor:**

ButtonGroup grup = **new** ButtonGroup();  
grup.add(**unary**);  
**poliP** = **new** JLabel(**"Polinomul P :"**,JLabel.***LEFT***);  
**c**.**gridx** = 0;  
**c**.**gridy** = 1;  
pane.add(**poliP**, **c**);  
  
**editarePolinomP** = **new** JTextField(10);  
**editarePolinomP**.setActionCommand(**"Ceva text"**);  
**c**.**gridx** ++;  
pane.add(**editarePolinomP**, **c**);  
**poliQ** = **new** JLabel(**"Polinomul Q :"**,JLabel.***LEFT***);  
**c**.**gridx** = 0;  
**c**.**gridy** = 2;  
pane.add(**poliQ**, **c**);

**add** = **new** JButton(**"+"**);  
**c**.**gridx** = 3;  
**c**.**gridy** = 1;  
pane.add(**add**, **c**);  
**add**.addActionListener(**this**);  
  
**sub** = **new** JButton(**"-"**);  
**c**.**gridx**++;  
**sub**.addActionListener(**this**);  
pane.add(**sub**, **c**);  
  
**mul** = **new** JButton(**"\*"**);  
**c**.**gridx**++;  
**mul**.addActionListener(**this**);  
pane.add(**mul**, **c**);  
  
**div** = **new** JButton(**"/"**);  
**div**.addActionListener(**this**);  
**c**.**gridx**++;  
pane.add(**div**, **c**);

**result** = **new** JLabel(**"Rezultat :"**,JLabel.***LEFT***);  
**c**.**gridx** = 0;  
**c**.**gridy** = 3;  
pane.add(**result**,**c**);  
**result**.setFont(**result**.getFont().deriveFont(18.0f));

**resultTextField** = **new** JTextField(20);  
 **c**.**gridx**++;  
 **c**.**ipadx** = 15;  
 **c**.**ipady** = 10;  
 **c**.**anchor** = GridBagConstraints.***LAST\_LINE\_START***;  
 **resultTextField**.setEditable(**false**);  
 **resultTextField**.setBackground(**new** Color(180,190,210));  
 pane.add(**resultTextField**,**c**);  
 **c**.**gridx** = 0;  
 **c**.**gridy** = 0;  
  
 pane.add(Box.*createRigidArea*(**new** Dimension(50,0)));  
  
 **reset** = **new** JButton(**"C"**);  
 **reset**.addActionListener(**this**);  
 **reset**.setFont(**reset**.getFont().deriveFont(9.0f));  
 **c**.**gridy** = 3;  
 **c**.**gridx** = 6;  
 **c**.**ipadx** = 1;  
 **c**.**ipady** = 1;  
 pane.add(**reset**,**c**);  
  
}

## 

1. **Concluzii**

Cu ajutorul acestui priect am reusit sa imi reamintesc notiunile de programare orientate pe obiect din semestrul trecut. Am incercat sa implementez cat mai corect operatiile cerute in ducumentatie acestui proiect, desi la unele am avut problem. De aceea cred ca tema realizata de mine mai are nevoie de imbunatatiri pentru o dezvoltare ulterioara cat mai corecta si simplista. Sper ca proiectul sa fie usor de inteles pentru utilizatori, problemele pe care le rezolva fiind de nivel mediu.

1. **Bibliografie**

Resursle care m-au ajutat in dezvoltarea acestui priect sunt:

<https://www.youtube.com/watch?v=jTKYU3cKY18>

<https://www.symbolab.com/solver/polynomial-equation-calculator>