**CALCULATOR POLINOAME**

**Nume prenume: Dulău Marius Cristian**

**Grupa: 30225**

**Profesor laborator: Asist. Antal Marcel**

Contents

[1. Cerințe funcționale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv principal 3](#_Toc476131447)

[2.2. Obective secundare 3](#_Toc476131448)

[3. Analiza problemei](#_Toc476131449) 4

[4. Proiectare](#_Toc476131450) 5

[4.1. Structuri de date](#_Toc476131451) 5

[4.2. Diagrama de clase](#_Toc476131452) 5

[4.3. Algoritmi](#_Toc476131453) 5

[5. Implementare](#_Toc476131454) 10

[6. Testare](#_Toc476131455) 13

[7. Concluzii si dezvoltări ulterioare](#_Toc476131456) 16

[8. Bibliografie](#_Toc476131457) 17

# Cerințe funcționale

Dezvoltați un calculator de polinoame care să permită următoarele operații:

* Citirea unui polinom de la tastatură sub forma 3X^4+2X^3-X+2
* Adunare
* Scădere
* Înmultire
* Împărțire
* Derivare
* Integrare

# Obiective

## Obiectiv Principal:

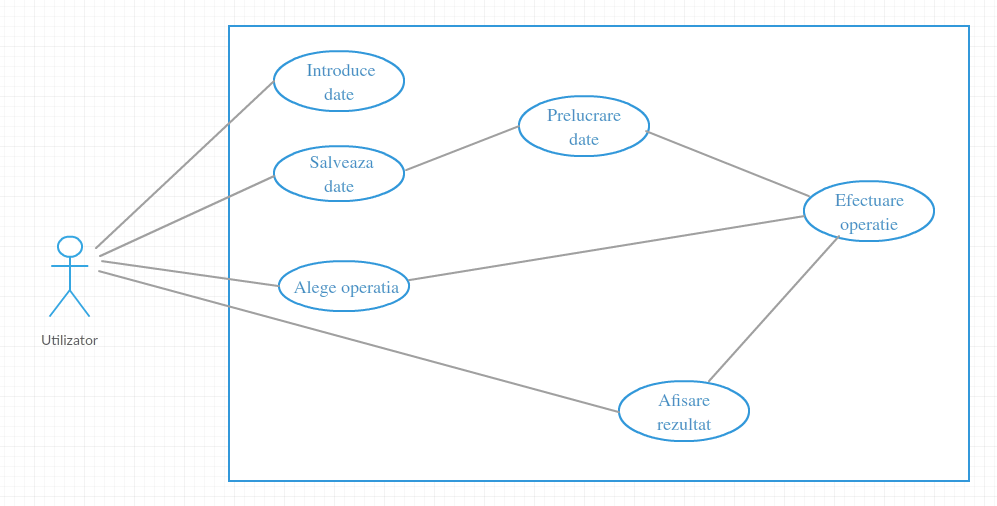
Propuneți, proiectați și implementați un sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi.

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri și scenarii | Descrierea în limbaj natural a scenariilor de utilizare și funcționare aplicației, însoțite de o diagramă. | 3 |
| Alegerea structurilor de date | Pentru a stoca monoamele unui polinom s-a folosit un ArrayList. | 4 |
| Împărțirea pe clase | Proiectul este împărțit în 5 clase: Logic, MyException, Monom, Polinom, GUI. | 5 |
| Dezvoltarea algoritmilor | S-au dezvoltat algoritmi pentru sortarea monoamelor în caz că utilizatorul introduce date într-o ordine oarecare, pentru returnarea informațiilor legate de monomul de ordin maxim. | 5 |
| Implementarea soluției | Implementare interfață grafică și a metodelor specifice operațiilor pe polinoame | 5 |
| Testare | Testarea efectuării operațiilor în interfața grafică și efectuarea de teste cu ajutorul Junit | 6 |

# Analiza Problemei

Utilizatorul introduce de la tastatură două polinoame sub forma 3X^4+2X^3-X+2. El va trebui să-și salveze datele în caz că dorește efectuarea de operații pe aceleași polinoame. În momentul în care își salvează datele programul va verifica dacă datele sale sunt introduse corect. Dacă sunt introduse greșit aplicația nu va stoca reprezentarea internă a polinoamelor și îl va avertiza pe utilizator asupra acestui lucru. Dacă datele au fost introduse corect utilizatorul poate alege una din cele 6 operații disponibile. La operațiile de scădere, împărțire, derivare și integrare poate să aleagă din două opțiuni. La scădere poate alege între f – g și g – f, la împărțire între f/g și g/f, iar la operațiile de derivare și integrare poate selecta polinomul căruia să-i aplice operatorul. Apoi, operația se realizează și se transmite rezultatul la utilizator prin intermediul interfeței grafice.

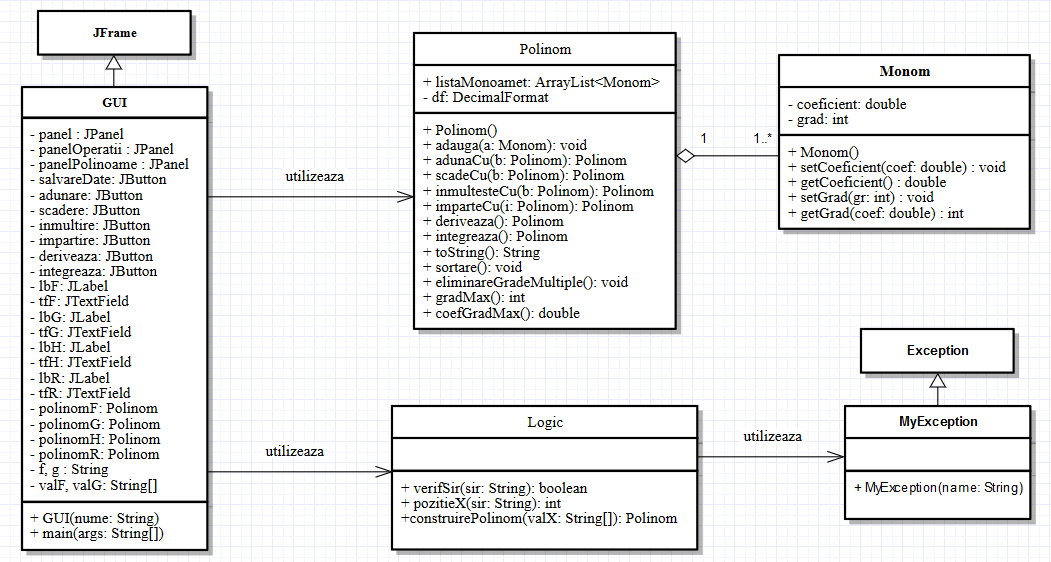


# Proiectare

## Structuri de date

Pentru a stoca monoamele unui polinom s-a folosit structura de date (colecția) ArrayList. Clasa ArrayList este o implementare a interfeței List. Această clasă se folosește atunci când se dorește să se acceseze elemente aflate la pozitții aleatoare în colecție, în schimb are dezavantajul că este mai înceată la inserarea și ștergerea de elemente la pozitții aflate în mijlocul colecției. Un ArrayList este mai util în utilizare decăt un vector, din punct de vedere al memoriei utilzate. În locul colecției ArrayList se poate utiliza colecția Set.

## Diagrama de clase



## Algoritmi

Pentru a realiza operațiile pe polinoame și pentru a le afișa în mod corect s-au implementat o suită de algoritmi.

Pentru a sorta lista de monoame din cadrul clasei **Polinom** s-a folosit următorul algoritm:

**public** **void** sortare()

{

Collections.*sort*(**this**.listaMonoame, **new** Comparator<Monom>(){

@Override

**public** **int** compare(Monom a, Monom b)

{

**if**(a.getGrad() == b.getGrad()) **return** 0;

**return** a.getGrad() > b.getGrad() ? -1:1;

}

});

}

Pentru a implementa operația de adunare și scădere a fost necesara parcurgerea polinoamelor f și g și stocarea monoamelor într-un polinom auxiliar c. Apoi s-a aplicat algoritmul de eliminarea a gradelor multiple din polinomul c.

**public** Polinom adunaCu(Polinom b)

{

Polinom a = **this**;

Polinom c = **new** Polinom();

a.listaMonoame.forEach(elem -> c.adauga(elem));

b.listaMonoame.forEach(elem -> c.adauga(elem));

c.sortare();

c.eliminareGradeMultiple();

**return** c;

}

**public** Polinom scadeCu(Polinom b)

{

Polinom a = **this**;

Polinom c = **new** Polinom();

a.listaMonoame.forEach(elem -> c.adauga(elem));

**for**(Monom i: b.listaMonoame)

{

Monom b2 = **new** Monom();

b2.setCoeficient((-1) \* i.getCoeficient());

b2.setGrad(i.getGrad());

c.adauga(b2);

}

c.sortare();

c.eliminareGradeMultiple();

**return** c;

}

Pentru a realiza operațiile de adunare și scădere în sine se folosește o metodă pentru a elimina gradele care se repetă și adunarea coeficienților la un singur grad de acel grad detectat.

**public** **void** eliminareGradeMultiple()

{

**for**(**int** i = 0 ; i < **this**.listaMonoame.size() - 1; i++)

{

Monom a = listaMonoame.get(i);

Monom b = listaMonoame.get(i + 1);

**if**(a.getGrad() == b.getGrad())

{

Monom aux = **new** Monom();

aux.setCoeficient(a.getCoeficient() + b.getCoeficient());

aux.setGrad(a.getGrad());

**if**(aux.getCoeficient() == 0.0f) listaMonoame.remove(a);

**else** listaMonoame.set(i, aux);

listaMonoame.remove(b);

}

}

}

Pentru implementarea operației de înmulțire se parcurge unul dintre polinoame, înmulțindu-se fiecare termen (monom) cu celălalt polinom, rezultatul fiind stocat în polinomul c care se returnează utilizatorului ca răspuns.

**public** Polinom inmultesteCu(Polinom b)

{

Polinom a = **this**;

Polinom c = **new** Polinom();

**for**(Monom i: a.listaMonoame)

{

**for**(Monom j: b.listaMonoame)

{

Monom aux = **new** Monom();

aux.setCoeficient(i.getCoeficient() \* j.getCoeficient());

aux.setGrad(i.getGrad() + j.getGrad());

c.adauga(aux);

}

}

c.sortare();

c.eliminareGradeMultiple();

**return** c;

}

Pentru a efectua operația de împărțire se utilizează următorul algoritm:

**while**(d.gradMax() >= i.gradMax())

{

Monom monomAux = **new** Monom();

monomAux.setCoeficient(d.coefGradMax() / i.coefGradMax());

monomAux.setGrad(d.gradMax() - i.gradMax());

c.adauga(monomAux);

Polinom polinomAux = **new** Polinom();

polinomAux.adauga(monomAux);

Polinom polinomAux2 = polinomAux.inmultesteCu(i);

d = d.scadeCu(polinomAux2);

d.sortare();

d.eliminareGradeMultiple();

}

//d = deîmpărțit

//i = împărțit

La derivare fiecare coeficient se înmulțește cu gradul monomului respectiv. Apoi gradul fiecărui monom se decrementează cu o unitate. La integrare gradul fiecărui monom se incrementează cu o unitate. Coeficientul fiecărui monom se împarte cu noua valoare a gradului respectiv.

**public** Polinom deriveaza()

{

Polinom c = **new** Polinom();

**for**(Monom i: **this**.listaMonoame)

{

Monom aux = **new** Monom();

aux.setCoeficient(i.getCoeficient() \* i.getGrad());

aux.setGrad(i.getGrad() - 1);

c.adauga(aux);

}

**return** c;

}

**public** Polinom integreaza()

{

Polinom c = **new** Polinom();

**for**(Monom i : **this**.listaMonoame)

{

Monom aux = **new** Monom();

aux.setGrad(i.getGrad() + 1);

aux.setCoeficient(i.getCoeficient() / (i.getGrad() + 1));

c.adauga(aux);

}

**return** c;

}

# Implementare

Proiectul conține 5 clase:

* Logic
* MyException
* Monom
* Polinom
* GUI

Utilizatorul introduce polinoamele de la tastatură. Interfața grafică preia datele sub formă de șiruri de caractere (String). Clasa **Logic** este destinată prelucrării șirului de caractere și returnează reprezentarea internă a polinomului în aplicație dat ca și dată de intrare. Această clasă conține 3 metode, dar nu are câmpuri. Cele 3 metode sunt:

* construirePolinom(String[] valX) **throws** MyException : cu ajutorul metodelor verifSir și pozitieX se construiește reprezenarea internă a polinomului.
* verifSir(String sir) **throws** MyException : funcția este utilizată pentru a verifica dacă un monom a fost introdus corect. În caz că se găsește o eroare în reprezentarea din șirul de caractere se aruncă o excepție care îl atenționează pe utilizator despre incorectitudinea datelor.
* pozitieX(String sir) : returnează poziția caracterului “X” în interiorul unui monom. Metoda(funcție) este utilă atunci când se prelucrează datele pentru a reprezenta un polinom în memoria internă.

Clasa **MyException** moștenește clasa **Exception** și este utilizată în clasa **Logic** pentru a arunca o excepție dacă utilizatorul a introdus date greșit de la tastatură.

Un polinom este format din unul sau mai multe monoame. Clasa **Monom** stochează informații despre fiecare monom din polinomul introdus de la tastatură. Clasa conține 2 câmpuri și 4 metode.

Câmpurile sunt:

* coeficient – reprezintă coeficientul monomului
* grad – reprezintă gradul monomului

Metodele aferente clasei sunt:

* setCoeficient(**double** coef) : metoda este folosită pentru a seta câmpul **coeficient** din clasă prin atribuirea la acesta a unei valori date ca parametru
* getCoeficient() : returnează valoarea câmpului **coeficient**
* setGrad(**int** gr) : metoda este folosită pentru a seta câmpul **grad** din clasă prin atribuirea la acesta a unei valori date ca parametru
* getGrad() : returnează valoarea câmpului **grad**

Clasa **Polinom** conține următoarele câmpuri:

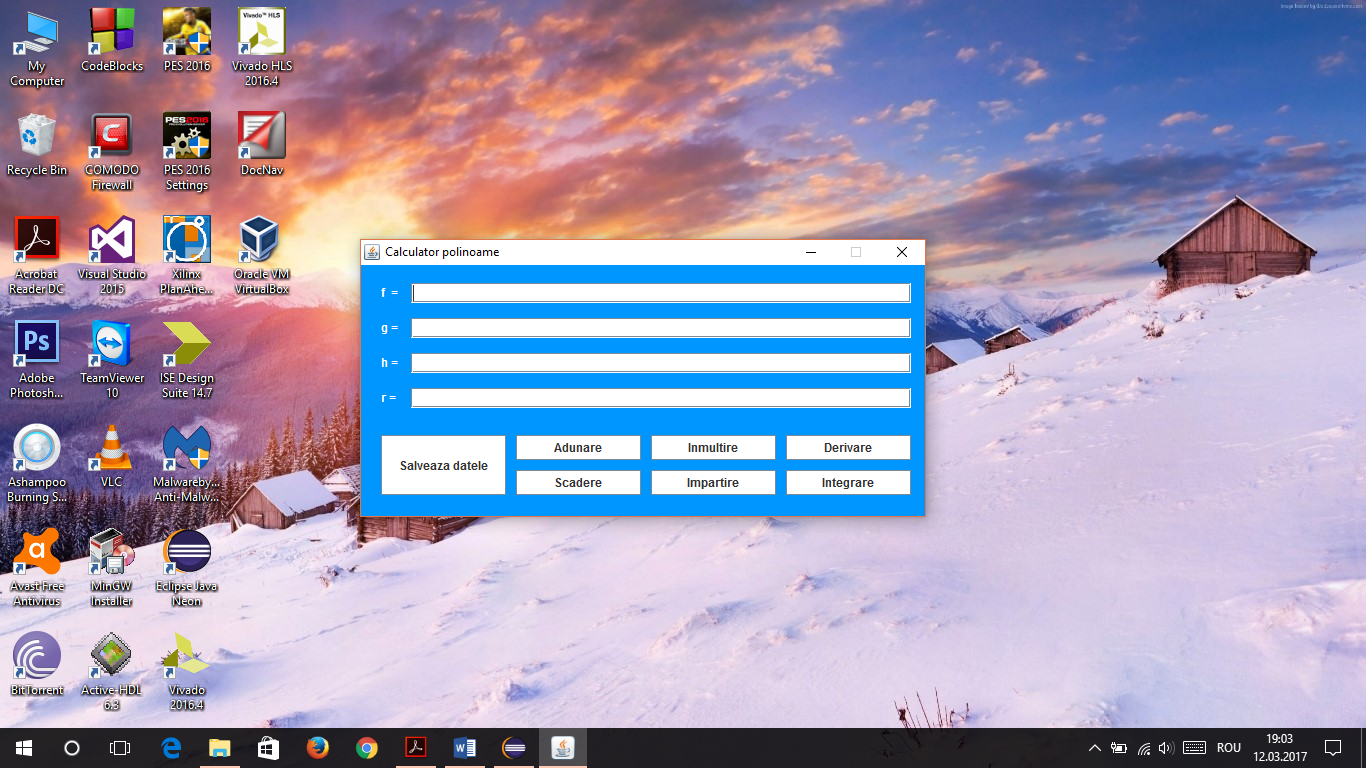
* listaMonoame : stochează monoamele din sirul de caractere dat de utilizator
* df (decimalFormat): acest câmp se folosește la construirea șirului de caractere a reprezentării polinomului. Dacă un coeficient are valoarea 2,000 se va scrie 2, nu 2,000.

Metodele acestei clase sunt:

* adauga(Monom a) : primește ca paramentru un obiect de tip **Monom** și îl introduce în listaMonoame
* adunaCu(Polinom b) : metoda primește ca parametru un obiect Polinom și îl adună la polinomul curent (cel din care se apelează metoda), returnând valoarea obținută. Se declară un nou polinom (cel care va fi returnat) care are propria sa listă de monoame. În acestă listă se introduc monoamele celor două polinoame. Cu ajutorul metodei eliminareGradeMultiple() se efectuează operația de adunare.
* scadeCu(Polinom b) : analog cu adunaCu(Polinom b) până la declararea unui polinom de returnat, inclusiv. În lista polinomului de returnat se introduc monoamele polinomului din care se apelează metoda. Monoamele polinomului dat ca paramentru vor fi introduse cu coeficientul negat.
* inmultesteCu(Polinom b) : fiecare monom din polinomul care apelează metoda este înmulțit cu fiecare monom din polinomul dat ca parametru. Analog cu gradele, doar că acestea se adună în loc să se înmulțească.
* imparteCu(Polinom i) : returnează câtul din împărțirea celor două polinoame.
* deriveaza() : returnează polinomul derivat. Coeficientul fiecărui monom este înmulțit cu gradul său. Apoi gradul monomului se scade cu o unitate.
* integreaza() : returnează polinomul integrat. Gradul fiecărui monom crește cu o unitate. Coeficientul monomului se împarte cu noua valoare a gradului.
* toString() : creează un șir de caractere ce conține reprezentarea internă a polinomului pentru a fi înțeles de utilizator
* sortare() : sortează listaMonoame după gradul monoamelor stocate
* eliminareGradeMultiple() : metoda
* gradMax() : returnează gradul maxim al polinomului
* coefGradMax() : returnează coeficientul aferent monomului de grad maxim

Ultimele două metode sunt folosite în procesul de împărțire a polinoamelor.

Interfața grafică a aplicației conține 7 butoane, 4 label-uri și 4 casete de text.



|  |  |
| --- | --- |
| Buton | Rezultatul unui click |
| Salveaza date | Preia șirurile de caractere ale celor două polinoame introduse (f și g) și creează reprezentarea internă a datelor introduse |
| Adunare | Realizează adunarea celor două polinoame |
| Scadere | Realizează scăderea celor două polinoame |
| Inmultire | Returnează rezultatul înmulțirii celor două polinoame |
| Impartire | Furnizează câtul și restul împărțirii |
| Derivare | Derivează unul din polinoamele introduse. Utilizatorul trebuie să aleagă polinomul de derivat. |
| Integrare | Integrează unul din polinoamele introduse. Utilizatorul trebuie să aleagă polinomul de integrat. |

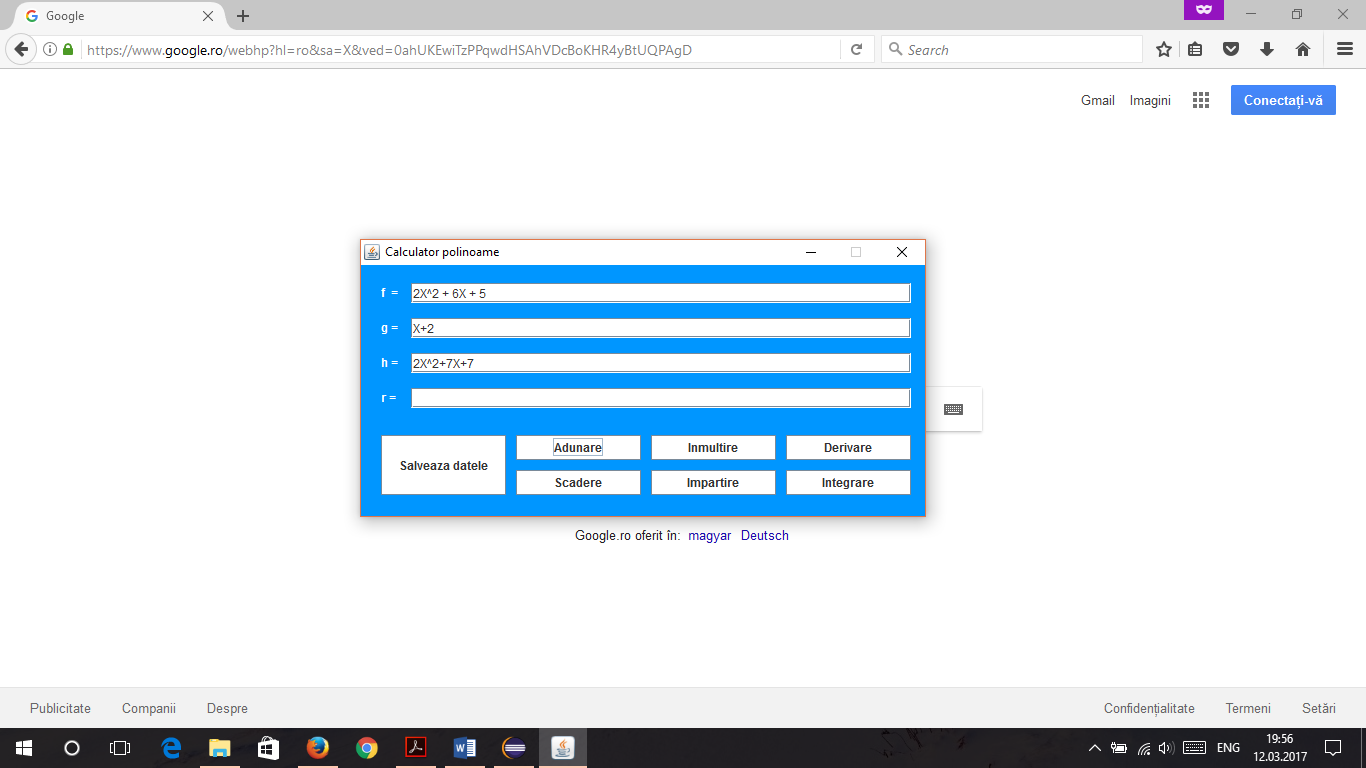
# Testare

Considerăm următoarele două polinoame:

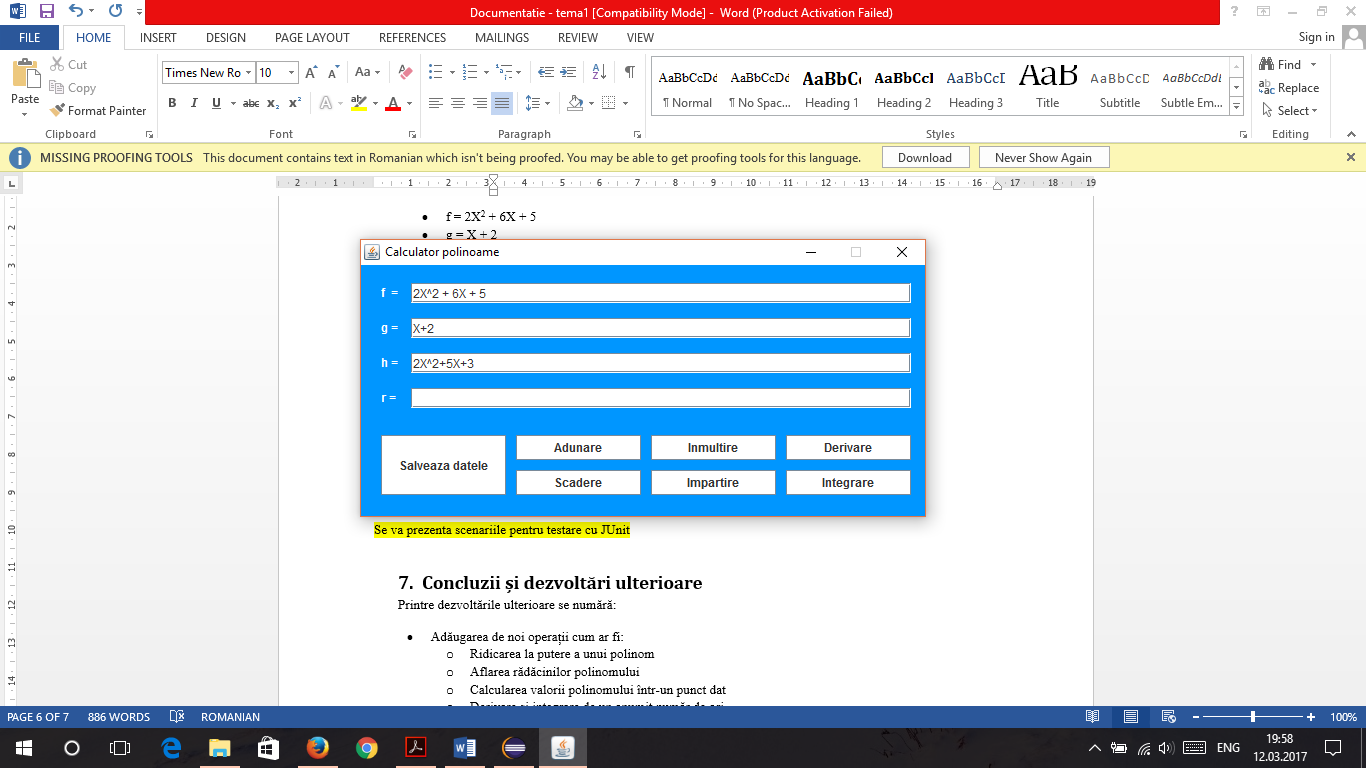
* f = 2X2 + 6X + 5
* g = X + 2

Se vor aplica toate operațiile disponibile în interfața grafică.

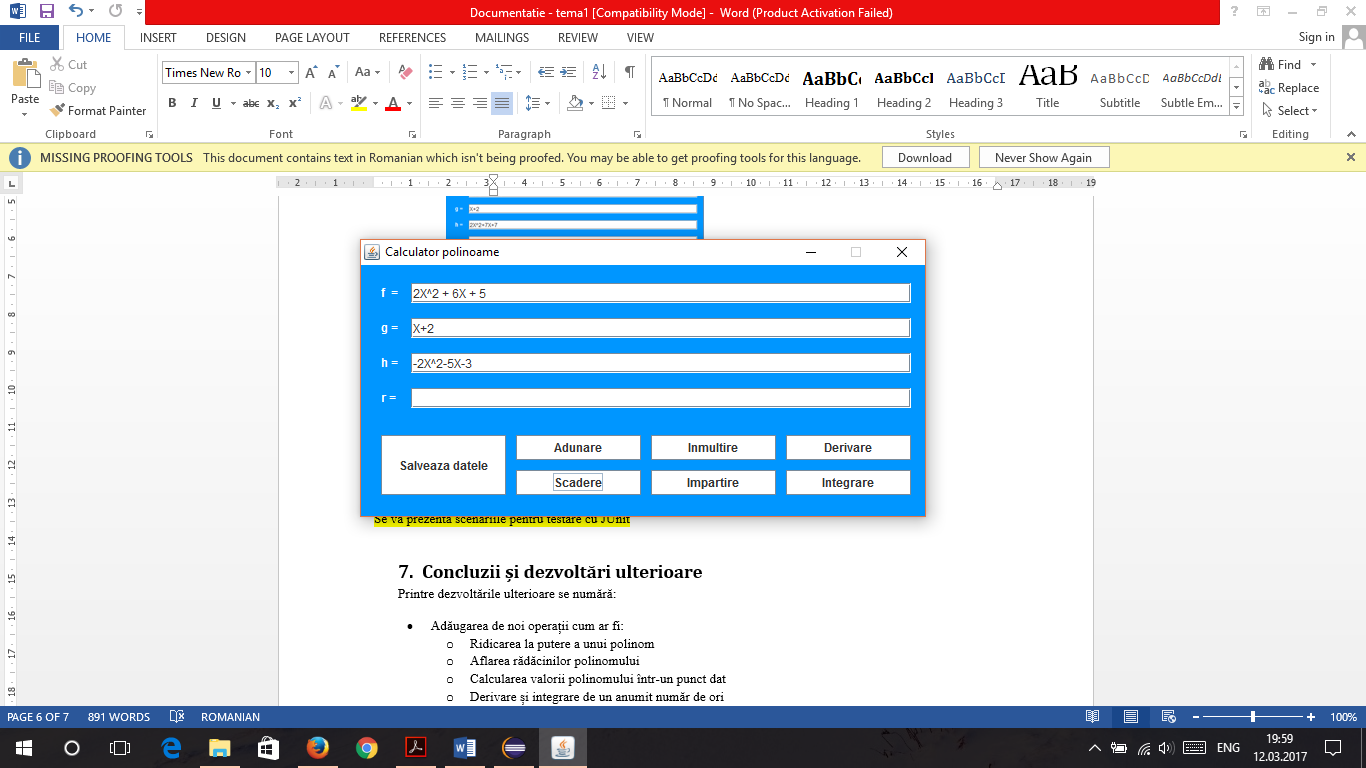
* adunare : h = f + g = 2X2 + 7X + 7



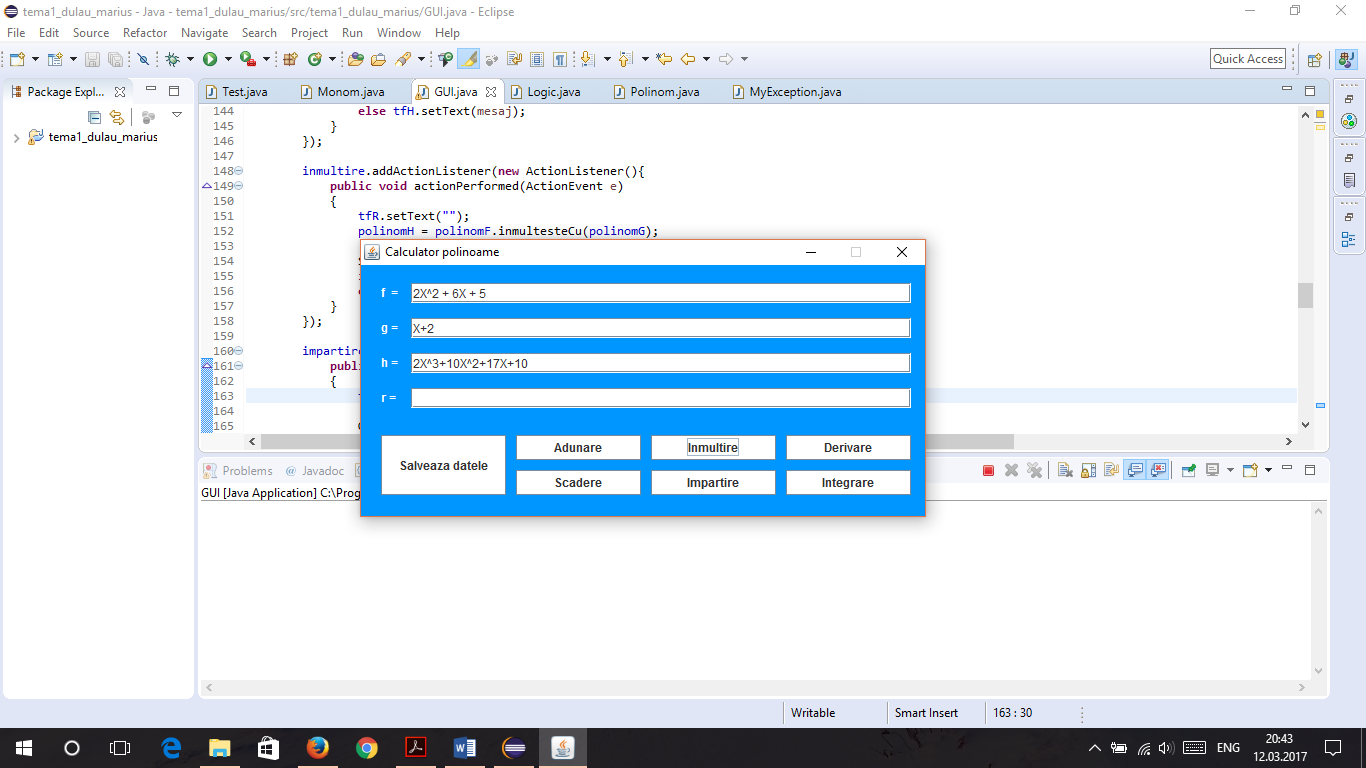
* scădere :
  + h = f – g = 2X2 + 5X + 3



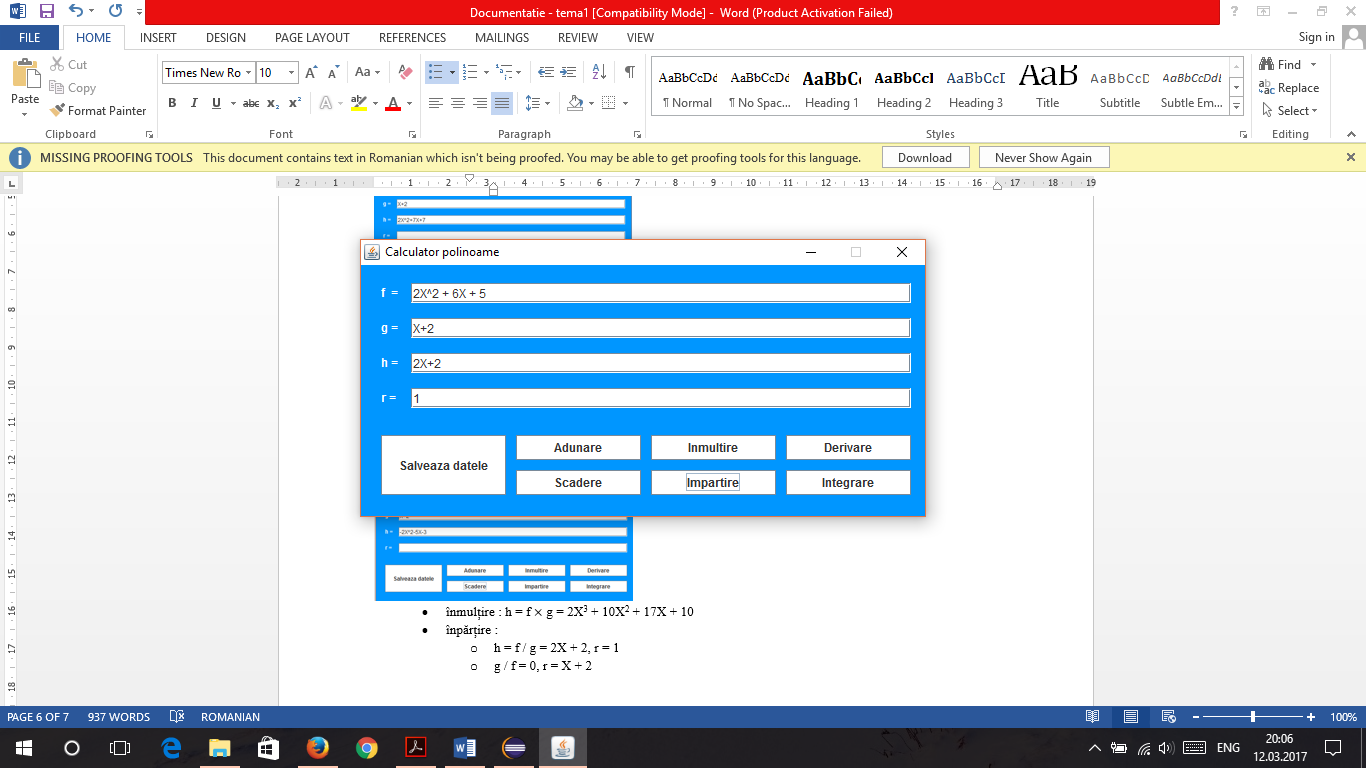
* + h = g – f = -2X2 - 5X – 3



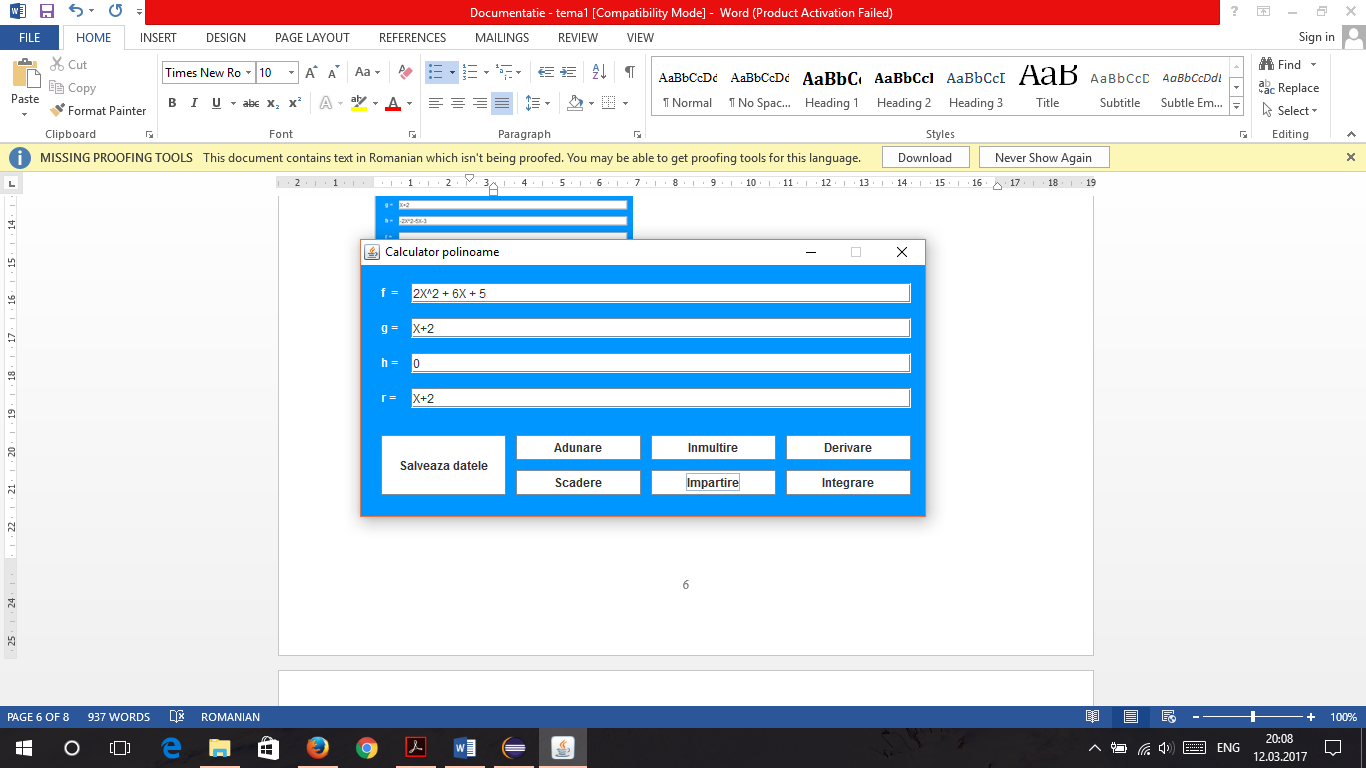
* înmulțire : h = f g = 2X3 + 10X2 + 17X + 10



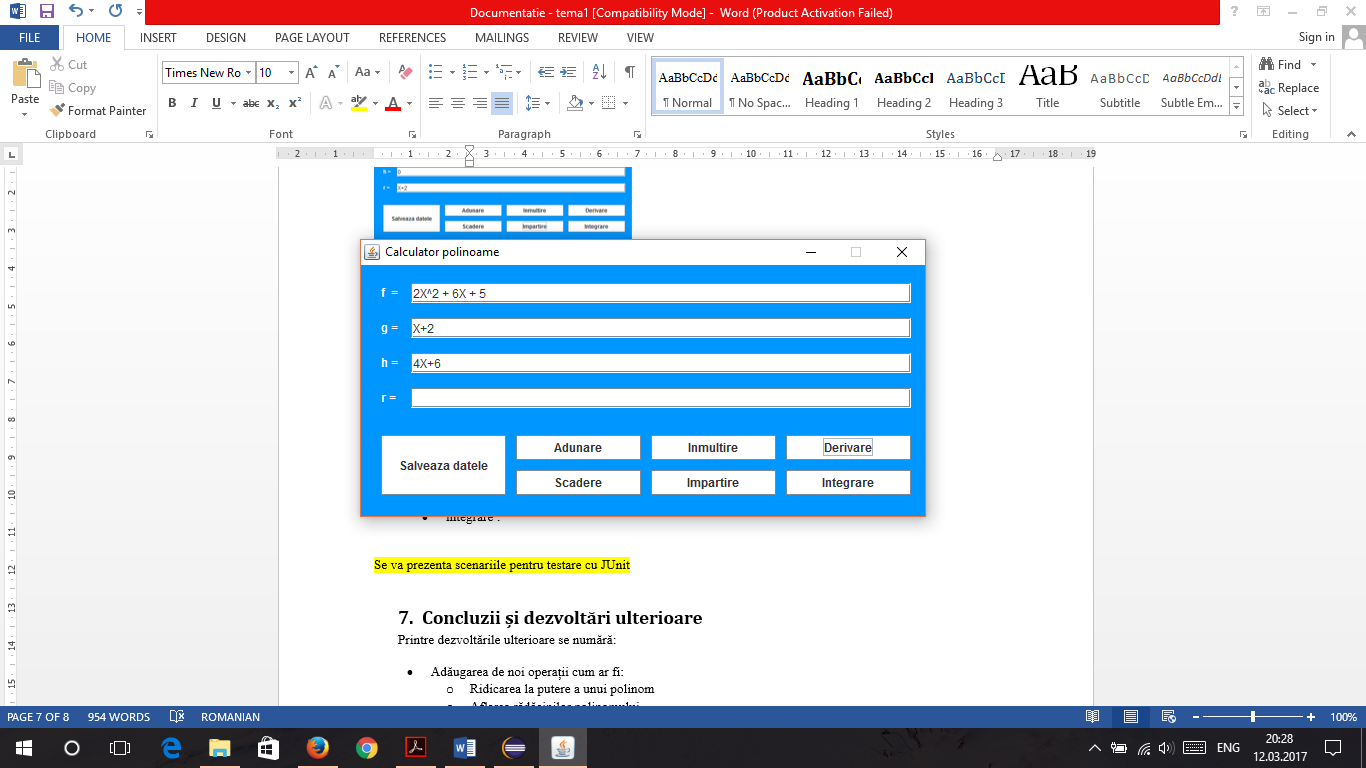
* înpărțire :
  + h = f / g = 2X + 2, r = 1



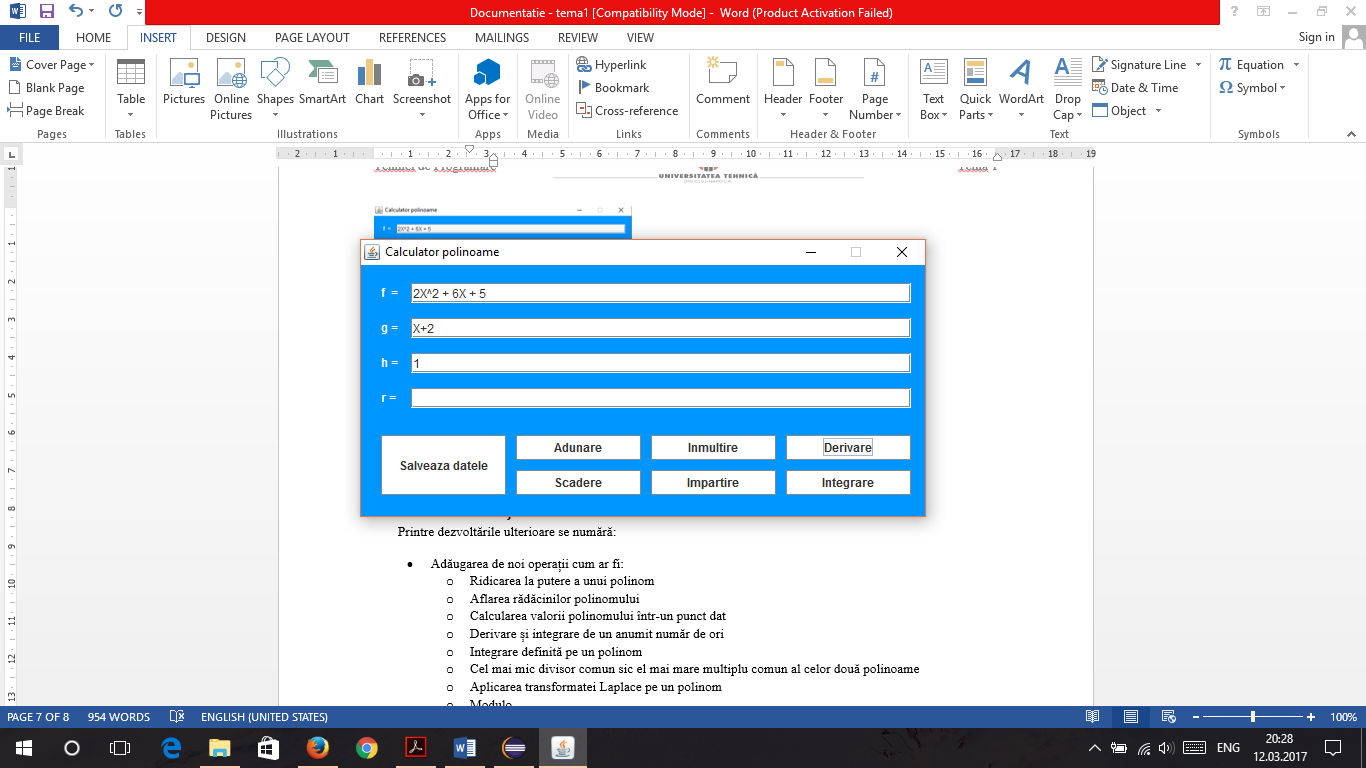
* + h = g / f = 0, r = X + 2



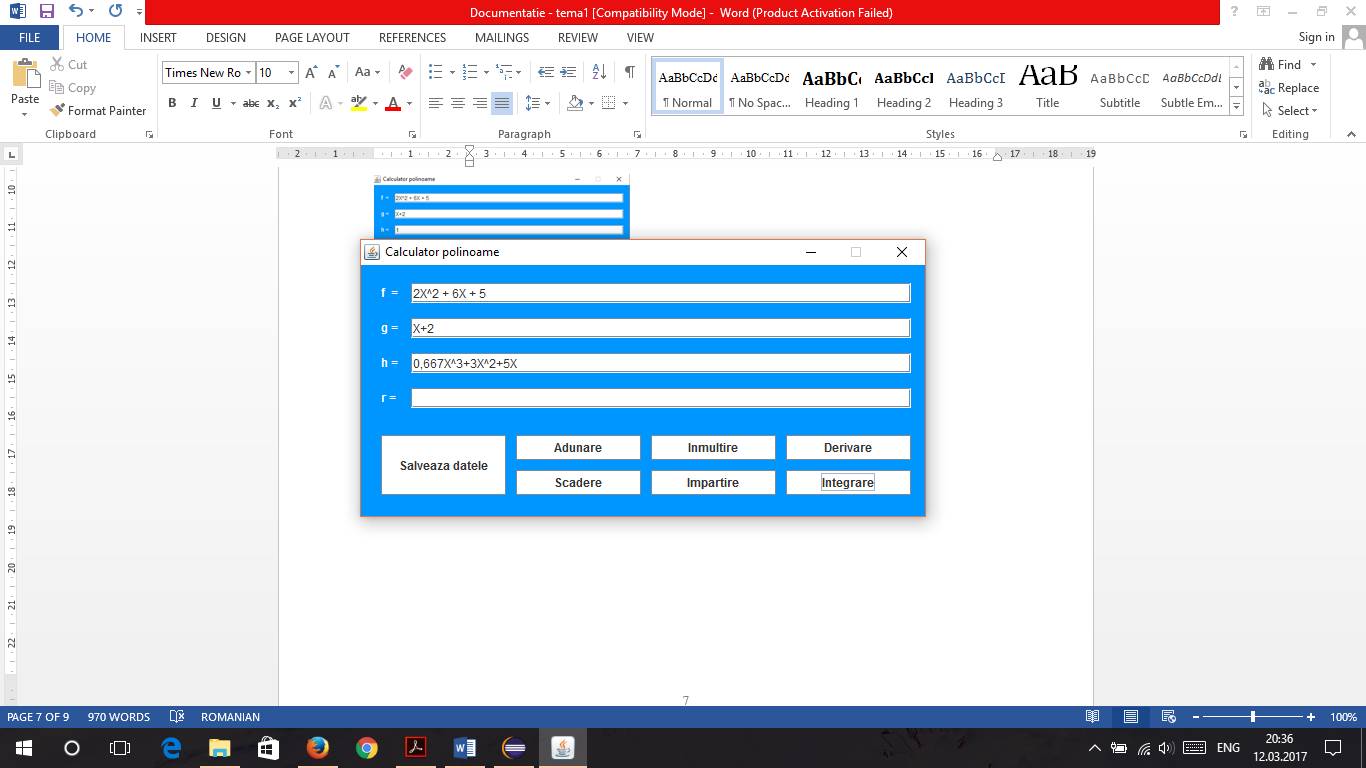
* derivare :
  + h = f ' = 4X+6



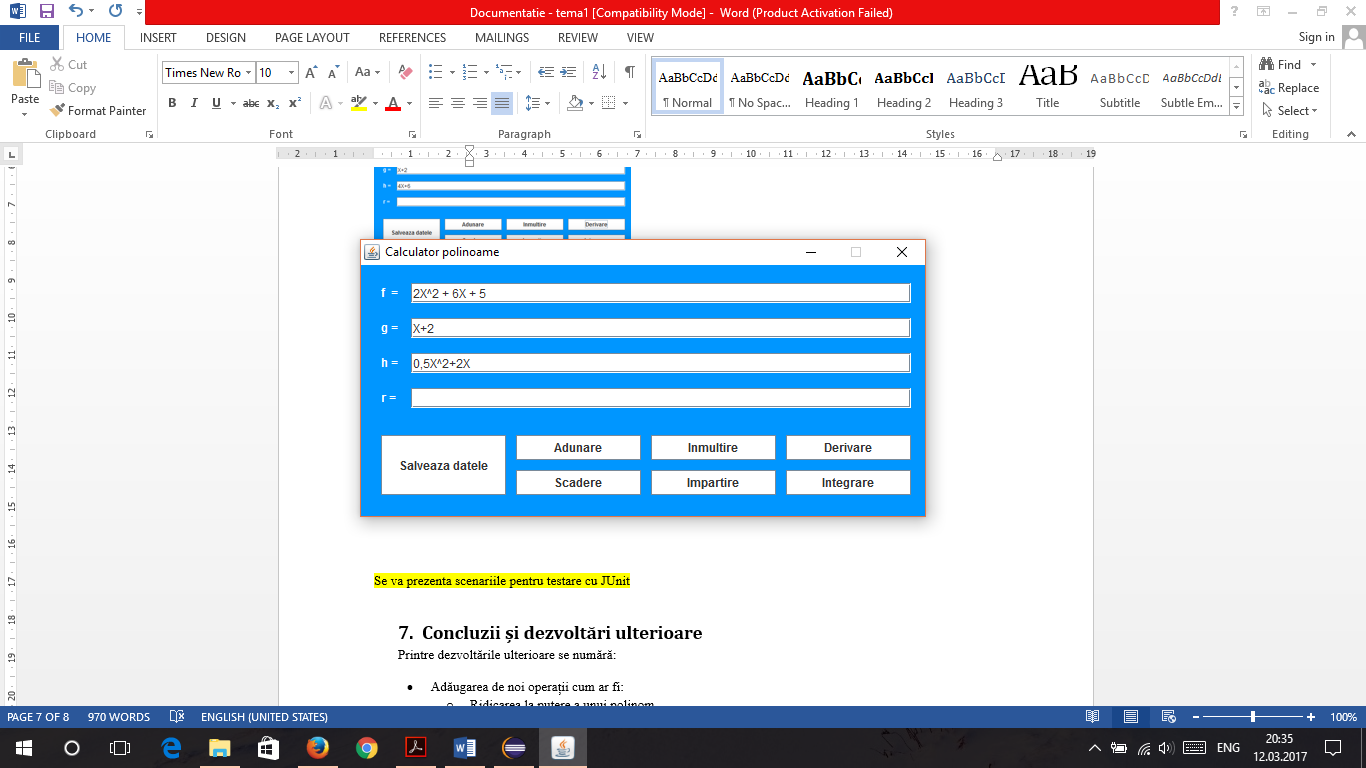
* + h = g' = 1



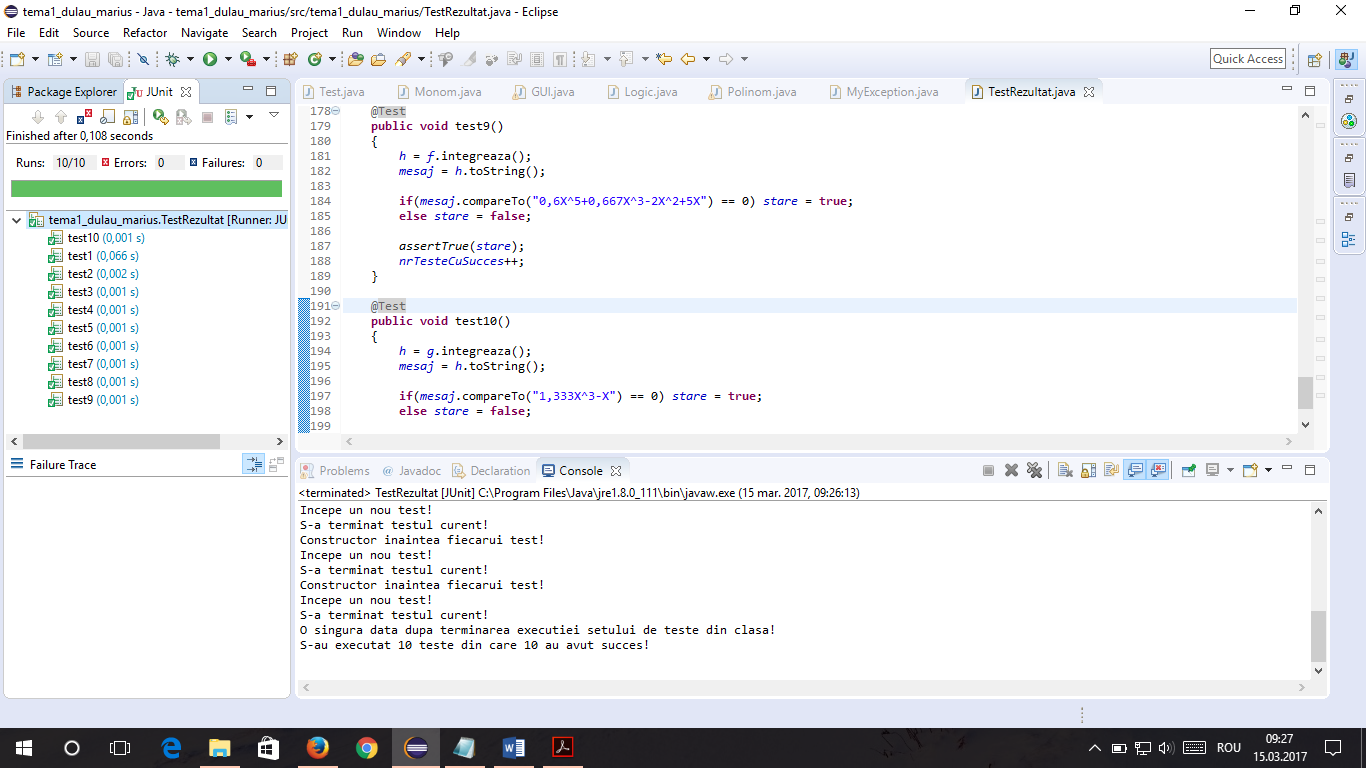
* integrare :
  + h = = 0,667X3 + 3X2 + 5X



* + h = = 0,5X2 + 2X



Pentru testare cu Junit s-a folosit o clasă de test. În interiorul acestei clase s-au declarat variabile necesare pentru a efectua operațiile cu polinoame. S-au efectuat teste pentru fiecare operație în parte. Rezulatele returnate au fost pozitive, din punct de vedere al corectitudinii rezultatului.



Rezultatele obținute atât în urma efectuării testelor din clasa anterioară, cât și rezultatele obținute în urma testării interfeței grafice (prin rularea clasei GUI) s-au dovedit a fi corecte din punct de vedere matematic.

# Concluzii și dezvoltări ulterioare

În urma dezvoltării acestui proiect am învățat să realizez o interfață grafică din cod (fără drag and drop), să poziționez obiecte în fereastra grafică după coordonate, să utilizez o clasă în interiorul alteia, să lucrez cu datele dintr-un ArrayList, să scriu numere flotante care sunt întregi fără virgulă.

Printre dezvoltările ulterioare se numără:

* Adăugarea de noi operații cum ar fi:
  + Ridicarea la putere a unui polinom
  + Aflarea rădăcinilor polinomului
  + Calcularea valorii polinomului într-un punct dat
  + Derivare și integrare de un anumit număr de ori
  + Integrare definită pe un polinom
  + Cel mai mic divisor comun sic el mai mare multiplu comun al celor două polinoame
  + Aplicarea transformatei Laplace pe un polinom
  + Modulo
* Reprezentarea grafică a polinoamelor
* Reprezentarea prin animații a operației ce se efectuează
* Dezvoltarea aplicației la introducerea de polinoame cu mai multe variabile
* Scrierea rezultatului sub forma 3X4 + 2X3 + 7X2 - X + 2
* Introducere de polinoame cu coeficienți reali sau complecși
* Transformarea aplicației într-un soft educațional pentru elevii de liceu

# Bibliografie

<http://stackoverflow.com/questions/4066538/sort-an-arraylist-based-on-an-object-field>

<http://stackoverflow.com/questions/18581531/in-java-how-can-i-determine-if-a-char-array-contains-a-particular-character>

<https://docs.oracle.com/javase/tutorial/uiswing/layout/none.html>

<http://stackoverflow.com/questions/18031704/jframe-how-to-disable-window-resizing>

<http://tutiez.com/how-to-show-dialogbox-in-java-complete-joptionpane-tutorial/>

<http://stackoverflow.com/questions/1037139/loop-through-jpanel>

<http://stackoverflow.com/questions/7359189/how-to-change-the-mouse-cursor-in-java>

<http://www.colorspire.com/rgb-color-wheel/>