Calculator de polinoame

Tehnici de programare

Tema 1

Alexandru Nistor Botolan
An 2 Seria B Grupa 30227
Facultatea de Automatica si Calculatoare
Sectia Calculatoare si Tehnologia Informatiei

1.Obiectivul temei

Obiectivul principal al temei este de a crea un calculator de procesare al polinoamelor, de o singura variabila cu coeficienti intregi.

Obiectivele secundare sunt reprezentate de realizarea unei interfete grafice pentru a introduce polinoamele efective si descompunerea idei de polinom in mai multe componente pentru a usura munca cu acestea.

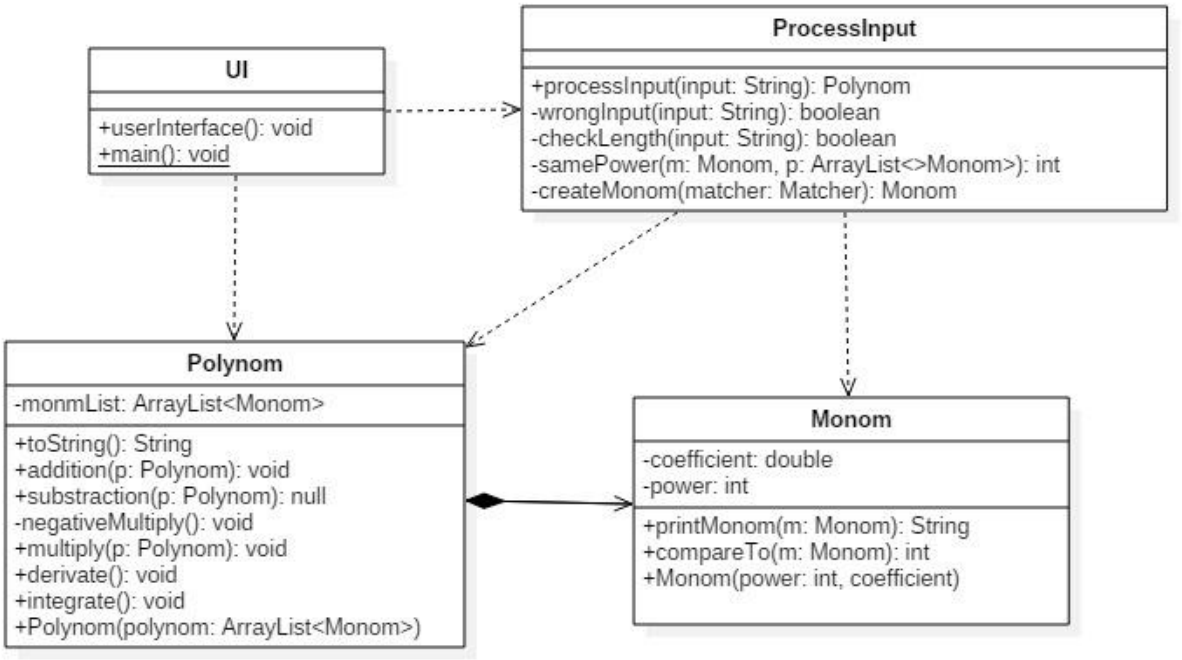
2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori

Acest calculator de polinoame trebuie sa realizeze operatii de baza pe polinoame: adunare, scadere si inmultire, derivare si integrare. Un polinom este compus din mai multe monoame, iar un monom este alcatuit dintr-un coeficient intreg si o putere numar natural. Operatiile trebuie sa se realizeze intre 2 polinoame, astfel avem nevoie de doi operanzi. In cazul in care in care unul din polinoame este introdus gresit, utilizatorul trebuie anuntati printr-un mesaj corespunzator, astfel nu vom face asumptie la corectitudinea datelor de intrare, astfel vom avea nevoie de o solutie care sa ne semnaleze cand introducerea este incorecta.

Presupunand introducerea corecta a polinoamelor, utilizatorul trebuie sa selecteze operatia pe care acesta o doreste, si afisarea rezultatului.

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

Diagrama UML



Deoarece un polinom este alcatuit din mai multe monoame, am optat prin descrierea a doua clase: „Monom” si „Polynom”, acestea doua constituind polinomul pe care se vor realiza operatiile, acestea doua fiind grupate intr-un pachet numit „objects”. Memorarea polinomului introdus se va realiza printr-un ArrayList<Monom>, fiecare element de tip Monom din lista va fi caracterizat de un camp „coefficient” care constituie coeficientul monomului si „power” care va constitui exponentul monomului.

Operatia de input am realizat-o intr-o clasa separata care se ocupa doar cu procesarea sirului de caractere primit de utilizator, separarea acestuia in monoame si trimiterea acestuia mai departe spre clasa „Polynom” pentru a fi memorata.

Clasa UI se va ocupa de relatia dintre utilizator si program. Aceasta trebuie compusa din doua campuri pentru a introduce polinoamelor, un camp pentru afisarea rezultatului si butoane pentru a selecta operatia dorita de catre utilizator.

La final avem clasa Main din care se va face apelul pentru interfata utilizator care va duce la pornirea programului.

4. Implementare

Incepand cu interfata utilizator, aceasta a fost realizata cu ajutorul librariei Swing. Se creaza un JFrame care va contine 2 textfield-uri prin intermediul carora utilizatorul va introduce polinomul, 2 label-uri in dreptul fiecarui textfield care va indica ordinea polinoamelor, un label care va afisa rezultatul operatiei si 5 butoane care va reprezenta cate una dintre cele 5 operatii: adunare, scadere, inmultire, derivare si integrare. Am impartit fereastra in 3 paneluri: unul care va contine cele 2 textfield-uri si cele 2 label-uri care sunt responsabile de input, un panel pentru rezultat care va contine label-ul care afiseaza rezultatul, iar ultimul panel va contine cele 5 butoane responsabile pentru operatii.

Clasa" Monom" este alcatuita din doua campuri: „coefficient", care va memora coeficientul monomului, si „power" care va memora exponentul monomului. Aceasta clasa, pe langa metodele get si set, implementeaza: „printMonom(Monom)", „compareTo(Monom)". Metoda „printMonom(Monom)" este utilizata pentru a returna un sir de caractere care este echivalent cu valoarea monomului. Parametrul transmis ca si parametru este utilizat pentru a determina daca monomul respectiv este primul in lista de monoame, pentru a realiza o afisare cat mai prietenoasa. Aceasta metoda este utilizata pentru a returna polinomul rezultat in urma unei operatii, metoda apelata in clasa „Polynom", unde va fi dezoltata in descrierea clasei „Polynom". Metoda „compareTo(Monom)" este utilizata pentru a compara monoamele in functie de exponent, metoda care ne ajuta pentru ordonarea listelor care utilizeaza aceasta clasa, dar si pentru afisarea polinoamelor in ordine crescatoare, in functie de exponent.. Desi polinomul utilizeaza coeficienti intregi, am optat pentru utilizarea tipului „double" in loc de „int" pentru a elimina complicatia utilizarii fractiilor si pentru o afisare clara. Un exemplu unde este utila aceasta

decizie este afisarea 0.5x^2 in loc de 1/2x^2, sau 0.49X^25143 in loc de 12544/25144x^25144.

Clasa „Polynom” are un singur atribut: o lista de monoame care reprezinta valoarea polinomului. Metodele care sunt implementate in aceasta clasa sunt:

a. toString() care este folosita in afisarea polinomului, aceasta returnand un sir de caractere. Aceasta metoda apeleaza „printMonom(Monom)” din clasa „Monom”, primind pe rand cate un sir de caractere, valoare fiecarui monom din lista.
b. addition(Polynom) este utilizata pentru a realiza operatia de adunare a doua polinoame. Apelul metodei nu este static, astfel primul operand este polinomul pe care se face apelul, iar al doilea este cel trimis ca parametru. Se parcurge lista celui de-al doilea operand, luand un element din lista, se cauta monomul cu exponentul comun din lista primului operand. In cazul in care este gasit, monomului din lista primului operand ii este atribuita suma celor doi coeficienti. In cazul in care nu este gasit monomul din al doilea operand este adaugat in lista primului operand. Daca valoare coeficientului este 0 in urma operatiei, monomul este memorat intr-o lista separata, iar la finalul operatiei toate monoamele care au valoare coeficientului 0 vor fi eliminate din lista. Pentru ca monoamele pot sa nu fie ordonate in urma operatiei, va fi apelata metoda „Collections.sort(monoms)” care va sorta lista cu ajutorul metodei „compareTo()” din clasa „Monom”
c. negativeMultiply() este o metoda care va inmulti fiecare coeficient al monoamelor listei de monoame cu -1.
d. substraction(Polynom) are o implementare simpla. Pentru ca o scadere a doua polinoame este practic inmultirea cu -1 si adunarea lor, metoda apeleaza doua metode descrise mai sus. Mai intai se apeleaza „negativaMultiply()” pentru a inversa semnul coeficientilor, iar apoi se apeleaza metoda „addition(Polynom)”.
e. mutiply(Polynom) este o metoda mai complexa. Avem nevoie de doua liste: una pentru rezultat si alta pentru stergerea monoamelor cu coeficientul 0. Pentru simplitate ma voi referi la polinoamele p si q, p->primul operand, q-> al doilea. Se va parcurge lista de monoame p, iar pentru fiecare element din p va fi parcursa lista lui q, coeficientul/exponentul monomului curent

din p va fi inmultit/adunat cu fiecare monom din q, rezultatul acestei operatii va fi memorat intr-o variabila auxiliara de tip Monom. Se verifica daca exponentul acestui monom se regaseste in lista de rezultate, in caz afirmativ si aduna coeficientii, in caz contrar se adauga un nou element in lista de rezultat. La fel, pentru a fi eliminate elementele cu coeficientii 0, se formeaza o lista cu aceste monoame, iar la finalul operatiei sunt eliminate din lista de rezultate. Se sorteaza lista pentru aspect, lista variabilei de instanta este curatata, iar in aceasta sunt copiate elementele listei de rezultat.

f. derivate() implementeaza operatia de derivare a polinomului. Pentru fiecare element din lista de monomame, valoarea coeficientului este inmultia cu exponentul, iar valoarea exponentului este decrementata cu 1. In cazul in care valoarea exponentului ultimului element din lista este -1, inseamna ca a fost derivata o constanta, astfel aceasta trebuie eliminata din lista.
g. Integrate() implementeaza operatia de integrare a polinomului. Pentru fiecare element din lista de monoame valoarea coeficientului este impartita la valoarea exponentului + 1, iar exponentul este incrementat cu 1.

Toate metodele din aceasta clasa au accesul public, mai putin metoda de negativeMultiply() care este private.

Clasa „ProcessInput” se ocupa cu prelucrarea sirului de caractere primit de la utilizator si despartirea acestuia in monoame pentru a fi mai usor de procesat. Se implementeaza urmatoarele metode:

a. processInput(String) primeste ca parametru un sir de caractere care reprezinta polinomul introdus de utilizator. Ne vom ajuta de o expresie regulata cu ajutorul careia vom identifica usor componentele

monomului, dar si daca acesta a fost introdus gresit. Pentru a verifica daca utilizatorul a introdus un un polinom valid sau nu, ne vom ajuta de metoda „wrongInput(String)” care va fi descrisa mai jos. Daca rezultatul este negativ, atunci putem trece la procesarea polinomului si despartirea acestuia in monoame. Cat timp expresia regulata va descoperi un match in sirul introdus(match) va trebui sa despartim monomul descoperit in coeficient si exponent, problema rezolvata de metoda „createMonom(Matcher)”. Dupa ce avem monomul descompus in coefficient si exponent, verificam daca exonentul exista in polinomul construit. In caz afirmativ, coefficientul este adunat, altfel este adaugat listei. La finalul acestei metode, vom returna un obiect de tip „Polynom”, care constituie polinomul valid introdus de catre utilizator.

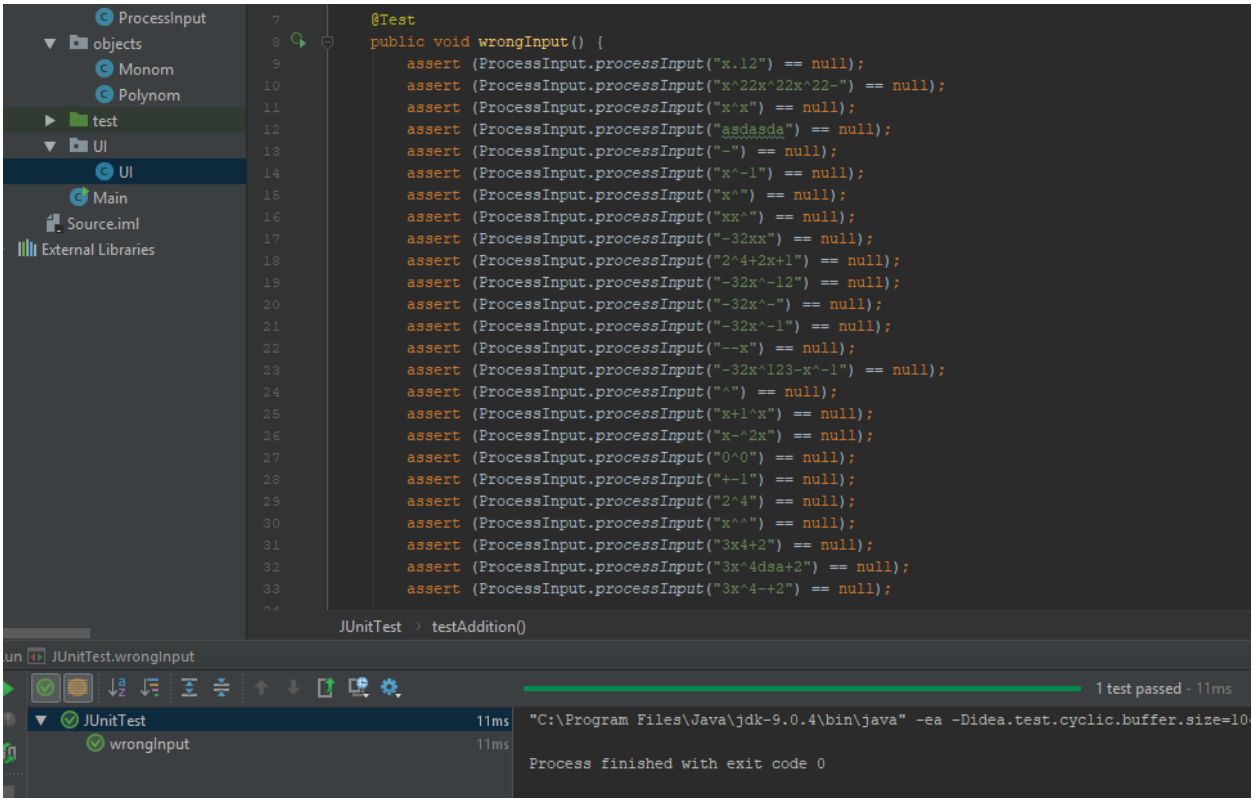
b. wrongInput(String) primeste ca parametru un sir de caractere care reprezinta polinomul introdus de utilizator. Cu ajutorul unei expresii regulate, vom identifica daca cazuri mai speciale in care polinomul este invalid. In cazul in care nu este gasita o expresie ilegala vom calcula lungimea tuturor grupurilor gasite de expresia valida, in cazul in care lungimea calculata nu corespunde cu cea a sirului introdus, expresia este ilegala, astfel sirul introdus nu poate fi procesat, iar metoda va returna false. In cazul in care sirul introdus de catre utilizator este valid, metoda va returna true.
c. checkLength(String) primeste ca parametru un sir de caractere care reprezinta polinomul introdus de utilizator. Cu ajutorul unei expresii regulate vom calcula lungimea tuturor grupurilor gasite, si vom compara lungimea sirului cu valoarea calculata, in cazul in care sunt egale returneaza true, altfel false

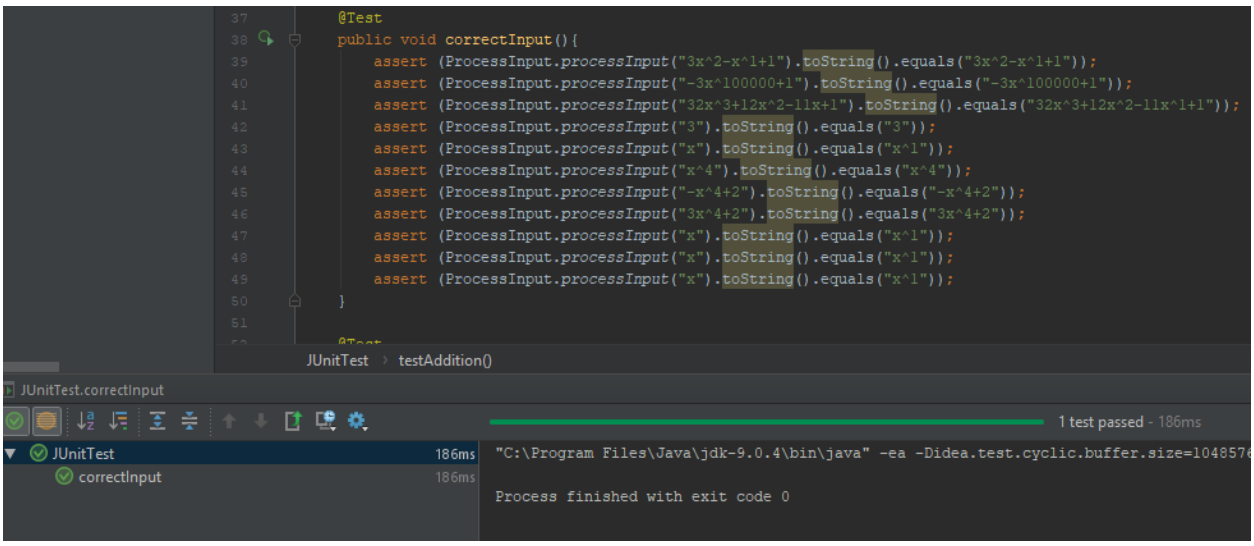
d. samePower(Monom, ArrayList<Monom>) este utilizata pentru a gasi indexul elementului Monom trimis ca parametru in lista ArrayList<Monom>. In cazul in care este gasit elementul in lista, va fi returnat indexul elementului in lista, altfel va fi returnat -1.
e. createMonom(Matcher) este utilizata pentru a crea monomul care a fost gaist de catre parametrul Matcher. Grupul 3 si 4 din Matcher este

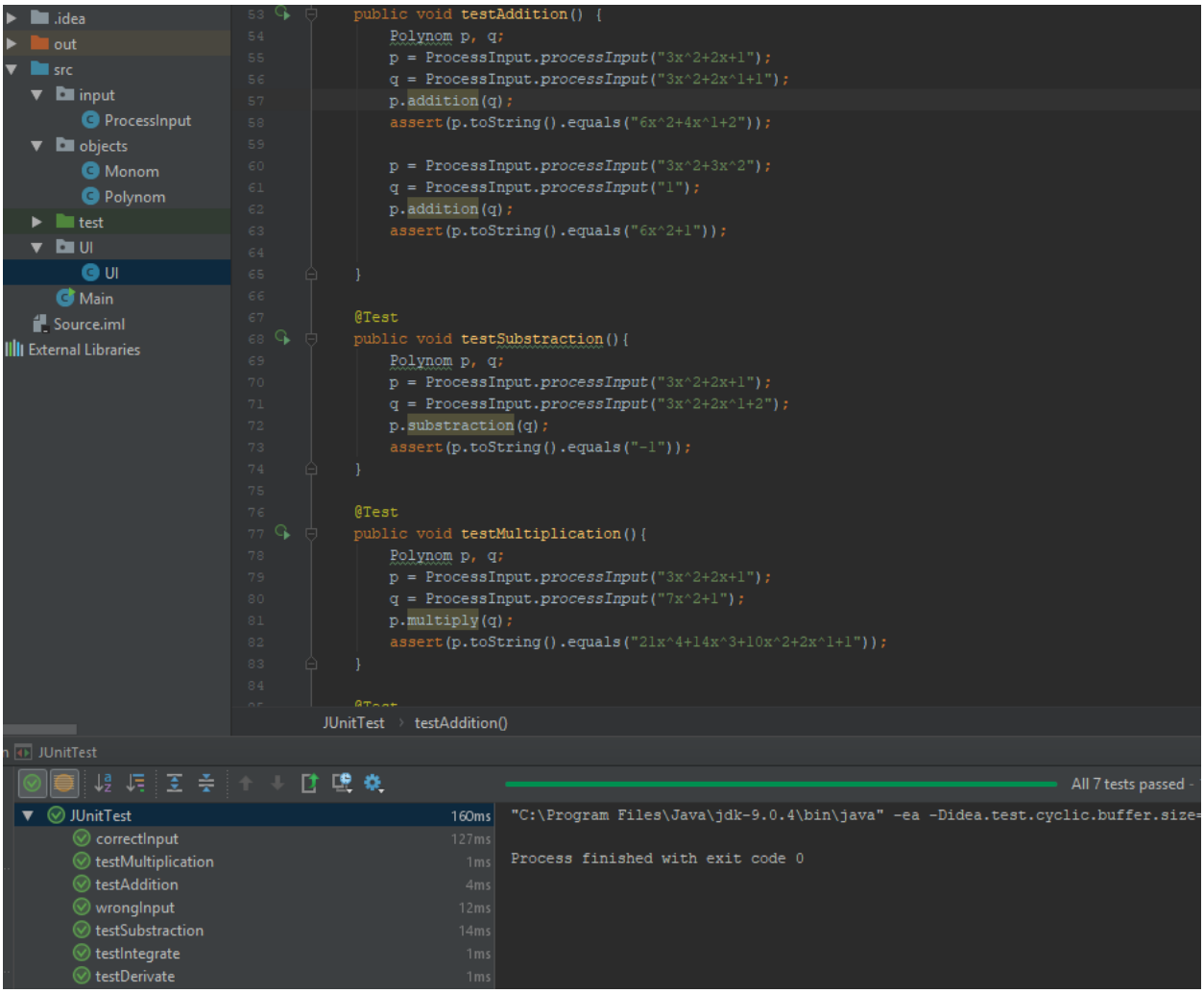
ocupat de catre semnul coeficientului, daca exista si este -/+, coeficientul monomului este asignat la -/+1. In cazul in care acesta nu exista, semnul implicit este +. Grupul 5 este ocupat de catre valoare numerica a coeficientului, in cazul in care acesta exista, este convertit la double si inmultita cu valoarea rezultata in urma definirii semnului. Grupul 7 se refera la existenta elementului „x”, in cazul in care acesta exista, exponentul are valoarea cel putin 1, in caz contrat valoarea sa este 0. Grupul 9 se refera la valoarea exponentului, in cazul in care exista, valoarea este convertita la int si este asignata exponentului. La final este returnat un nou obiect de tip Monom cu valoarea coeficientului si exponentului gasite mai sus.

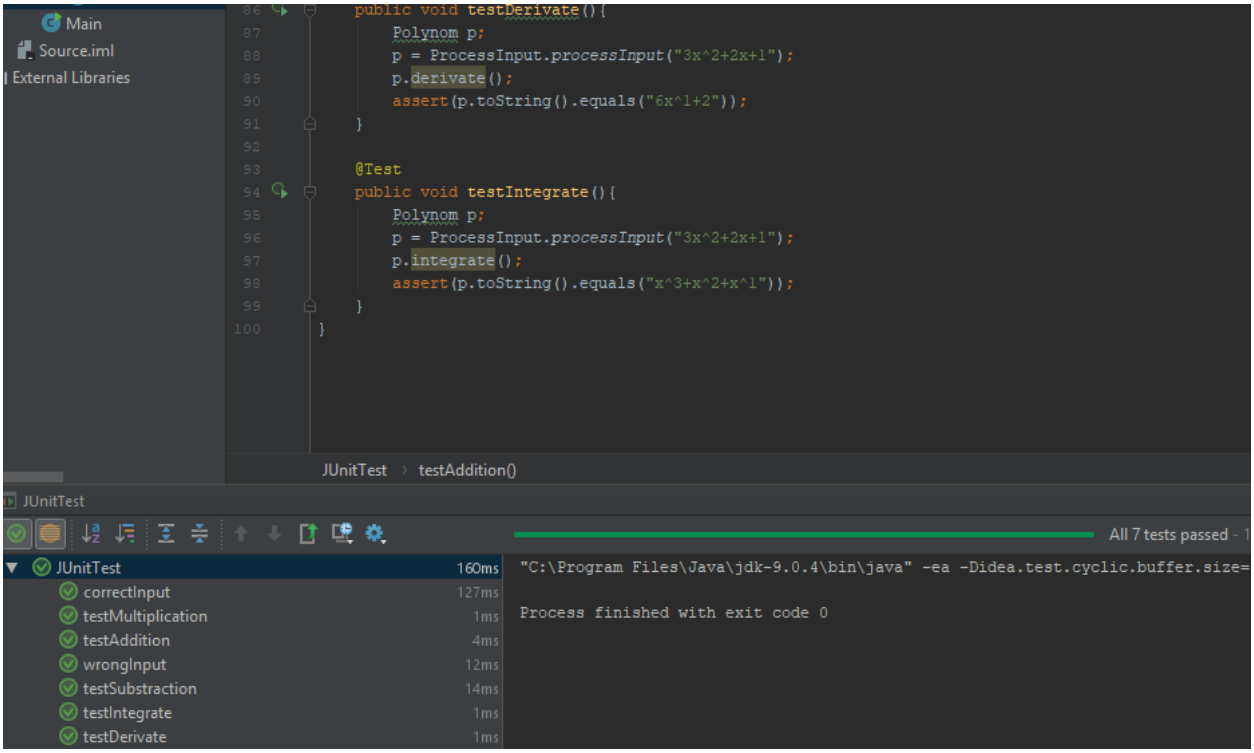
5. Testare

Pentru testare am utilizat Junit. Pentru test, am acoperit testele pentru input corect/gresit si operatiile de adunare, scadere, inmultire, impartire.



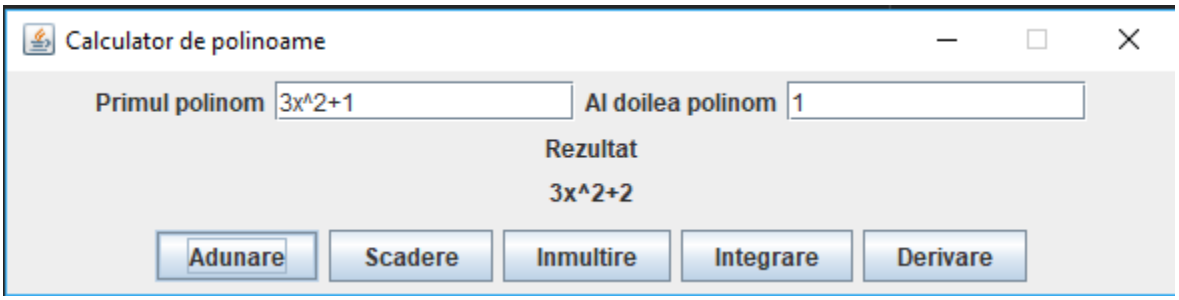



.idea
out
src
input
ProcessInput
objects
Monom
Polynom
test
UI
UI
Main
Source.iml
External Libraries
public void testAddition() {
Polynom p, q;
p = ProcessInput.processInput("3x^2+2x+1");
q = ProcessInput.processInput("3x^2+2x^1+1");
p.addition(q);
assert(p.toString().equals("6x^2+4x^1+2"));
p = ProcessInput.processInput("3x^2+3x^2");
q = ProcessInput.processInput("1");
p.addition(q);
assert(p.toString().equals("6x^2+1"));
}
@Test
public void testSubstraction(){
Polynom p, q;
p = ProcessInput.processInput("3x^2+2x+1");
q = ProcessInput.processInput("3x^2+2x^1+2");
p.substraction(q);
assert(p.toString().equals("-1"));
}
@Test
public void testMultiplication(){
Polynom p, q;
p = ProcessInput.processInput("3x^2+2x+1");
q = ProcessInput.processInput("7x^2+1");
p.multiply(q);
assert(p.toString().equals("21x^4+14x^3+10x^2+2x^1+1"));
}
JUnitTest › testAddition()
JUnitTest
All 7 tests passed
JUnitTest 160ms
correctInput 127ms
testMultiplication 1ms
testAddition 4ms
wrongInput 12ms
testSubstraction 14ms
testIntegrate 1ms
testDerivate 1ms
"C:\Program Files\Java\jdk-9.0.4\bin\java" -ea -Didea.test.cyclic.buffer.size
Process finished with exit code 0

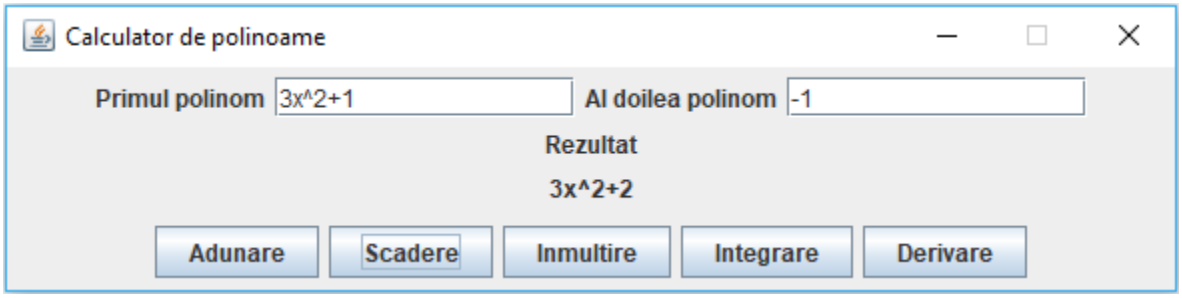


Pentru testele de input, se primeste ca si date de intrare un sir de caractere si se testeaza sirul de caractere rezultat. Pentru expresii ilegale se asteapta un null, iar petru expresii valide se asteapta un sir de caractere, care este comparat cu ceea ce trebuie sa rezulte. Pentru testele de operatii se primeste unul sau doi parametrii, si se testeaza sirul de caractere rezultat in urma operatiei care trebuie sa corespunda cu sirul de caractere constant, care este definit in teste ca si rezultatul care trebuie sa apara.

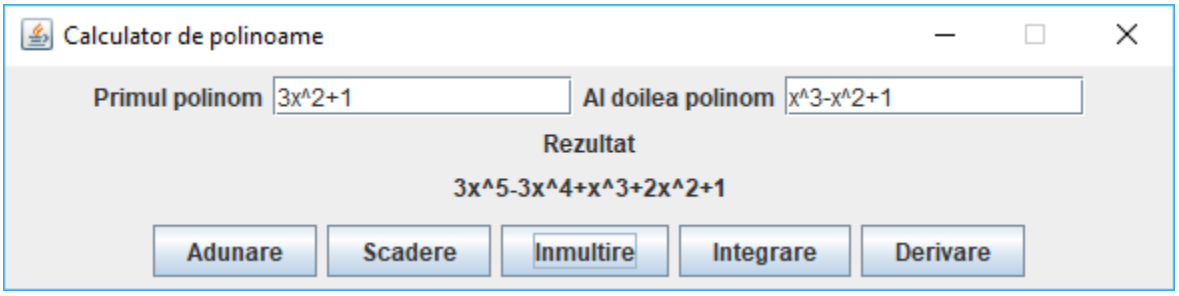
6. Rezultate



Pentru screenshotul de mai sus, se observa textfieldurile pentru introducerea polinomului, labelul de rezultat, dar si butoanele de operatii. Pentru exemplul de mai sus polinomul 3x^2+1 este adunat cu polinomul 1, iar rezultatul este 3x^2+2.



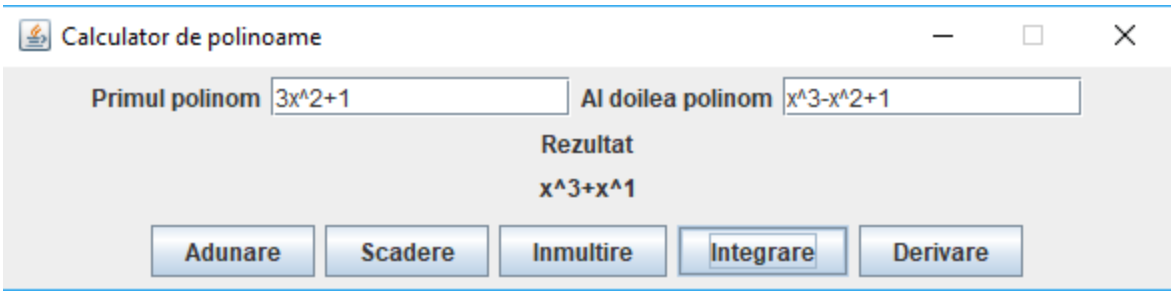
Pentru acest exemplu se realizeaza operatia de scadere a celor doua polinoame, rezultatul pentru (3x^2+1)-(-1) este 3x^2+2.



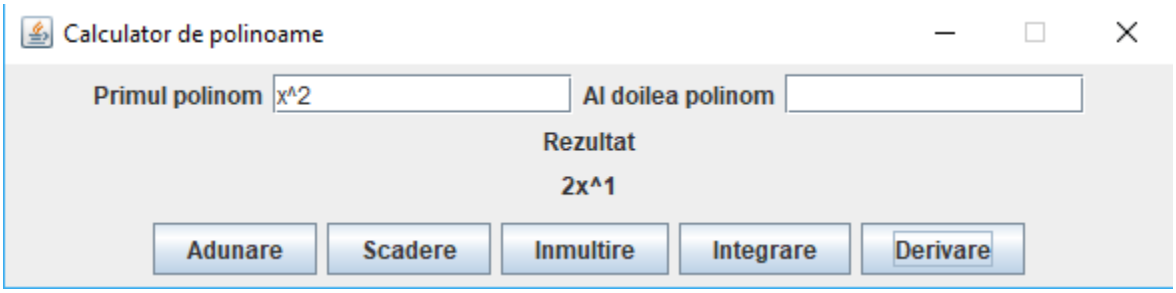
Pentu exemplul de mai sus se incearca inmultirea a doua polinoame. Astfel operatia (3x^2+1)*(x^3-x^2+1) =

3x^5–3x^4+3x^2+x^3-x^2+1 =

3x^5-3x^4+x^3+2x^2+1



Operatia de integrare se realizeaza pe primul polinom, astfel 3x^2+1 integrat este 3/3x^3+1/1x^1 = x^3+x^1



Operatia de derivate se realizeaza pe primul polinom, astfel x^2 derivat este 2x^1.

7. Concluzii

Acest calculator de polinoame este util pentru elevii de liceu pentru a calcula usor si repede operatii simple pe polinoame mai complexe care ar dura ceva timp pe foaie. Din aceasta tema am invatat dezvoltarea de interfete grafice, de a lucra cu elemente caracteristice OOP dar si elementele de baza pentru a dezvolta acest tip de aplicatii. Ca si dezvoltari ulterioare se pot adauga calculul valorii polinomului intr-un punct, de a desena graficul functiei, de a determina concavitatea/convexitarea sau continuitatea functiie.