DOCUMENTATIE TEMA 1

CALCULATOR POLINOAME

Nume si prenume: Fagadar Ionela-Catalina

Grupa: 30226

Profesor Laborator: Marin Oana

Contents

[1.Obiectivul temei 4](#_Toc3413562)

[2.Analiza problemei, modelare, scenarii, cazuri de utilizare 4](#_Toc3413563)

[3.Proiectare 4](#_Toc3413564)

[3.1. Decizii de proiectare 4](#_Toc3413565)

[3.2. Diagrame UML 4](#_Toc3413566)

[3.3. Structuri de date 4](#_Toc3413567)

[3.4. Proiectare clase 4](#_Toc3413568)

[3.5. Proiectare interfete 5](#_Toc3413569)

[3.6. Relatii 5](#_Toc3413570)

[3.7. Pachete 5](#_Toc3413571)

[3.8. Algoritmi 5](#_Toc3413572)

[3.8.0. Algoritmul de parsare 5](#_Toc3413573)

[3.8.1. Algoritmul de adunare 6](#_Toc3413574)

[3.8.2. Algoritmul de scadere 6](#_Toc3413575)

[3.8.3. Algoritmul de inmultire 6](#_Toc3413576)

[3.8.4.Algoritmul de impartire 6](#_Toc3413577)

[3.8.5. Algoritmul de derivare 7](#_Toc3413578)

[3.8.6. Algoritmul de integrare 7](#_Toc3413579)

[4. Implementare 7](#_Toc3413580)

[4.1. Clasa Monom 7](#_Toc3413581)

[4.2. Clasa Polinom 7](#_Toc3413582)

[4.3. Clasa Parsare 7](#_Toc3413583)

[4.4. Clasele Adunare, Scadere, Inmultire, Impartire, Derivare, Integrare 8](#_Toc3413584)

[4.5. Clasa PolinomModel 8](#_Toc3413585)

[4.6. Clasa PolinomView 8](#_Toc3413586)

[4.7. Clasa PolinomController 8](#_Toc3413587)

[4.8. Clasa PolinomMVC 8](#_Toc3413588)

[5.Rezultate 9](#_Toc3413589)

[6.Concluzii si dezvoltari ulterioare 9](#_Toc3413590)

[7.Bibliografie 9](#_Toc3413591)

# **1****.Obiectivul temei**

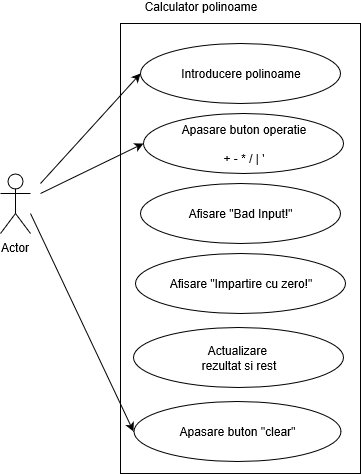
Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

|  |  |  |
| --- | --- | --- |
| Obiectiv | Descriere | Capitol |
| dezvoltarea de use case-uri si scenarii |  | 2 |
| alegerea structurilor de date |  | 3 |
| impartirea pe clase |  | 3 |
| dezvoltarea algoritmilor |  | 3 |
| implementarea solutiei |  | 4 |
| testare |  | 5 |

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

Programul este conceput astfel incat, utilizatorul introduce doua polinoame de la tastatura, sub forma „coeficient\*x^putere” ( de ex: -2\*x^3+4\*x-3)(nu trebuie sa contina toate puterile).Dupa ce s-au introdus ambele polinoame, se apasa pe una dintre operatiile constituite din butoane: +, -, \*, /. Pentru operatiile |, ‘ se poate introduce doar polinomul asupra caruia se efectueaza operatia. Dupa apasare, rezultatul, si restul (in cazul impartirii) este afisat in textbox. Butonul de clear sterge rezultatele. In cazul in care, utilizatorul introduce un polinom in care monoamele sale se repeta (sunt la aceeasi putere), calculatorul de polinoame aduna implicit cele doua monoame, astfel, daca s-a introdus “x^2+x^2”, polinomul se transforma in “2\*x^2”. In cazul in care, se introduce un polinom a carui variabila e diferita de „x”, programul afiseaza un mesaj de eroare „Bad Input!”, la fel ca si in cazul in care nu respectam introducerea polinoamelor sub forma „coeficient\*x^putere”. In cazul in care nu introducem un polinom, calculatorul de polinoame NU il considera polinomul nul. La impartirea cu zero, programul va afisa un mesaj de eroare „Impartire cu zero!”.

Diagrama use-case a acestui proiect este:



# **3.Proiectare**

## **3.1. Decizii de proiectare**

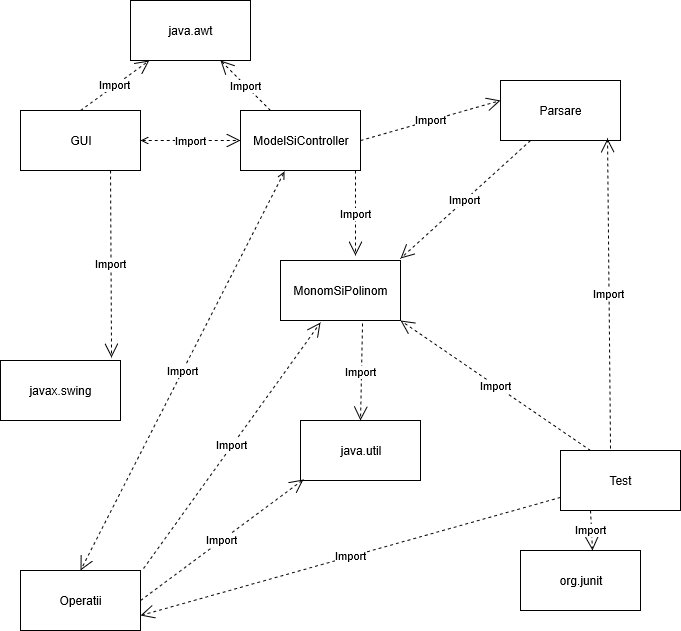
Am ales sa proiectez acest proiect in modul MVC: Model View Controller, impartindu-l in 6 pachete: ModelSiController, GUI, MonomSiPolinom, Operatii, Parsare, Test. Initial, am ales ca polinomul sa fie citit sub forma de sir de caractere, de la tastatura, apoi parsat ( clasa Parsare se ocupa de aceasta operatie+validare polinom) , fiind despartit in Monoame (clasa Monom cu campurile coeficient si putere), iar fiecare Monom sa apartina Polinomului ( clasa Polinom cu campul poli care contine monoamele polinomului). Fiecare operatie aplicata polinomului este construita intr-o clasa ( Adunare, Scadere, Inmultire, Impartire, Derivare, Integrare) care implementeaza interfata Operatie. Testele aplicate proiectului se afla in pachetul Test.

## **3.2. Diagrame UML**

Diagrama de clase a acestui proiect:

## 

Diagrama de pachete a acestui proiect:



## 

## **3.3. Structuri de date**

Singura structura de date folosita in acest proiect este Lista, folosita in cadrul clasei polinom. Aceasta contine o lista sub forma de ArrayList de monoame, pentru a fi stocat intregul polinom introdus de la tastatura.

## **3.4. Proiectare clase**

Se foloseste o clasa pentru parsarea sirului de caractere introdus de la tastatura (clasa Parsare),se folosesc clase pentru stocarea fiecarui monom din polinom (clasa Monom), o clasa pentru a stoca monoamele obtinute (clasa Polinom), sase clase pentru operatiile efectuate pe polinoame (Adunare, Scadere, Inmultire, Impartire, Derivare, Integrare), o clasa pentru interfata cu utilizatorul (clasa PolinomView – view din modelul MVC), o clasa pentru controlul aplicatiei (clasa PolinomController – controller din modelul MVC – se efectueaza controlul asupra butoanelor din interfata), o clasa pentru modelul aplicatiei (clasa PolinomModel – model din modelul MVC – contine calculele care descriu comportamentul butoanelor din interfata), o clasa MVC pentru a imbina clasele PolinomModel, PolinomController, PolinomView (clasa PolinomMVC – clasa care contine metoda main() ), cat si o clasa pentru testarile JUnit ale aplicatiei (clasa TestPolinom).

## **3.5. Proiectare interfete**

Singura interfata construita este Operatie, care descrie comportamentul operatiilor de adunare, scadere, inmultire, impartire, derivare, integrare. Aceasta contine antetul a doua metode: Polinom calculare(Polinom p1, Polinom p2); si Polinom calculare(Polinom p); care descriu cele doua metode ce fiecare clasa de operatie trebuie sa le implementeze. S-au ales doua metode ce compun interfata deoarece adunarea, scaderea, inmultirea si impartirea sunt operatii ce necesita doi operanzi ( doua polinoame), iar derivarea si integrarea sunt operatii ce necesita doar un operand (un polinom).

## **3.6. Relatii**

Relatiile folosite in acest proiect sunt relatii de mostenire intre Polinom si Monom (clasa Polinom extinde clasa Monom), cat si intre clasa Parsare si Polinom (clasa Parsare extinde clasa Polinom) , clasa PolinomView si clasa JFrame (clasa PolinomView extinde clasa JFrame), relatii de implementare intre cele 6 operatii : adunare, scadere, inmultire, impartire, derivare, integrare si interfata Operatii ( de exemplu clasa Adunare implementeaza interfata Operatie).

## **3.7. Pachete**

Proiectul e structurat in 6 pachete: ModelSiController, GUI, MonomSiPolinom, Operatii, Parsare si Test. Pachetul ModelSiController contine clase ce determina logica si controlul interfetei cu utilizatorul (contine clasele PolinomController, PolinomModel, PolinomMVC). Pachetul GUI contine o singura clasa ce determina interfata grafica a programului, mai exact modul cum sunt afisate butoanele, casetele text, etichetele ( contine clasa PolinomView). Pachetul MonomSiPolinom contine modul de impartire a polinomului introdus in monoame (contine clasele Monom si Polinom), iar pachetul Parsare contine o clasa ce executa parsarea sirului de caractere introdus (contine clasa Parsare). Pachetul Operatii contine operatiile implementate pentru polinoame (contine clasele Adunare, Scadere, Inmultire, Impartire, Derivare, Integrare, cat si interfata Operatie). Pachetul Test efectueaza testarile JUnit pentru proiect (contine clasa TestPolinom).

S-au importat pachete implicite java, ca java.math.\*; java.util.\*; java.awt.event.ActionEvent; java.awt.event.ActionListener;java.awt.\*;

javax.swing.\*; java.awt.event.\*; org.junit.Test; org.junit.Assert.assertEquals;org.junit.Assert.*assertFalse*;org.junit.Assert.*assertNotEquals*;org.junit.Assert.*assertTrue*;

## **3.8. Algoritmi**

Algoritmii utilizati pentru acest proiect sunt algoritmii utilizati pentru operatii: adunare, scadere, inmultire, impartire, derivare, integrare, cat si algoritmul folosit pentru parsarea sirului de caractere corespunzator polinomului.

## **3.8.0. Algoritmul de parsare**

Acest algoritm are doua intrari, polinomul si sirul de caractere. La terminarea acestuia, polinomul trebuie sa contina monoame corespunzatoare sirului de caractere. Pentru a se face o parsare mai usor, am ales sa inlocuiesc toate semnele minus cu semne plus minus (adica -x devine +-x). Se verifica daca sirul de caractere incepea cu - , iar in caz afirmativ se taie primul caracter, care ar fi trebuit sa fie + (un string -x-2 devine +-x+-2, iar in pasul urmator se taie primul caracter pentru a deveni -x+-2). Aceste operatii de mai sus se fac pentru a putea imparti sirul de caractere in subsiruri despartite DOAR de semnul plus. Astfel, se creeaza un vector de siruri de caractere subs care corespund subsirurilor din sirul de caractere introdus. Pentru fiecare subsir, se procedeaza in urmatorul fel: se cauta semnul \*, daca se gaseste, coeficientul este substringul pana la semn, daca nu se gaseste, se cauta caracterul „x”. Daca acesta nu este gasit, puterea este zero, iar coeficientul este exact subsirul. In schimb, daca se gaseste, se verifica daca primul caracter este „-„ iar al doilea este „x”, iar coeficientul devine -1, iar daca primul caracter este „x”, coeficientul devine 1. In continuare, se cauta puterea polinomului, adica caracterul „^”. Daca este gasit, puterea devine subsirul de la indexul caracterului „^” pana la sfarsit, iar daca nu este gasit, se verifica daca nu s-a atribuit deja puterea cu zero, iar in cazul in care nu, puterea devine 1. In urmatoarea etapa se verifica daca se introduce un string de forma : „x^2+2+x^2”. Pentru fiecare monom din polinomul curent, se verifica daca puerea monomului rezultat din calcule este egala cu puterea monoamelor polinomului curent, iar in caz afirmativ, se aduna coeficientii. In cazul in care nu este gasit, monomul calculat se adauga in polinomul curent.

## **3.8.1. Algoritmul de adunare**

In algoritmul de adunare se compara gradele celor doua polinoame si se alege ca p1 ( polinomul 1) sa aiba gradul cel mai mare dintre cele doua polinoame. Cautam printre monoamele polinomului 2, monoamele corespunzatoare polinomului 1. Daca monomul polinomului 2 are puterea egala cu monomul polinomului 1, atunci coeficientii celor doua monoame se aduna, puterea ramane neschimbata, iar in polinomul suma rezultat se adauga acest monom rezultat. Daca insa nu se gaseste corespondenta intre puterea monomului polinomului 1 si puterea monomului polinomului 2, monomul polinomului 1 se adauga direct in polinomul suma. La sfarsit, se cauta monoamele din polinomul 2 care nu corespund cu niciun monom din polinomul 1 (nu au puterile egale) si se introduc in polinomul suma. Se sorteaza polinomul rezultat, apoi, se cauta coeficientii nenuli si se adauga in polinomul rezultat. Se verifica daca polinomul este gol, iar in caz afirmativ, se adauga un monom nul. Acest polinom e stocat in atributul clasei Adunare.

## **3.8.2. Algoritmul de scadere**

Acest algoritm se executa in acelasi fel ca si cel de adunare, diferenta apare doar in momentul in care gasim puterea monomului polinomului 1 egala cu puterea monomului polinomului 2, coeficientul rezultat este compus din diferenta coeficientului monomului polinomului 1 si coeficientul monomului polinomului 2. Atunci cand se cauta monoame din polinomul 2 a caror putere nu este egala cu nicio putere a monomului polinomului 1, in polinomul diferenta se adauga un monom ce are coeficientul de semn contrar si puterea egala cu a monomului anterior. Acest polinom e stocat in atributul clasei Scadere.

## **3.8.3. Algoritmul de inmultire**

In acest algoritm, se iau toate monoamele polinomului 1 cu toare monoamele polinomului 2. Coeficientul rezultat este compus din produsul celor doi coeficienti al celor doua monoame, iar puterea rezultat este compusa din suma celor doua puteri ale celor doua monoame. Rezulta un monom ce se adauga la polinomul rezultat, numit produs. Se sorteaza acest polinom. Pentru fiecare monom din polinomul produs, se verifica daca puterea acestuia coincide cu alt monom din acelasi polinom, iar in caz afirmativ, se aduna coeficientii celor doua monoame si se adauga in polinomul rezultat. La sfarsit, se verifica daca polinomul este gol, iar in acest caz se adauga un monom nul, iar in caz negativ se elimina duplicatele din polinomul rezultat. Acest polinom e stocat in atributul clasei Inmultire.

## **3.8.4.Algoritmul de impartire**

Primul pas din acest algoritm este verificarea gradului celor doua polinoame. Adaca gradul polinomului 1 este mai mic decat gradul polinomului 2, atunci rezultatul primeste un polinom nul, adica valoarea zero, iar restul primeste valoarea deimpartitului, adica a polinomului 1, iar algoritmul se termina. Daca in schimb, gradul polinomului 1 este mai mare decat gradul polinomului 2, impartirea se poate realiza. In continuare, se verifica daca polinomul 2 este egal cu polinomul zero, adica daca se face o impartire la zero, iar in acest caz se arunca o exceptie („Impartire cu zero!”). Totusi, este verificat si cazul in care al doilea polinom este o constanta, iar in acest caz, toti coeficientii monoamelor sunt impartit la constanta, puterea nu se schimba, iar monoamele rezultate se adauga in campul m\_rezultat.Nu avem rest! In cazul in care nu ne aflam in niciuna din cazurile de mai sus, impartirea clasica are sens. Se copiaza valoarea polinomului 1 intr-un polinom denumit rest. Cat timp gradul polinomului rest este mai mare decat gradul polinomului 1, se creeaza un monom ce are ca coeficient impartirea coeficientilor polinoamelor, iar ca putere, diferenta dintre cele mai mari puteri ale monoamelor polinoamelor 1 si 2 (grad polinom 1 – grad polinom 2). In polinomul ParteCat se memoreaza ultimul monom adaugat. „Cat” este un polinom in care se regaseste catul impartirii. Polinomul aux contine rezultatul inmultirii ultimului monom adaugat (parteCat) cu polinomul y. Se executa diferenta dintre rest si aux, iar rezultatul se stocheaza in polinomul rest. Din polinomul parteCat se va sterge monomul, astfel incat la pasul urmator sa fie gol si sa se poata introduce urmatorul monom. Se sorteaza polinomul cat si polinomul rest, si se stocheaza in atributele clasei Imparire.

## **3.8.5. Algoritmul de derivare**

In acest algoritm, pentru fiecare monom din polinom, se verifica daca puterea acestuia este diferita de zero, iar in caz afirmativ in polinomul m\_rezultat se adauga un monom cu coeficientul compus din inmultirea coeficientului cu puterea monomului si cu puterea compusa din puterea monomului micsorata cu o unitate. La fel, se verifica daca polinomul m\_rezultat este gol, iar in caz afirmativ se adauga un monom nul in m\_rezultat.

## **3.8.6. Algoritmul de integrare**

In acest algoritm, pentru fiecare monom din polinom, se adauga in polinomul m\_rezultat, un monom cu coeficientul compus din impartirea coeficientului cu puterea monomului adunata cu 1 si cu puterea compusa din puterea monomului la care se adauga o unitate. La fel, se verifica daca polinomul m\_rezultat este gol, iar in caz afirmativ se adauga un monom nul in m\_rezultat.

# **4. Implementare**

## **4.1. Clasa Monom**

Atribute: coeficient, putere

Constructori , Getter, Setter.

## **4.2. Clasa Polinom**

Atribute: lista monoame

Constructori , Getter, Setter, metoda grad() – returneaza gradul polinomului, esteGol() – returneaza daca polinomul este gol sau nu, adaugaMonom(Monom m) – adauga un monom in lista de monoame, stergeMonom(Monom m) – sterge un monom din lista de monoame, contineMonom(Monom m) – returneaza true daca lista de polinonoame contine monomul, goleste() – sterge lista de monoame, eliminaDuplicate(), copiaza(Polinom p), equals(Polinom p).

## **4.3. Clasa Parsare**

Atribute: nu

Metoda converteste ce a fost prezentata in algoritmul de parsare.

Metoda gasireMonomCuAceeasiPutere(Monom m) care verifica daca in polinom exista un monom la acceasi putere, in caz afirmativ se schimba coeficientul acestuia cu suma coeficientilor celor doua monoame, iar in caz negativ, este pur si simplu introdus in polinom.

## **4.4. Clasele Adunare, Scadere, Inmultire, Impartire, Derivare, Integrare**

Atribute: m\_rezultat sau m\_rezultat si m\_rezultat2 (in cazul impartirii)

Metode: calculare(Polinom p1, Polinom p2) si calculare (Polinom p1)

In cazul clasei Impartire, se adauga metode ca:

Metoda initializare(), care sterge continutul rezultatului si restului, metoda gradePolinoameGresite(Polinom x,Polinom y) care face impartirea in cazul in care gradul numitorului e mai mare decat gradul numaratorului, metoda impartireCuZero(Polinom y) care verifica daca urmeaza sa se faca o impartire cu zero, iar in caz afirmativ, se arunca o exceptie, metoda polinoameDeGradZero(Polinom x,Polinom y) face impartirea in cazul in care numitorul e doar un coeficient.

## **4.5. Clasa PolinomModel**

Atribute: m\_rezultat, m\_rezultat2

Contructori, Getter, Setter.

Metode: add(Polinom p1,Polinom p2), sub(Polinom x, Polinom y), mul(Polinom x, Polinom y), div(Polinom x, Polinom y), deriv(Polinom x), integrala(Polinom x) se efectueaza operatiile si se memoreaza in campurile m\_rezultat si m\_rezultat2. Metoda round(double nr, int zecimale) care face rotunjirea unui nr in virgula flotanta. Metode getValue() si getValue2() care fac convertirea inversa a polinomului, adica converteste polinomul in string.

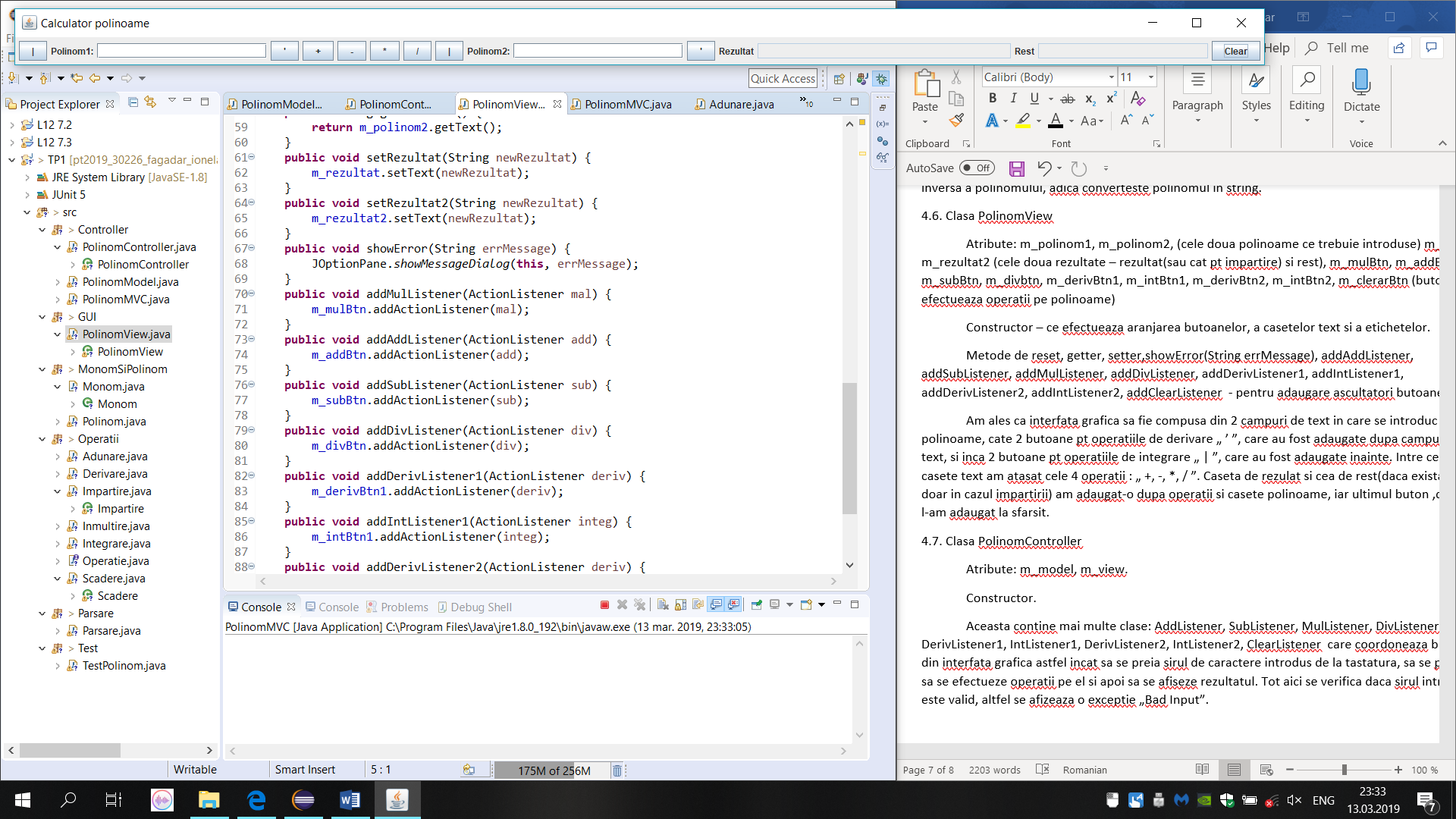
## **4.6. Clasa PolinomView**

Atribute: m\_polinom1, m\_polinom2, (cele doua polinoame ce trebuie introduse) m\_rezultat, m\_rezultat2 (cele doua rezultate – rezultat(sau cat pt impartire) si rest), m\_mulBtn, m\_addBtn, m\_subBtn, m\_divbtn, m\_derivBtn1, m\_intBtn1, m\_derivBtn2, m\_intBtn2, m\_clerarBtn (butoanele ce efectueaza operatii pe polinoame)

Constructor – ce efectueaza aranjarea butoanelor, a casetelor text si a etichetelor.

Metode de reset, getter, setter,showError(String errMessage), addAddListener, addSubListener, addMulListener, addDivListener, addDerivListener1, addIntListener1, addDerivListener2, addIntListener2, addClearListener - pentru adaugare ascultatori butoane.

Am ales ca interfata grafica sa fie compusa din 2 campuri de text in care se introduc polinoame, cate 2 butoane pt operatiile de derivare „ ’ ”, care au fost adaugate dupa campurile de text, si inca 2 butoane pt operatiile de integrare „ | ”, care au fost adaugate inainte. Intre cele doua casete text am atasat cele 4 operatii : „ +, -, \*, / ”. Caseta de rezulat si cea de rest(daca exista rest – doar in cazul impartirii) am adaugat-o dupa operatii si casete polinoame, iar ultimul buton ,de reset, l-am adaugat la sfarsit. Aceasta arata in felul urmator:



## **4.7. Clasa PolinomController**

Atribute: m\_model, m\_view.

Constructor.

Aceasta contine mai multe clase: AddListener, SubListener, MulListener, DivListener, DerivListener1, IntListener1, DerivListener2, IntListener2, ClearListener care coordoneaza butoanele din interfata grafica astfel incat sa se preia sirul de caractere introdus de la tastatura, sa se parseze, sa se efectueze operatii pe el si apoi sa se afiseze rezultatul. Tot aici se verifica daca sirul introdus este valid, altfel se afizeaza o exceptie „Bad Input”.

## **4.8. Clasa PolinomMVC**

Aici se utilizeaza PolinomController, PolinomView si PolinomModel. Aceasta clasa contine doar metoda main.

# **5.Rezultate**

In clasa TestPolinom s-au efectuat testari folosing utilitarul JUnit, cate doua pentru fiecare operatie, si cate 2 pentru operatia de parsare, una cu succes, iar alta nu. Testarile pentru parsare s-au efectuat in urmatorul mod: s-a ales un sir de caractere, care a incercat sa fie convertit. Daca aceasta operatie nu se putea face, atunci aparea o exceptie, care era tratata, iar in cazul in care era succes, se testa daca polinomul rezultat corespunde cu polinomul descris de sirul de caractere. Pentru operatiile aritmetice, s-au ales 2 siruri de caractere, sau unul, dupa caz, s-au convertit la polinoame si apoi s-a creat un polinom numit test care contine rezultatul asteptat in urma operatiei. Se testeaza daca rezultatul operatiei este egal cu polinomul test. La testarile pe operatii, s-au ales doua exemple, unul in care rezultatul asteptat era unul corect introdus, iar unul in care rezultatul asteptat era unul grasit, pentru a putea si observata corectitudinea metodelor.

# **6.Concluzii si dezvoltari ulterioare**

In concluzie, consider ca acest proiect este de ajutor pentru persoanele care invata lucrul cu polinoame, deoarece este o varianta utila de a rezolva in timp scurt, operatii cu acestea. In acest proiect s-a invatat structurarea acestuia in pachete, lucrul cu polinoamele.

Ca si dezvoltari ulterioare, programul ar putea genera pasii pe care ii face pentru rezolvare, proiectul putand fi utilizat ca material didactic auxiliar. La acesta s-ar mai putea modifica afisarea rezulatului, astfel: cand coeficientul e un intreg, sa nu fie reprezentat prin virgula mobila ( numarul 3.0 sa fie reprezentat simplu 3). O alta dezolvare ar consta in cazul in care utilizatorul nu foloseste „\*” si „^”, astfel, introducerea de exemplu a unui polinom x2+2x+1 sa fie interceptat ca x^2+2\*x+1.

Pentru a se putea intelege foarte bine operatiile pe polinoame, se recomanda studierea operatiilor matematice pe acestea, mai ales operatia de impartire.

# **7.Bibliografie**

* **Junit:** http://www.mkyong.com/tutorials/junit-tutorials/
* **Add libraries Eclipse:** http://www.wikihow.com/Add-JARs-to-Project-Build-Paths-in-Eclipse-%28Java%29
* **Swing:** http://zetcode.com/tutorials/javaswingtutorial/ http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
* **Books:** SCJP, Java 8 & JavaFX http://ptgmedia.pearsoncmg.com/images/9780321927767/samplepages/0321927761.pdf