**Sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi**

**Balus Dan**

Grupa 30227

1. *Obiectivul temei*

Polinoamele împreună cu operațiile cu acestea sunt unele dintre cele mai importante și commune structure matematice. De asemenea, polinoamele pot aproxima orice funcție continuă și derivabilă suficient de bine încât să fie folosite în domenii diverse. Cu ajutorul acestora, se pot determină sau anticipa anumite fenomene în știință.

Scopul acestui proiect este de a implementa o interfață de lucru cu polinoame de o singură variabila cu coeficienți întregi cu scop demonstrativ pentru a exemplifica modul în care acestea sunt create și manipulate prin intermediul Programării Orientate pe Obiecte.

Operațiile sunt de adunare, de scădere, de înmulțire, de împărțire, de derivare și de integrare

# *Analiza problemei și modelarea acesteia*

Prin analiza problemei, ne referim la un prim set abstract de operații și proprietăti prin care încercam să depistăm eventualele însușiri și comportamente ale proceselor necunoscute. Programarea orientate ne permite să tratăm problemă la un nivel superior fără să fim constrânși de eventualele caracteristici tehnice.

Aceastra strategi de conceptulizare este benefică din punct de vedere al atribuirii componentelor constituent structure cu o legătură directă în lumea înconjurătoare.

Se poate pleca de la specificaria proiectului pentru a ne forma o idee generală asupra proiectului și anume:

* Substantive - pot devein clase
* Verbe -pot devein eventuale metode

Programul va putea fi accesat de mai multe persone, iar acest lucru duce la creearea interfeței. Este important și modul de codificare a informației, alegând introducerea în forma canonică a acestuia. Ex:

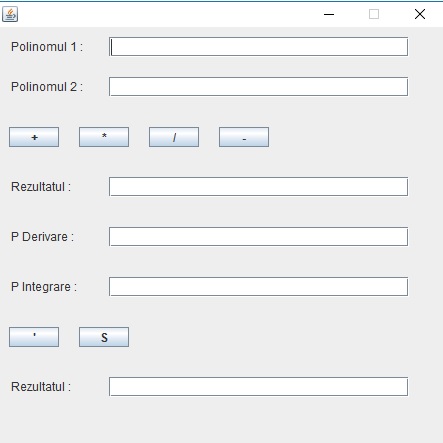
5x^3 + 4x^2 -5x^1 + 3x^0

* 1. *Cazuri de utilizare*

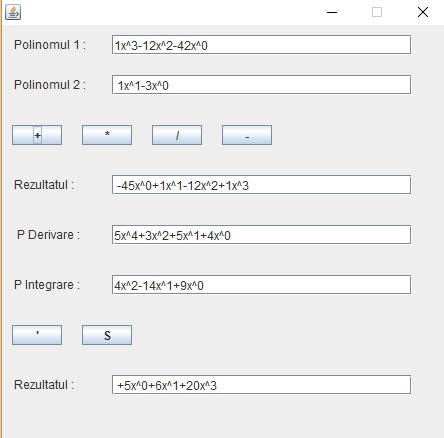
După stabilirea metodei de intrare vom putea vorbi de următoarele:

* utilizatorul introduce un polinom sub forma ax^n+ bx^n-1+….+ zx^0, unde a, b…z sunt coeficienții, x este variabila de derivat, iar n este puterea; de precizat că ordinea în care se introduc monoamele ce conțin polinomul nu contează
* suma, diferența, produsul, împărțirea între primul polinom și cel de al doilea
* integrarea sau derivarea unui polinom , separate față de cele introduse, pentru primele 4 operații

Interfața:



Un exemplu de caz de utilizare:



* 1. *Scenarii*
* Introducerea altor caractere decât cele premise
* Introducerea a necunoscute suplimentare, pe lângă x/ X
* Cazurile în care apar operațiile cu 0, respective rezultatul 0
* Introducerea numerelor cu semn
* Introducerea puterilor începând cu cifra 0 (Ex: 02 în loc de 2)
* Introducerea de text fără semnificație numerică
* Spații multiple (“ “)

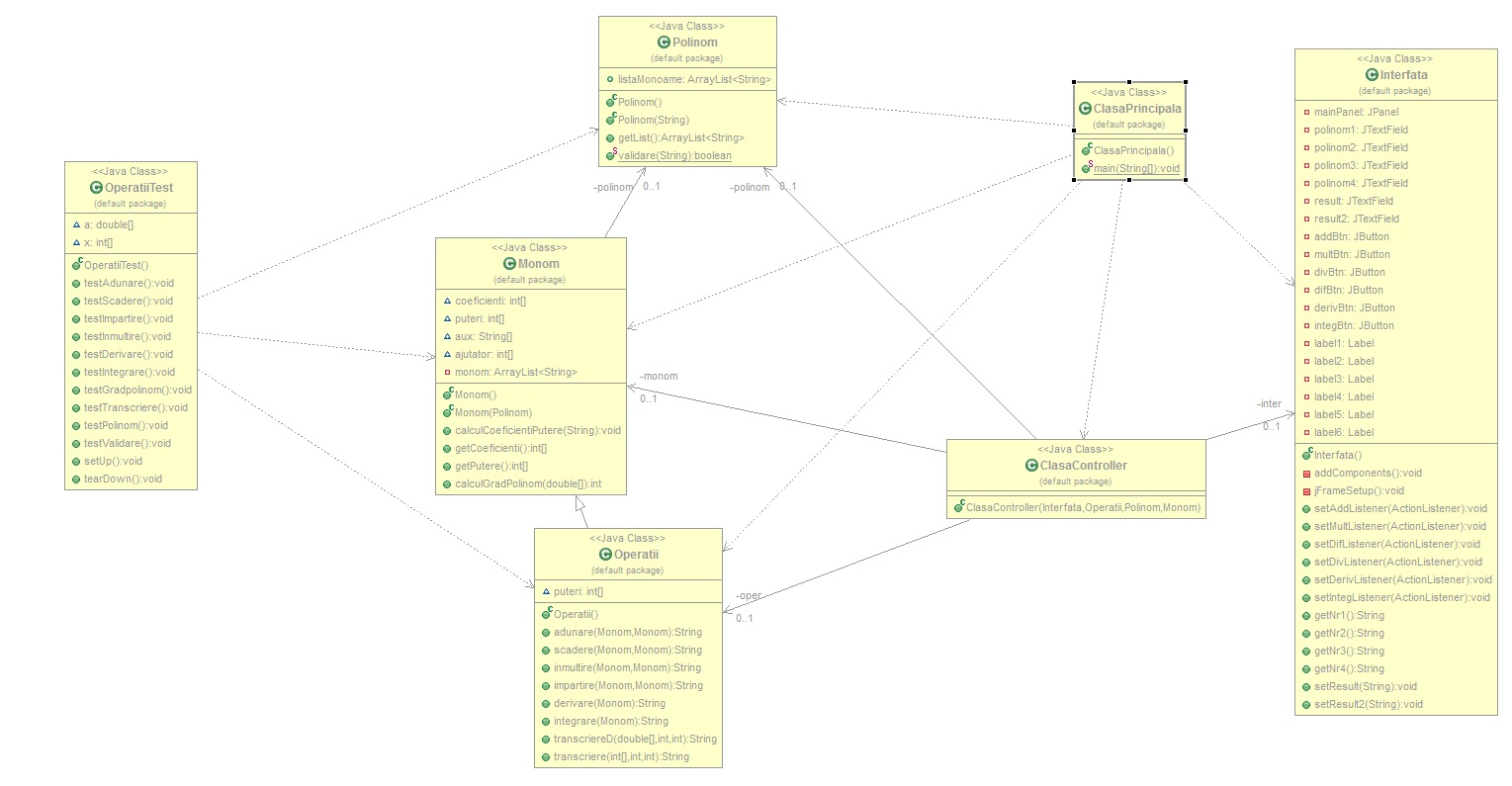
1. *Proiectare si Implementare*
   1. *Diagrame UML*

UML furnizează mecanisme a claselor membre, precum attribute, metode, și informații adiționale despre ele.

O relație este un termen general care acoperă tipuri specifice de conexiuni logice găsite în diagramele de clase și de obiecte.UML descrie următoarele relații:

* Asociere
* Agregare
* Generalizare
* Dependența

Diagrama UML a proiectului:



* 1. *Preiectare clase*
     1. *Clasa Interfata*

Cu ajutorul acesteia se creează interfața grafică folosită . Aici sunt inițializați, de asemenea, ascultătorii, Cu rolul de a surprinde acțiunile ulterioare ale utilizatorului cu ajutorul evenimentelor

* + 1. Clasa ClasaController

Această “controleaza” ascultătorii și trimite către clasa Polinom Stringul cu polinomul.

* + 1. Clasa Polinom

Procesează datele primite de la clasa ClasaController și trimite mai departe către clasă Monom, lista de monoame.

* + 1. Clasa Monom

Prelucrează datele primite de către clasă Polinom.

* + 1. Clasa Operații

Implementează, efectiv, operațiile cu polinoame .

* + 1. ClasaPrincipala

Conține funcția main, iar aici sunt inițializate celelalte clase.

* + 1. Clasa OperatiiTest

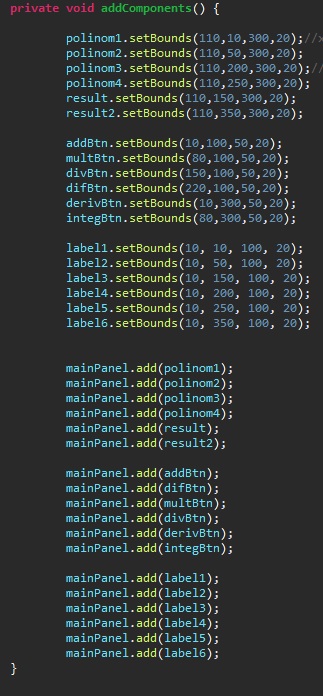
Implemeteaza tetareA proiectului făcută cu Junit .

* 1. *Detalii implementare clase* 
     1. *Clasa Interfata*

Conține un jPanel pe care se contrueste intreagă intergata. Aici sunt declarate Text Field-urile, Butoanele și etichetele .



Nu am optat pentur niciun Layout anume, așezarea componentelor în interfată o voi face cu setBounds, în metoda addComponents unde voi adaugă butoanele în interfată.



De asemenea voi inițializa fereastra pentru 450 x 450 pxeli și voi folosi comanda: setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

pentru a ieșii din program atunci când se întâmplă evenimentul (apăsăm X). Tot aici sunt inițializate și get erele și set-erele care for citi, respective scrie în label, inclusive ascultătorii, după cum am precizat mai sus.



Constructorul este suprascris pentru a apela metodele dorite si anume:

Add((mainPanel());

mainPanel.setLayout((**null)**);;

addComponents());;

jFrameSetup() () ;;

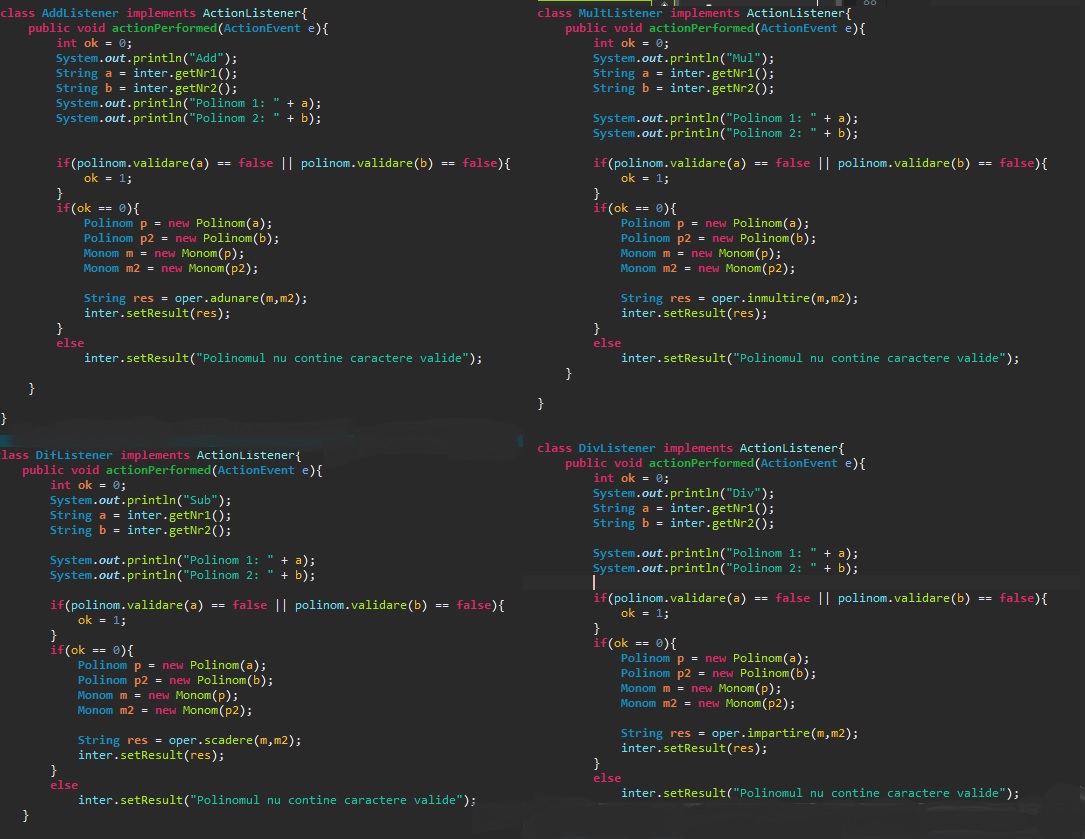
* + 1. *Clasa Controller*

Este clasa care gestionează toate acțiunile venite din partea utilizatorului și le corelează cu funcționalitățile implementate în proiect. Din aceste motive aici se implementează interfața ActionListener și implicit metoda actionPerformed() care primește ca parametru un eveniment produs de către utilizator. Sunt implementate metode pentur fiecare operație, acest lucru nefiind neapărat necesar deoarece nu există paralelism(nu se pot executa două operații în același timp), dar am optat pentru această metodă pentru a fi mai ușor de observat diferențele și pentur a evita metode de lungime mare.

În continuare voi exemplifica pentru adunare.

Aici apelăm metodele **getNr1()** și **getNr2()** care vor lua ca stringuri datele introduse de utilizator, după care se verifică dacă s-au introdus caractere valide (Dacă nu, se părăsește afișând în interfață un mesaj), după care se instanțiază polinom și monom pentru a se putea apela operația efectivă. La sfârșit se va afișa mesajul în label-ul destinat.

Celelalte operații sunt similare, puțin diferite fiindca avem derivare și integrare, unde se va folosi doar un polinom, cu altă locație destinată în interfață.



Pentru **Derivare** si **Integrare**



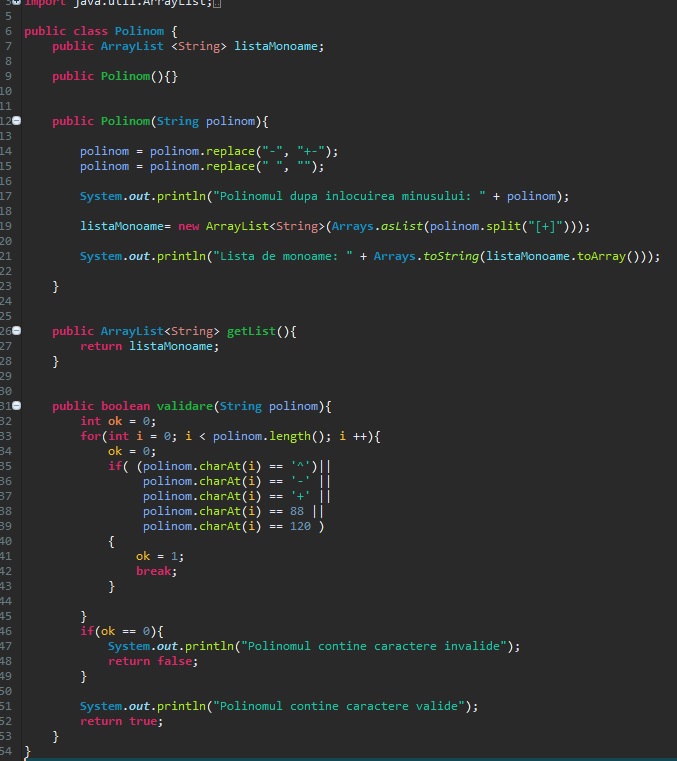
* + 1. *Clasa Polinom*

În clasa Polinom vom creea și partaja stringul scris de către utilizator într-o listă de monoame. de tipul ArrayList.

În primul pas, se va înlocui semnul – cu semnele +- pentru a putea face, mai apoi, split după semnul +. Tot aici vom elimina și eventualele spații plasate la introducerea datelor.

Vom implementa metoda getList de tip ArrayList pentru a putea returna lista de monoame către clasa Monom.

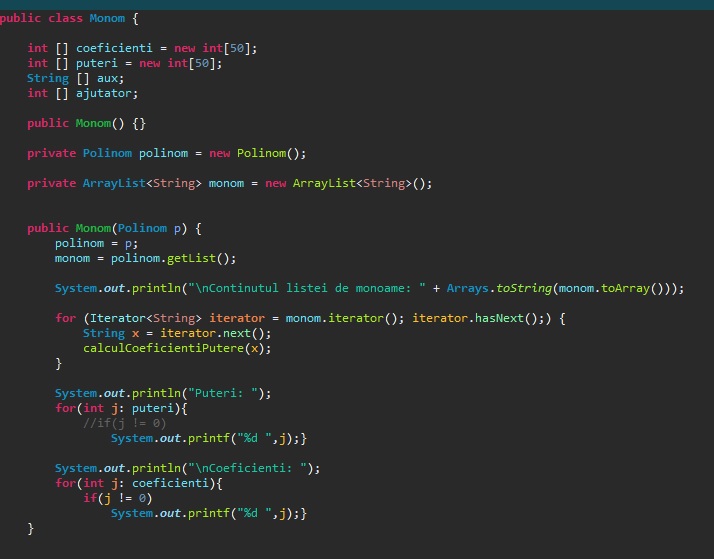
Aici este implementată metodă booleană validare .



* + 1. *Clasa Monom*

Aici se va lucre cu lista de monoame primită de la clasa Polinom importandu-se si aici java.util.ArrayList;

Pentru trecerea prin aceasta se va folosi **for** cu interator pentru ca lucram cu liste.

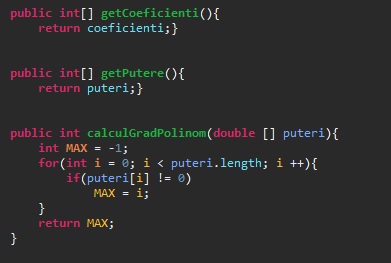


Pentru fiecare monom din lista de monoame se va aplică metoda split după caracterele “x^” și se for salva pentru fiecare în parte în 2 vectori unde vor fi stocați coeficienții pe indicele puteri și un vector de apariții ale puterilor(se va folosi conversie pentru a putea obține int).

După split, listele de monoame vor arată așa: [a, b], unde a este coeficientul și b este puterea.



Cu ajutorul metodelor **getCoeficienti getPutere**  ce returnează doi vectori vom putea trimite mai departe vectorii de coeficienți și putere De asemenea aici este implementată și metoda de calculare a gradului polinomului pe care o vom folosi la implementarea operațiilor.



* + 1. *Clasa Operatii*

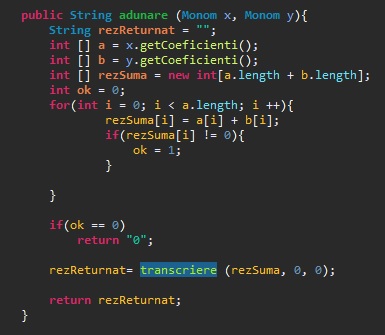
Aici se implementeaza operatiile

Operatiie se vor efectua folosind vectorii prelucrați de clasa monom, în urma cărora se vor forma stringuri cu rezultatulele apelând metodele **transcriere** și **transcriereD** care vor fi afișate în interfață.

Clasa **Operații** va moștenii clasa **Monom** deoarece vom avea aici nevoie de metodele impementate de aici.

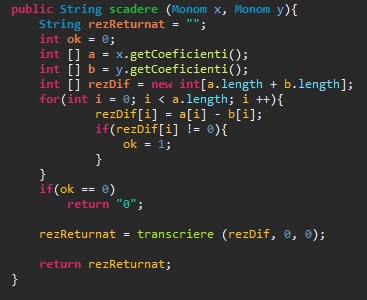
* + - 1. *Adunarea*

Aici se vor aduna coeficienții (preluați de către get ere) de pe același indici, iar dacă toți coeficienții sunt 0 se va afișa “0” în interfață .



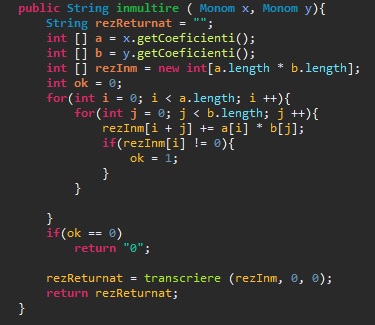
* + - 1. *Scaderea*

Similara adunarii, diferă faptul că aici coeficientii se scad.



* + - 1. *Înmulţirea*

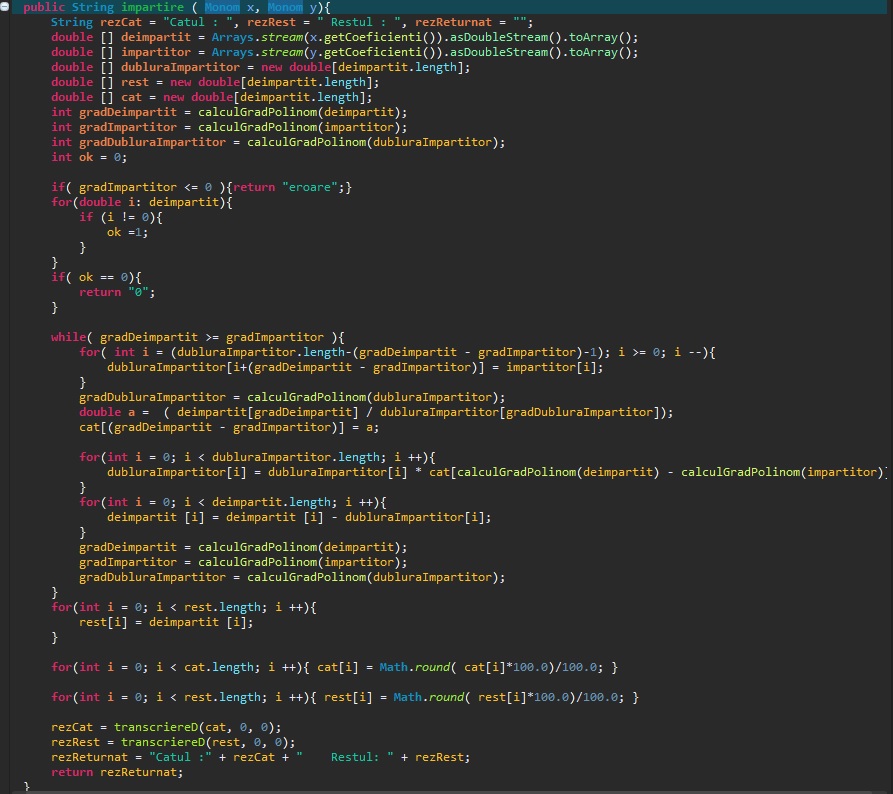
Similară înmulțirii și grupării polinoamelor în matematică, se vor folosi două **for-uri** care vor înmulți, pe rând, coeficienții de la puterile egale și “grupand” în forma restrânsă .



* + - 1. *Împărțirea*

Se va implementa împărțirea polinoamelor după algoritmul folosit în matematică( schema lui Horner) doar că aici termenii vor fi “shift ati” în dreaptă , cu numărul de poziții egal cu diferența gradelor celor 2 polinoame, deoarece coeficienții sunt ordonați crescător, nu descrescător că în căzut schemei.

Dacă împărțitorul este 0, atunci nu se va putea efectuă efectuă împărțirea și se va returna stringul “eroare” la interfață, iar dacă deîmpărțitul este 0, atunci rezultatul împărțirii este , prin urmare se va returna stringul “0”. Spre final se va folosi metoda round() împreună cu un artificiu pentru a afișa doar 2 zecimale după calcularea coeficientului/ coeficienților câtului și al restului.



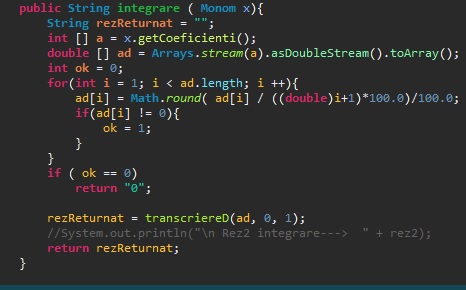
* + - 1. *Derivarea*

La derivare se va înmulți indicecele (puterea) cu coeficientul și se va trimite la funcția transcriere puterea cu un ordin mai mic începând de la elementul 1, deoarece de primul nu vom avea nevoie indifferent de polinomul introdus.



* + - 1. *Integrare*

Similar derivării, doar că aici se va trimite de la indicele 0 și se va “incrementa” cu 1 puteream iar ceficientii se vor impartti la indicele incrementat(puterea+1), păstrând doar două zecimale.



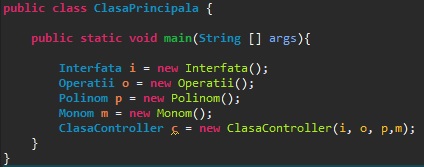
Metodele de transcriere primesc că parametrii vectorul de coeficienți, indexul de început al for ului de afișare și puterea afisată

Acestea construiesc stringul care va fi mai apoi trimis clasei ClasaController și afișat la interfată către utilizator



* + 1. *Clasa ClasaPrincipala*

Aici vom creea obiecte care vor fi trasmise Clasei ClasaController



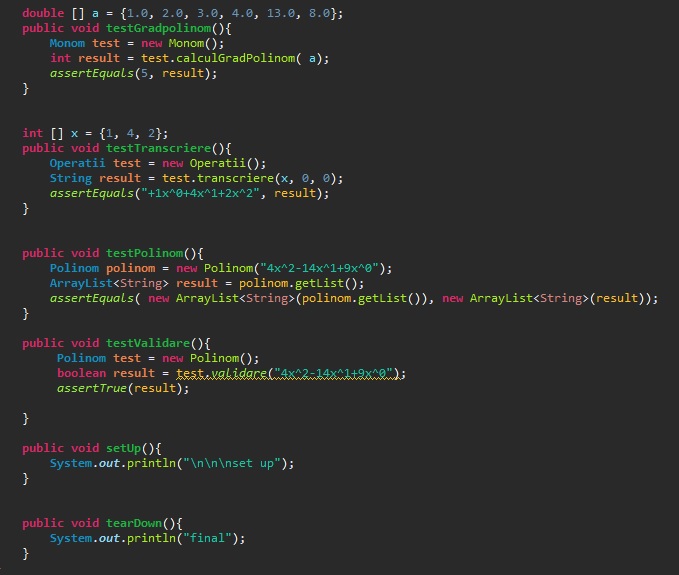
* + - 1. Clasa OperatiiTest

Aici se vor testa principalele meode cu ajutorul unității de testare framework Junit, una din familia de testare unit cunoscută ca xUnit, original SUnit.

Sumar, principiul folosirii acesteia poate fi rezumat la falpul că se trece manual valoarea de intrare și de ieșire a proiectului, iar în urma rulării acestuia cu exemplul adt de noi se verifică dacă rezultatul returnat de program este identic cu cel dat de către noi.

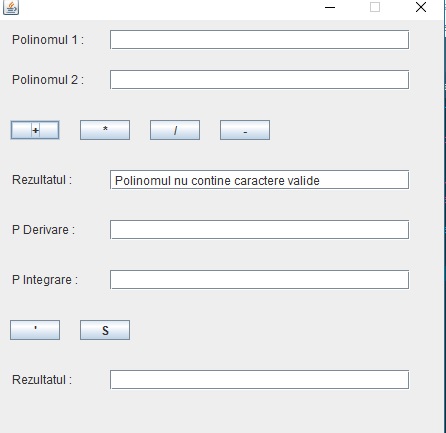
Se vor suprascrie metodele se setup și teardown pentru a afișa la consolă începutul și finalul testării

Clasă va moștenii clasă TestCase și se va folosi de metodele implementate de această.

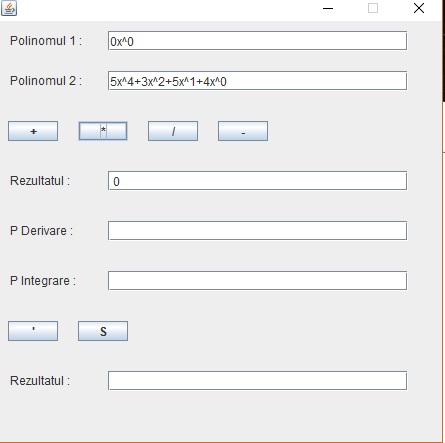


1. *Testarea datelor de intrare:*

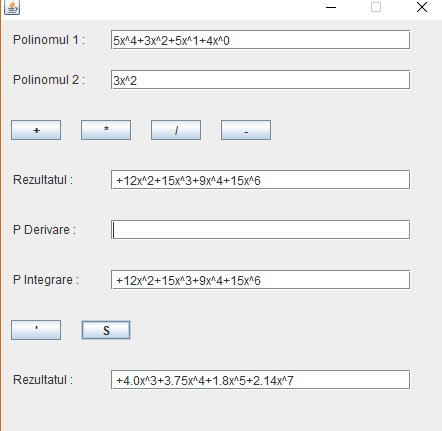
Cand campurile sunt goale si se aplica o operatie



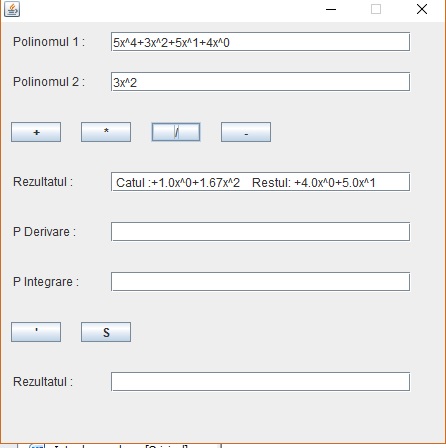
Inmultirea cu 0:



Integreare:



Impartirea a doua polinoame



1. *Rezultate*

Rezultatele, cât și valorile se află în grilă celor așteptate, programul având un oarecare grad de flexibilitate la nivelului modului de procesare a polinoamelor și parsare a expresiilor special, adica a string-utilor preluate.

# *Concluzii*

Acest proiect implementează cu succes un sistem de procesare a polinoamelor cu o singură variabilă și cu coeficienți întregi introduși sub formă canonică. Este un excelent punct de început pentru îmbunătățiri ulterioare care ar putea fi implementate ușor și cu succes.

Se poate modifica acest proiect astfel încât să se poată realiza operații cu polinoame cu două sau mai multe variabile și cu coeficienți de tip double. De asemenea se mai pot adduce modificări astel încât de se poată introduce polinomul și cu termini liberi, fără x, de genul **2x^2 + 5x +3** sau **x^2.** O altă dezvoltare ar putea fi optimizarea sau realizarea altor operații. Ex: limită, sumă etc.

Datorită acestei teme am aprofundat cunostiintele dobândite în primul semestru la Programarea Orientată pe Obiecte, sesizând importanța, tehnicile, convențiile folosite pentru programare și legătură lor cu viață reală.

Proiectul este realizat astfel încât să poată fi folosit cu ușurință și de către alte persone, nu neapărat avizate. De asemenea, aplicația poate fi cu ușurință folosită pentru verificarea operațiilor complexe.

# *Bibliografie*

* <https://stackoverflow.com>
* https://docs.oracle.com
* <https://rosettacode.org/wiki/Polynomial_long_division>
* <https://www.youtube.com/>
* <https://docs.oracle.com/javase/7/docs/api/>