**Tehnici de programare fundamentale**

**Tema 1**

**Operatii cu polinoame**

Pentek Tamas

Grupa: 30228

**1. Obiectivul temei**

Obiectul principal al temei este crearea unui calculator care lucreaza cu polinoame facand operatiile urmatoare: adunare, scadere, inmultire, impartire, derivare si integrare. Polinoamele sunt de o singura variabila si cu coeficienti intregi. Acest calculator trebuie sa aiba o interfata grafica, prin care utilizatorul trebuie sa introduca polinomul, iar dupa acesta poate sa vada rezultatul.

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

Utilizatorul poate sa aleaga dintre 6 operatii. Utilizatorul poate sa obtine rezultatul mar multor operatii, nu trebuie sa inchida si din nou sa deschida programul, cu o singura pereche de polinoame introdusa poate sa faca toate operatiile. Langa aceasta utilizatorul are posibilitatea de a reintroduce alte doua polinoame cu care din noua poate sa faca orice operatie. Utilizatorul este ajutat prin mai multe guidelineuri, de exemplu: daca introduce gresit polinomul si apasa Enter automat apare o noua fereastra in care programul scrie ce a gresit si unde a gresit. Un alt feedback pentru utilizator este in momentul introducerii polinomului: daca polinomul a fost introdus corect casuta de text va schimba culoarea in verde confirmand un polinom introdus corect.

In ceea ce urmeaza, sunt prezentate toate operatiile ca o diagrama use-case si prin descriere use-case sub forma unei liste, pe care daca utilizatorul urmareste va obtine rezultatul dorit.

**Use Case: Adunare**

**Primary Actor: utilizator**

**Main Success Scenario:**

1. Utilizatorul deschide aplicatia
2. Utilizatorul introduce primul polinom in prima casuta de text si apasa Enter
3. Programul verifica daca polinomul a fost introdus corect
4. Casuta de text va schimba culoarea in verde, confirmand ca polinomul a fost introdus corect
5. Utilizatorul introduce al doilea polinom in a doua casuta de text si apasa Enter
6. Programul verifica daca polinomul a fost introdus corect
7. Casuta de text va schimba culoarea in verde, confirmand ca polinomul a fost introdus corect
8. Utilizatorul selecteaza operatia de „Adunare”
9. Utilizatorul apasa butonul „OK”
10. Programul face operatia si afiseaza in casuta de text rezultatul operatiei afisand polinomul rezultat

**Alternative Sequences:**

a.) Utilizatorul introduce gresit primul polinom si apasa Enter

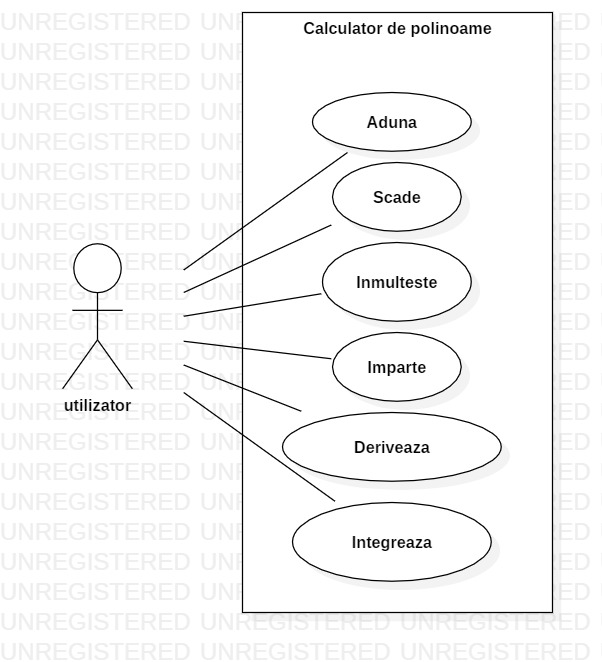
1. Programul afiseaza intr-o fereastra noua un mesaj de eroare
2. Utilizatorul apasa butonul „OK”
3. Utilizatorul continua cu punctul 2. din Main Success Scenario

b.) Utilizatorul selecteaza o alta operatie din greseala

1. Utilizatorul apasa operatia dorita
2. Utilizatorul continua cu punctul 9. din Main Success Scenario

c.) Utilizatorul introduce polinomul, dar nu apasa Enter si rezultatul este 0

1. Utilizatorul face click in prima casuta de text
2. Utilizatorul apasa Enter
3. Utilizatorul face click in a doua casuta de text
4. Utilizatorul apasa Enter
5. Utilizatorul continua cu punctul 8 din Main Success Scenario

****

**Use Case: Derivare**

**Primary Actor: utilizator**

**Main Success Scenario:**

1. Utilizatorul deschide aplicatia
2. Utilizatorul introduce primul polinom in prima casuta de text si apasa Enter
3. Programul verifica daca polinomul a fost introdus corect
4. Casuta de text va schimba culoarea in verde, confirmand ca polinomul a fost introdus corect
5. Utilizatorul introduce al doilea polinom in a doua casuta de text si apasa Enter
6. Programul verifica daca polinomul a fost introdus corect
7. Casuta de text va schimba culoarea in verde, confirmand ca polinomul a fost introdus corect
8. Utilizatorul selecteaza operatia de „Derivare”
9. Apare un meniu unde alege „Polinom 1” daca doreste derivarea primului polinom si alege „Polinom 2” daca doreste derivarea al doilea polinom
10. Utilizatorul apasa butonul „OK”
11. Programul face operatia si afiseaza in casuta de text rezultatul operatiei afisand polinomul rezultat

**Alternative Sequences:**

a.) Utilizatorul introduce gresit primul polinom si apasa Enter

1. Programul afiseaza intr-o fereastra noua un mesaj de eroare
2. Utilizatorul apasa butonul „OK”
3. Utilizatorul continua cu punctul 2. din Main Success Scenario

b.) Utilizatorul selecteaza o alta operatie din greseala

1. Utilizatorul apasa operatia dorita
2. Utilizatorul continua cu punctul 9. din Main Success Scenario

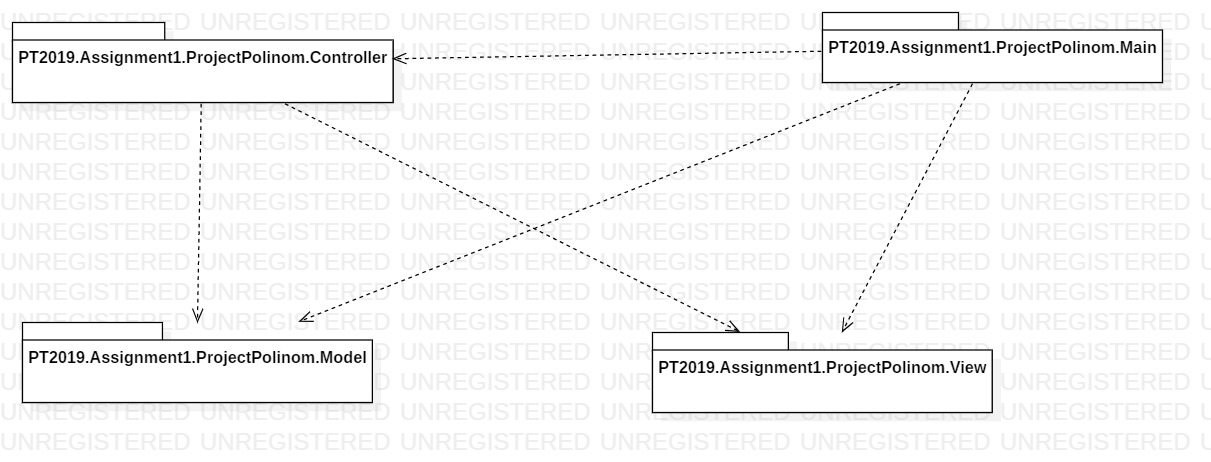
c.) Utilizatorul introduce polinomul, dar nu apasa Enter si rezultatul este 0

1. Utilizatorul face click in prima casuta de text
2. Utilizatorul apasa Enter
3. Utilizatorul face click in a doua casuta de text
4. Utilizatorul apasa Enter
5. Utilizatorul continua cu punctul 8 din Main Success Scenario

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

Acest proiect este impartit in 4 pachete. Pachetele sunt facute in asa fel incat sa respecte proiectarea MVC (Model-View- Controller). Primul pachet este PT2019.Assignment1.ProjectPolinom.Controller, care contine o clasa numita GeneralController, aceasta reprezinta Controllerul, aici este legat Model cu View.

Al doilea pachet este PT2019.Assignment1.ProjectPolinom.View in care avem o singura clasa GeneralView care reprezinta View, adica in aceasta clasa este descrisa, este contruita interfata grafica cu utilizatorul. Al treilea pachet este PT2019.Assignment1.ProjectPolinom.Model in care sunt descrise algoritmii de calcul a fiecarei operatie. Acest pachet contine doua clase: Monom si Polinom, un polinom fiind alcatuit din unu sau mai multe monoame. Ultimul pachet este PT2019.Assignment1.ProjectPolinom.Main, acest pachet contine o singura clasa, MainClass in care avem descrisa functia main, de aici pornim aplicatia. Diagrama UML de pachete arata in felul urmator:



Din figura rezulta faptul ca pachetul Main depinde de toate celelalte pachete, ceea ce este absolut normal, fiindca fara Model, View si Controller nu putem rula aplicatia, daca rulam fara un pachet nu vom avea rezultatul dorit. Un alt lucru ce rezuta de pe figura este faptul ca Controllerul are dependente fata de Model si View, fara aceste dependente Controllerul ar pierde rolul: sa lege Modelul cu View si prin acesta rezultand proiectul final cu interfata grafica, calculul operatiilor si legatura dintre aceste doua componente.

Clasele respecta proiectarea OOP, prin structura si prin componente, variabile si operatii folosite.

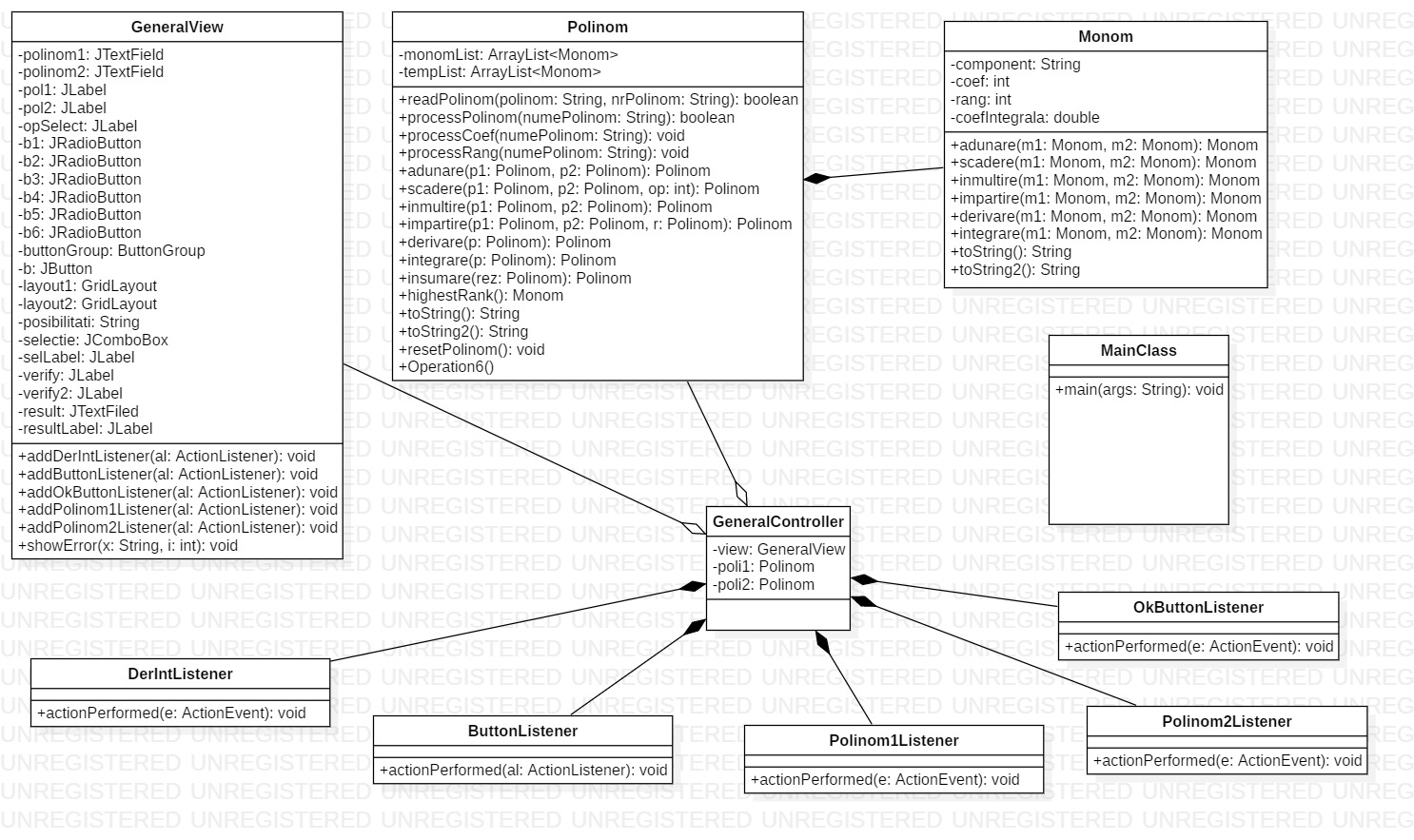
Clasa Monom are ca si variabile instante 4 componente. Aceste 4 variabile caracterizeaza un monom: „coef” reprezinta coeficientul monomului, „rang” reprezinta rangul monomului, „component” reprezinta monomul in format text, adica sub forma de String, iar „coefIntegrala” este tot coeficientul monomului, acest coeficient este folosit la operatia de integrare unde poate sa apara fractii si de aceea trebuie sa folosim reprezentarea double ca sa obtinem rezultatul corect. Langa variabile gasim mai multe metode in aceasta clasa si anume: adunare, scadere, inmultire, impartire, derivare, integrare. Toate aceste metode primesc 2 argumente de Monom, iar rezultatul este returnat ca si un Monom. Toate aceste metode sunt statice, prin acesta putem sa apelam direct cu numele clasei, fara obiecte suplimentare. Mai avem cateva metode getter si setter, pentru a accesa din afara clasei variabilele de instanta care au vizibilitate private. O alta metoda important este toString() care ajuta la afisarea corecta a rezultatului pentru utilizator. Pentru toate operatiile este folosit metoda toString(), cu exceptia operatiei integrare cand folosimtoString2().

Clasa Polinom are o structura foarte asemanatoare cu clasa Monom. Metodele sunt la fel construite, in aceste metode apelam metodele din clasa Monom pentru a face operatia intre fiecare monom care construieste polinomul. Clasa Polinom mai are 4 metode esentiale, ce nu gasim in clasa Monom: readPolinom, processPolinom, processCoef si processRang. Metoda readPolinom este folosita pentru citirea polinomului, iar urmatoarele trei metode au rol esential in procesarea polinoamelor, mai ales in gasirea coeficientilor si a rangului fiecarui monom care se afla in lista de monoame, adica monoame care construiesc polinomul. Ca variabile instante clasa Polinom are doua ArrayList<Monom> numit monomList si tempList. in aceste liste sunt salvate componentele fiecarui polinom, adica monoamele. Variabila tempList este folosit doar la inceput, adica la citrea polinomului si procesarea polinomului, dupa aceea rezultatele si componentele esentiale se gasesc in lista monomList.

Clasa GeneralView contine elementele interfetei grafice, de exemplu: JTextField, JLabel, JButton, JComboBox, JRadioButton etc. In constructorul clasei este construit interfata grafica, clasa fiind practic un JFrame. Fiecare componenta este pusa intr-un JPanel, prin acesta putem sa aliniam mai usor componentele. Ca si metode apre niste gettere si settere si niste metode foarte importante in care sunt setate ActionListeners pentru fiecare componenta grafica.

Clasa GeneralController face legatura intre clasa Polinom si clasa GeneralView. Are ca si variabile instante un view de tip GeneralView si doua polinoame de tip Polinom, variabile cu care se face operatiile. In aceasta clasa avem niste clase interioare care reprezinta niste ActionListeners pentru buton, pentru text field etc. In aceste clase sunt folosite atat elemente din clasa View cat si din clasa Polinom. De exemplu: pentru a face adunarea avem un Listener care verifica daca butonul Adunare a fost selectat es a fost apasat butonul OK, si daca da atunci se executa metoda adunare din clasa Polinom, iar rezultatul va fi scris in JTextField numit rezultat. In asa fel utilizatorul obtine rezultatul foarte usor.

Diagrama UML de clase este prezentat pe figura urmatoarea, pe care apar toate varibile instante si toate metodele fiecarei clase si relatia intre ele.



**4. Implementare**

Clasa Polinom este o clasa foarte importanta deoarece aici se face citirea, procesarea si prelucrarea polinomului. Citirea polinomului este implementat in functia readPolinom.

public boolean readPolinom(String polinom, String nrPolinom) {

**public** **boolean** readPolinom(String polinom, String nrPolinom) {

**boolean** e = **false**;

String pattern = "([+-]\*\\d\*[x]\*\\^\*\\d\*)\*";

String pattern2 = "[+-]\*\\d\*[x]\*\\^\*\\d\*";

Pattern p = Pattern.*compile*(pattern2);

Matcher m = p.matcher(polinom);

**if** (Pattern.*matches*(pattern, polinom) && !(polinom.equals(""))) {

**while** (m.find() && !(m.group(0).equals(""))) {

Monom x = **new** Monom();

**if** (m.group(0).endsWith("^")) {

GeneralView.*showError*(nrPolinom, 0);

**return** e;

}

x.setComponent(m.group(0));

**this**.tempList.add(x);

e = **true**;

}

} **else** {

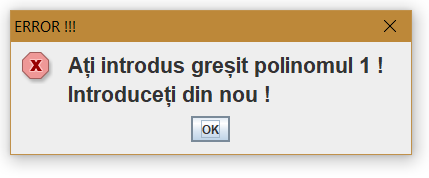
GeneralView.*showError*(nrPolinom, 0);

}

**return** e;

}

In aceasta metoda este folosit Regexul. Stringul pattern contine expresia prin care Matcherul gaseste monomul. Daca expresia introdusa de utilizator este corecta atunci monoamele care alcatuiesc polinomul vor fi salvate in tempList al polinomului. Tot in aceasta metoda este verificata situatia cand apare un input de ex: „x^4+4x^3+x^”. In aceasta situatie va aparea o fereastra noua cu un mesaj de error, ca si in figura urmatoare:



Daca apare orice alt tip de eroare, adica Matcherul nu recunoaste textul introdus va aparea tot o fereastra cu acelasi mesaj de error. Metoda returneaza un boolean, daca citirea s-a terminat cu succes va returna true, altfel false.

Operatiile sunt implementate aproape la fel, aici vom discuta despre functia adunare.

**public** **static** Polinom adunare(Polinom p1, Polinom p2) {

Polinom rez = **new** Polinom();

**for** (Monom m1 : p1.monomList) {

**for** (Monom m2 : p2.monomList) {

**if** (m1.getRang() == m2.getRang())

rez.monomList.add(Monom.*adunare*(m1, m2));

}

}

**return** rez;

}

Metoda adunare primeste 2 argumente de tip Polinom, adica operanzii si creeaza un al treilea Polinom care va returna ca si rezultat. Pentru fiecare monom din lista de monom ai polinoamelor p1 si p2 se va face adunarea intre doua monoame doar daca au acelasi rang, rangul obtinem prin getterul getRang(). Rezultatul adunarii se salveaza in monomList si la finalul metodei se returneaza. Acest polinom returnat va ajunge in casuta de text, bineinteles sub forma unui String.

O alta metoda importanta este metoda insumare() care are rolul de a insuma monoamele de acelasi rang dupa inmultirea polinoamelor.

**public** Polinom insumare(Polinom rez);

Langa calcularea adunarii, inmultirii etc. este foarte important clasa Polinom1Listener care are rolul de a face primul pas dupa ce a introdus utilizatorul primul polinom.

**class** Polinom1Listener **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

String polinom1 = view.getPolinom1().getText();

poli1.resetPolinom();

**boolean** read = poli1.readPolinom(polinom1, "polinomul 1");

**boolean** rez = poli1.processPolinom("polinomul 1");

**if** (rez == **true** && read == **true**)

view.getPolinom1().setBackground(Color.***GREEN***);

**else**

view.getPolinom1().setBackground(Color.***RED***);

}

}

Ca si primul pas se citeste Stringul introdus de utilizator intr-o variabila de String. Dupa aceea se face resetarea pilonomului, acest pas este foarte important atunci cand se introduce un alt polinom dupa polinomul curent. Pentru a lucra cu valori corecte si actuale este nevoie resetarea polinomului, adica stergerea valorilor din polinom. Urmeaza citirea propriu-zisa prin metoda readPolinom si dupa aceea procesarea polinomului, adica salvarea coeficientilor si a rangului in list de polinoame. Putem observa ca si metoda readPolinom si metoda processPolinom returneaza un boolean. Aceste variabile folosim pentru a asigur pe utilizator ca a introdus polinomul corect sau nu. Daca ambele variabile vor fi true, inseamna ca am avut in input corect si textfieldul in care a introdus polinomul corect se va schimba in verde. Daca a aparut ceva eroare, adica utilizator a introdus gresit datele, culoare a se va schimba in rosu, semnaland ca ceva este gresit.

O metoda importanta din clasa GeneralView este metoda in care adaugam butoanelor de adunare, scadere, inmultire si impartire cate un ActionListener. Adaugam pentru toate cele 4 butoane, pentru ca vrem ca daca utilizatorul selecteaza unul dintre aceste butoana sa nu apara meniul de selectie unde se poate alege care polinom vrea sa deriveze sau sa integreze. Acest meniu trebuie sa apara doar atunci cand utilizatorul alege butonul de Derivare sau Integrare. De aceea in cazul acestor 4 butoane trebuie sa inactivam meniul, adica sa nu fie vizibil, numai la momentul potrivit.

**public** **void** addButtonsListener(ActionListener al) {

b1.addActionListener(al);

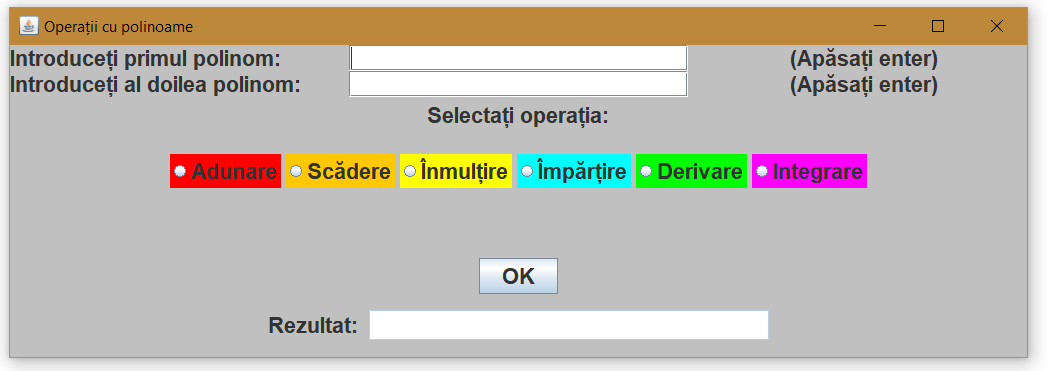
b2.addActionListener(al);

b3.addActionListener(al);

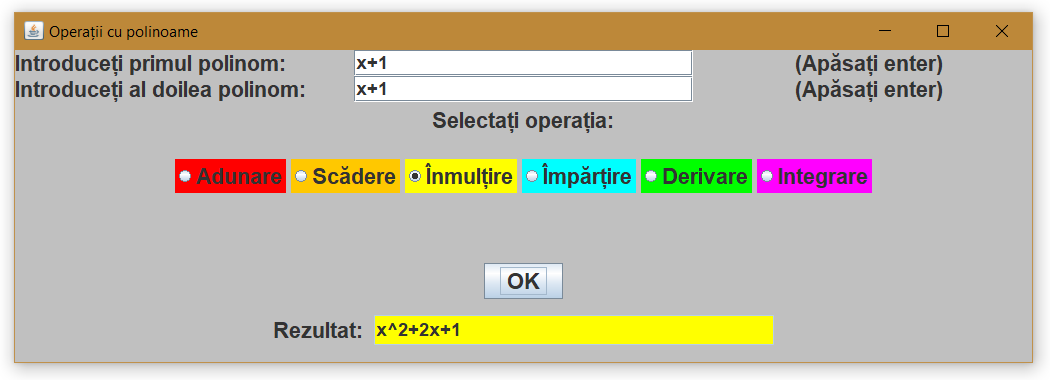
b4.addActionListener(al);

}

Interfata grafica arata in felul urmator:



Butoanele sunt afisate cu culori diferite ca sa fie mai usor pentru utilizator gasirea butonului corespunzator. Langa acesta si textfieldul va avea aceeasi culoare dupa efectuarea operatiei. De exemplu: daca introducem doua polinoame si vrem sa facem adunare intre ele, dupa alegerea butonului „Inmultire” si apasarea butonului „OK” rezultatul va fi afisat si textfieldul va avea culoarea galbena.



**5. Rezultate**

Pentru a obtine rezultate corecte utilizatorul trebuie sa introduca polinoamele intr-o forma corecta, pe care Regexul poate procesa. Aceasta varianta de calculator de polinoame recunoaste aproape orice „punct critic”. De exemplu daca introducem x, automat se transforma in x^1, sau daca introducem 2x^2+4, automat se transforma in 2x^2+4x^0, sau nu trebuie sa scriem semnul de „\*” cand un monom are un coeficient: 2x automat se transforma in 2\*x. Pentru a verifica daca rezultatul inmultirii, adunarii, derivarii etc. a fost corect folosim JunitTest. TestJunit contine toate operatiile facute si folosind assertEquals putem verifica ce rezultat a dat calculatorul si care este rezultatul corect. In clasa TestRunner in metoda main() rulam testul din clasa TestJunit si afisam erorile si cauza lor. La sfarsit afisam daca testul s-a terminat cu succes sau a esuat la un punct.

**public** **class** TestRunner {

**public** **static** **void** main(String[] args) {

Result result = JUnitCore.*runClasses*(TestJunit.**class**);

**for** (Failure failure : result.getFailures()) {

System.***out***.println(failure.toString());

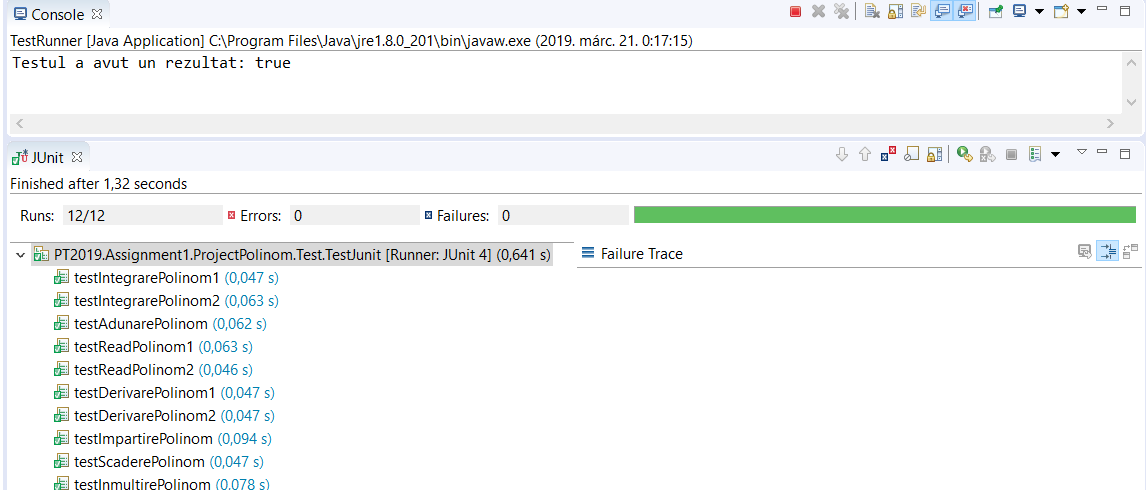
}

System.***out***.println("Testul a avut un rezultat: " + result.wasSuccessful());

}

}

Dupa rularea testarii din clasa TestRunner vom obtine rezultatele:



**6. Concluzii**

Acest calculator este o aplicatie foarte folositoare, mai ales daca utilizatorii sunt elevi sau persoane care lucreaza foarte mult cu polinoame, deorece prin acest calculator pot sa verifice foarte usor rezultatul obtinut de ei si rezultatul corect. Aceasta tema a fost folositoare deoarece am recapitulat notiunile legate de polinoame si am invatat cum se face impartirea polinomului. Acest calculator se poate dezvolta cu foarte mult, se mai pot adauga si alte operatii, de exemplu gasirea radacinilor sau la nivelul interfetei grafice: putem sa cream un „History” in care se afla toate operatiile facute anterior sau la o singura apasare de butom putem sa afisam toate cele 6 operatii.

**7.Bibliografie**

* Pentru diagrame StarUML
* https://stackoverflow.com/questions/8895337/how-do-i-limit-the-number-of-decimals-printed-for-a-double
* https://www.youtube.com/watch?v=FsotIB0Usvw
* https://www.tutorialspoint.com/junit/junit\_basic\_usage.htm
* https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
* https://stackoverflow.com/questions/26913923/how-do-you-change-the-size-and-font-of-a-joptionpane