



**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

# **SISTEM DE PROCESARE A POLINOAMELOR**

**-TEMA 1-**

**Nume: Bolduț Denisa-Teodora**

**Grupa: 30225**

# *Cuprins:*

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare (diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi)
4. Implementare si testare
5. Rezultate
6. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare
7. Bibliografie

## **Tema 1:**

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

### **1. Obiectivul temei**

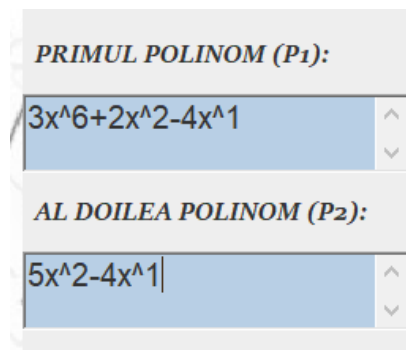
Obiectivul principal al acestei lucrari consta in realizarea unei aplicatii Java care sa ajute la calculul operatiilor pe polinoamele de o singura variabila cu coeficienti intregi.

Pentru realizarea acestui obiectiv, vin in ajutor cateva obiective secundare, descrise in aceste pagini, si anume:

- interpretarea polinomului introdus de utilizator
- realizarea operatiiei dorite:
  - adunare
  - scadere
  - inmultire
  - impartire
  - derivare
  - integrare
- generarea rezultatului si afisarea lui in chenarul corespunzator
- optiunea de stergere a datelor existente in fereastra si posibilitatea de resetare pentru a introduce alte date

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

O prima analiza efectuata asupra acestui proiect ar fi modul in care utilizatorul introduce datele, interpretarea lor si salvarea in tipuri de date corespunzatoare. Polinoamele trebuie introduse sub forma " $4x^5-3x^4+2x^1-3x^0$ ". In acest mod se specifica clar coeficientul si gradul fiecarui monom. O mica observatie ar fi faptul ca monoamele se introduc in ordine descrescatorie a gradelor pentru a forma polinomul dorit. Astfel introdus polinomul se porcedeaza la fel si pentru cel de al doilea in chenarele indicate:



PRIMUL POLINOM ( $P_1$ ):

$3x^6+2x^2-4x^1$

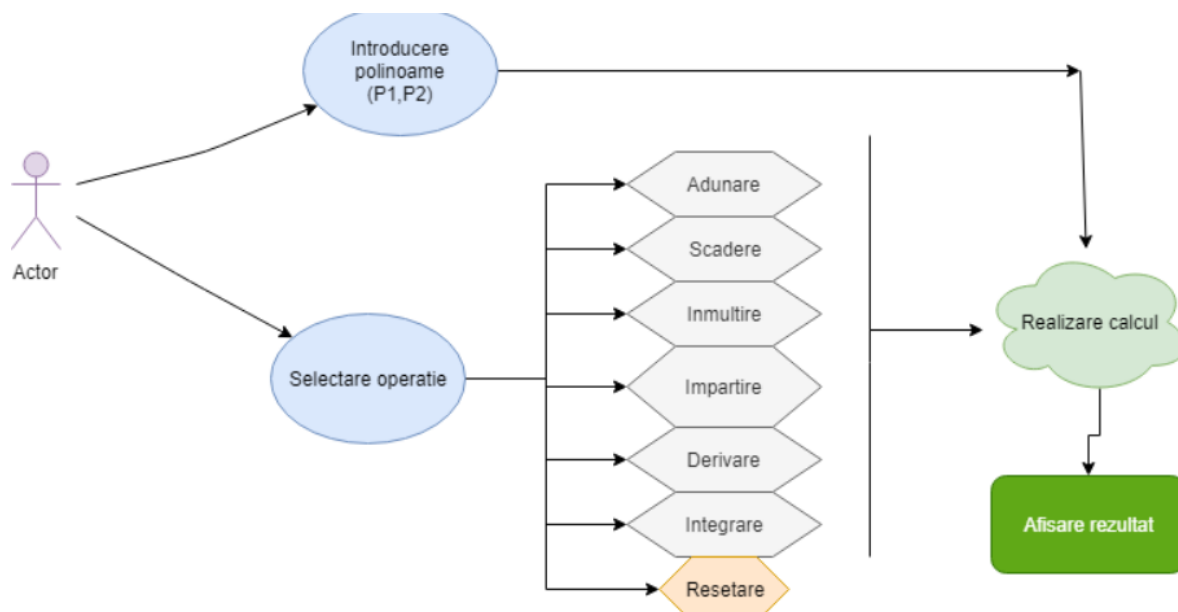
AL DOILEA POLINOM ( $P_2$ ):

$5x^2-4x^1$

Pentru interpretarea datelor introduse, respectiv stringul introdus de catre utilizator, este nevoie de spargerea acestuia pentru a stoca separat coeficientul si gradul fiecarui monom din cadrul polinomului corespunzator. Astfel avem nevoie de o clasa Polinom in care se va afla o lista cu monoamele specifice lui. Tinem cont si de faptul ca monoamele respective pot avea coeficientul atat intreg cat si real, mai exact in cazul realizarii operatiei de integrare. Astfel MonomIntreg si MonomReal sunt cele doua clase care vor extinde clasa parinte Monom.

In momentul de fata avem structurata interpretarea datelor introduse de catre utilizator. In continuare se asteapta ca utilizatorul sa isi aleaga o operatie. Aceste operatii sunt gestionate cu ajutorul butoanelor din interfata. In momentul in care un buton este apasat, si anume s-a ales operatia dorita, rezultatul corespunzator adunarii, scaderii, inmultirii sau impartirii va aparea in chenarul specific rezultatului. Pentru operatiile de derivare si integrare, rezultatul acestora are cate un chenar propriu. Motivul pentru care se intampla acest lucru este posibilitatea

de a alege dintre cele doua polinoame introduse asupra caruia se doreste efectuarea operatiei. Altfel spus, se poate integra si deriva ambele polinoame introduse, rezultatul putand fi observat in acelasi timp pentru fiecare operatie, respectiv polinom in parte. Polinomul rezultat va fi afisat sub aceasi forma ca cele introduse pentru a nu exista nicio ambiguitate sau neintelegere la nivel de vizualizare cu utilizatorul. Astfel, cand rezultatul poate fi observat, user-ul familiarizat fiind deja cu forma polinomului ii este usor sa inteleaga si sa proceseze datele rezultate.



Pentru o mai buna colaborare intre utilizator si autorul sau mai exact aplicatie, exista si butonul de resetare date. User-ul isi exprima astfel dorinta de a introduce un alt set de date. In momentul in care acest buton de stergere este apasat informatiile existente se sterg, semnaland locul in care utilizatorul ar trebui sa introduca noile date, putand sa aleaga din nou operatiile pe care doreste sa le realizeze.

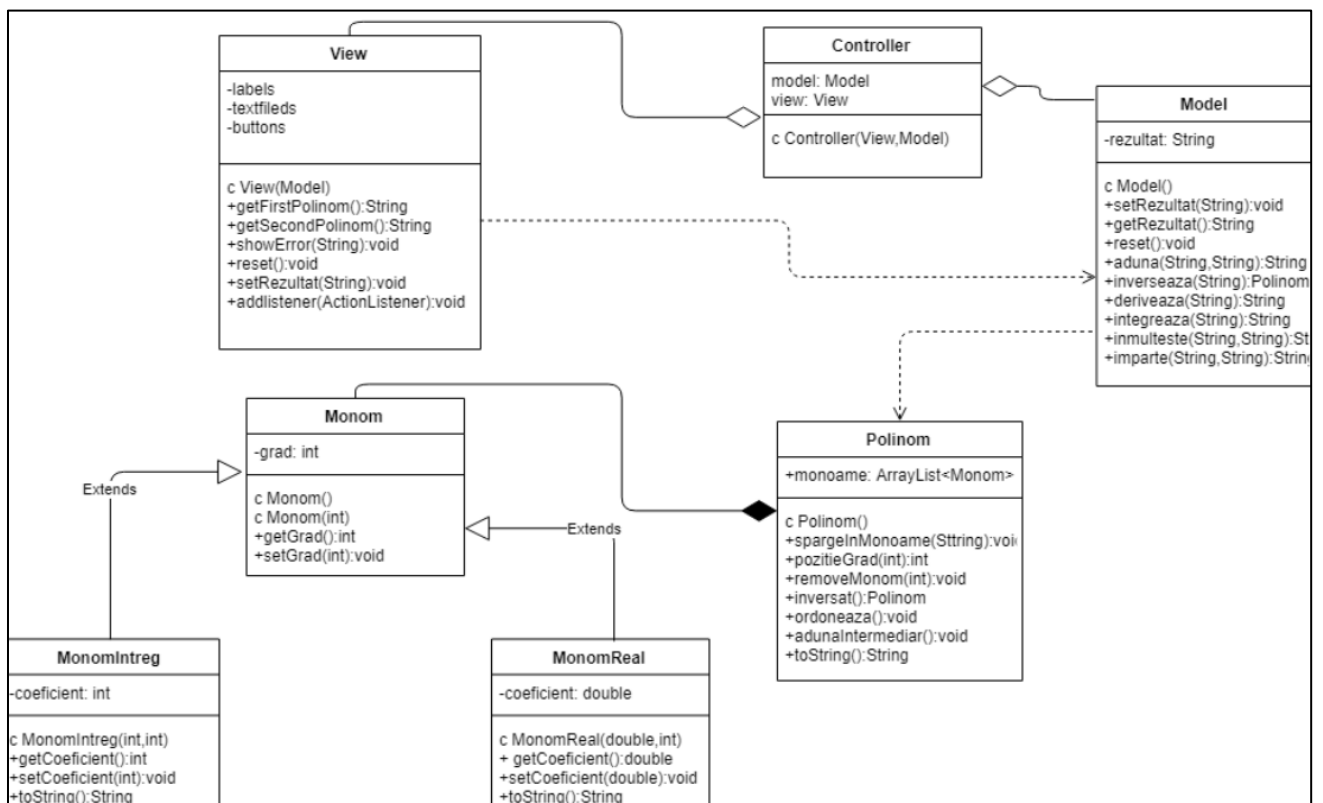
Algoritmii folositi pentru realizarea operatiilor care stau la indemana utilizatorului si la baza aplicatiei urmeaza sa fie explicati si amanunti in paginile urmatoare.

### 3.Proiectare

## Diagrame UML

-este un limbaj standard pentru descrierea de modele și specificații pentru software

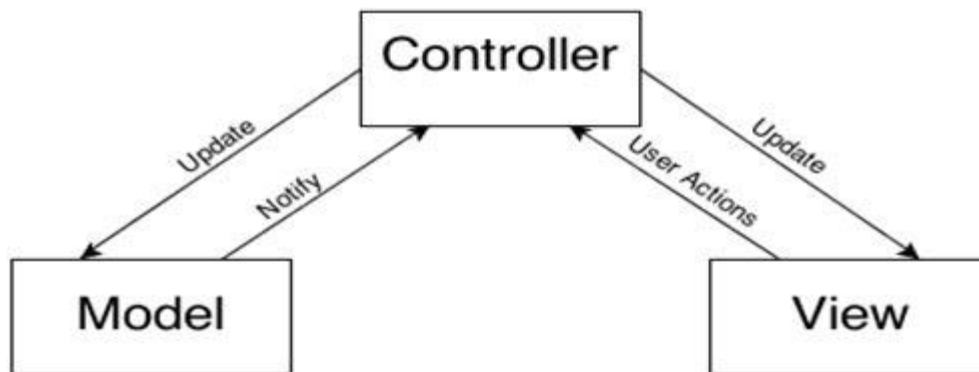
Modul in care este descris acest proiect este reprezentat de urmatoarea schema:



Astfel diagrama de clase este folosită pentru reprezentarea vizuală a claselor și a interdependențelor, taxonomiei și a relațiilor de multiplicitate dintre ele.

Diagramele de clasă sunt folosite și pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte, și a legăturilor concrete dintre acestea.

Structura proiectului este gandita sub modelul architectural **Model-view-controller (MVC)**:



## Structuri de date

Principala structura de date folosita in proiect est ArrayList-ul. Este folosita aceasta structura deoarece un ArrayList este o colecție de elemente indexate intr-o anumită ordine, dar nu neapărat sortate iar aceasta indexare permite accesul rapid la date, dar o insertie si stergere mai lenta. Clasa ArrayList este continuta in spatiul de nume System.Collections. Aceasta clasa ne permite sa construim un vector care sa creasca in marime, adaugand mereu elemente noi.

## Proiectare clase

Clasele care construiesc aplicatia sunt:

- MainClass
- Controller
- Model
- View
- Polinom
- Monom
- MonomIntreg
- MonomReal

In clasa MainClass se instantiaza Model, View, Controller si se realizeaza legaturile dintre acestea. Astfel datele introduse de catre utilizator ajung prin view la model cu ajutorul controller-ului. Model prelucreaza datele care i se dau, stie ce sa faca cu ele si genereaza rezultatul dorit. Controller-ul coordoneaza datele obtinute, le preia si le trimite clasei View pentru a le afisa.

In clasa Polinom avem lista de monoame. Implementarea ofera polinomului comportamentul de a sti sa se sparga in monoame, sa returneze pe ce pozitie se afla un anumit grad in lista de monoame ( o sa ne ajute asta la implementarea operatiilor), sa elimine un anumit monom specificat prin gradul sau, sa ordoneze lista de monoame crescator in functie de grad, sa schimbe semnele monoamelor( inmultire cu -1 care ne va ajuta la scadere) si sa suprascrie metoda toString pentru a genera stringul de afisat.

Clasa Monom are proprietatea un intreg in care se stocheaza gradul. La fel ca in fiecare clasa, avem un constructor, set & get, dar nu si coeficientul. Acesta din urma va fi stocat dupa caz in clasa MonomIntreg sau MonomReal. La unele operatii cum ar fi integrarea sau impartirea avem nevoie de coeficienti reali ai monoamelor pentru a putea reprezenta polinomul.

Clasele astfel obtinute constituie implementarea proiectului si duc la realizarea operatiilor cerute.

## Algoritmi

Algoritmii propriu-zisi pentru operatii sunt implementati in clasa Model. Acestia sunt coordonati mai departe de catre Controller care ii pune cap la cap pentru a realiza operatia ceruta.

### Adunarea

Algoritmul pentru realizarea adunarii consta in faptul ca se preiau polinoamele inca sub forma de stringuri si se sparg in monoame. Astfel avem create cele doua liste de monoame cu care putem gestiona mai usor gradele si coeficientii necesari operatiei. Parcurgem cu un for each prima lista de monoame si retinem gradul si coeficientul. Pasul urmator este de a verifica daca acelasi grad se gaseste si in cea de a doua lista pentru a le putea aduna coeficientii. Daca cumva nu se gaseste in cea de a doua lista atunci adaugam in polinomul rezultat monomul din primul polinom. In momentul in care gasim un grad de-al primului



polinom egal cu cel de-al doilea punem in rezultat un monom creat din suma coeficientilor si gradul respectiv. Eliminam din lista polinomului doi monomul pe care l-am folosit ca mai tarziu sa putem pune ce a ramas din polinomul doi direct in cel rezultat. Pe final comparam gradele monoamelor existente in rezultat cu cel pe care il introducem ca sa fim siguri ca acesta se introduce pe pozitia corecta ordonarii crescatoare in functie de grad.

### Scaderea

Pentru algoritmul de scadere ne folosim in marea masura de cel de la adunare. In model facem inmultirea celui de al doilea polinom cu -1. Controller-ul coordoneaza instructiunile astfel ca schimba semnele polinomului doi si il aduna cu primul polinom. Rezultatul obtinut este setat ca fiind rezultatul pe care vrem sa il afisam. Este trimis lui view care il afiseza in interfata grafica.

### Inmultirea

Primind ca parametrii cele doua stringuri care reprezinta cele doua polinoame, acestea se sparg in monoame pentru a genera interpretarea lor. Se parcurg cu for each ambele liste de monoame si se formeaza un nou monom care se adauga in lista polinomului rezultat. Monomului nou creat i se va seta coeficientul rezultatul obtinut prin inmultirea celor doi coeficienti existenti, iar grad este suma gradelor existente. Pentru a avea un rezultat conform gerintelor stabilite la inceput ordonom noua lista de monoame crescator in functie de gradul acestora. Pentru ca in urma algoritmului de inmultire s-ar putea sa obtinem la final doua monoame diferite dar cu acelasi grad, avem nevoie de o metoda care sa adune intr-un singur monom coeficienti acestora. Cu aceste aspecte prezentate se determina polinomul rezultat algoritmului de inmultire.

### Impartirea

Acest algoritm este gandit intr-un ciclu repetitiv, cat timp gradul deimpartitului intermediar este mai mic decat gradul impartitorului. Initial scadem gradele deimpartitului cu impartitorul pentru a vedea de cate ori se cuprinde impartitorul si le impartim coeficientii. Se construiesc un monom cu gradul si coeficientul obtinut anterior. Acesta se adauga in rezultat, iar rezultatul acesta intermediar se

inmulteste cu impartitorul. Ce se obtine urmeaza a se scadea din deimpartit, iar rezultatul obtinut din aceasta operatie devine deimpartitul intermediar. Ce vom optine la final in rezultatul pe care il actualizam la fiecare iteratii va reprezenta rezultatul impartirii. Ce va ramane in deimpartitul intermediar va reprezenta restul.

### Derivarea

Algoritmul de derivare se aplica pe unul dintre polinoamele alese. I se parcurge lista de monoame si la fiecare iteratie se creeaza un nou monom format din gradul monomului parcurs minus 1, iar coeficientul noului monom este produsul dintre gradul si coeficientul monomului parcurs. Urmand astfel formula matematica de derivare obtinem rezultatul dorit. In plus, se mai verifica cazul in care ajungem la un monom de grad 0, acesta ne mai avand rost sa il punem in lista rezultata. Polinomul rezultat este convertit cu ajutorul metodei toString() la un string si se returneaza. Acesta o sa fie preluat de Controller si trimis la View pentru a se afisa.

### Integrarea

Parametrul metodei de integrare este un string care se introduce de catre utilizator. Acesta este transformat in polinoame, respectiv monoame pentru a se putea manipula cu mai multa usurinta si eficienta. Se parcurge lista sa de monoame si la fiecare iteratie sa creeaza un nou monom rezultat din adunarea gradului monomului curent cu 1 si impartirea coeficientului sau cu (gradul +1). Acest monom obtinut se adauga in polinomul rezervat rezultatului, se convertește la string si este returnat.

## **Principii OOP**

In programarea procedurala, datele sunt descrise separat de prelucrari, astfel ca legatura dintre ele se realizeaza numai la nivel conceptual. Pentru tipurile de date primitive s-a adoptat conceptul, conform caruia prin tip de date se intelege o multime de valori, careia i se asociaza o multime de operatii care se pot efectua asupra valorilor respective. Constatam deci ca, inca de la inceputurile programarii in limbaje de nivel superior, s-a facut o legatura stransa intre date si operatii, legatura incorporata in fiecare limbaj de programare.

Un obiect care apartine unei clase se numeste si instanta a clasei (este o instantiere, o realizare particulara a clasei respective). De exemplu, clasa Barbat si clasa Femeie sunt subclase ale clasei Om. In schimb, Ion\_Popescu este o instanta a clasei Barbat (un obiect care apartine acestei clase), iar Maria\_Preda este o instanta a clasei Femeie.

Polimorfismul (engleza: polymorphism) permite ca aceeaasi operatie sa se realizeze in mod diferit in clase diferite. Sa consideram, de exemplu, ca in clasa Figura\_geometrica exista metoda arie(), care calculeaza aria figurii respective. Clasele Cerc, Triunghi, Patrat sunt subclase ale clasei Figuri\_geometrice si vor mosteni, deci, de la aceasta metoda arie(). Este insa evident ca aria cercului se calculeaza in alt mod decat aria patratului sau cea a triunghiului. Pentru fiecare din instancele acestor clase, la calcularea ariei se va aplica metoda specifica clasei sale.

Incapsularea (engleza: encapsulation) este proprietatea obiectelor de a-si ascunde o parte din date si metode. Din exteriorul obiectului sunt accesibile ("vizibile") numai datele si metodele publice. Putem deci sa ne imaginam obiectul ca fiind format din doua straturi.

Agregarea (engleza: aggregation) este proprietatea obiectelor de a putea incorpora alte obiecte. Asa dar, "datele" continute intr-un obiect pot fi nu numai date primitive, ci si obiecte. Se pot astfel crea obiecte cu structuri din ce in ce mai complexe.

Mostenirea (engleza: inheritance) este proprietatea unei clase de a contine toate attributele (variabilele) si metodele superclasei sale. In consecinta, trecerea de la clasa la subclasa se face prin adaugarea de attribute si/sau de metode. De exemplu, clasa Barbat si clasa Femeie au ambele toate attributele clasei Om, dar fiecare din acestea are si attribute specifice.

## 4. Implementare si testare

### Interfata cu utilizatorul

La rularea programului, utilizatorul este intampinat cu fereastra urmatoare:

The screenshot shows a window titled "Sistem de procesare a polinoamelor". It contains two input sections for polynomials, a set of operation buttons, and a result section.

PRIMUL POLINOM (P1):	Derivare(P1)	Integrare(P1)
$3x^6+2x^2-4x^1$	$+18x^5+4x^1-4x^0$	$+0,43x^7+0,67x^3-2,00x^2$

AL DOILEA POLINOM (P2):	Derivare(P2)	Integrare(P2)
$5x^2-4x^1$	$+10x^1-4x^0$	$+1,67x^3-2,00x^2$

Below the input sections are four buttons for operations:

- Adunare(P1+P2)
- Scadere(P1-P2)
- Inmultire(P1\*P2)
- Impartire(P1/P2)

The result section is labeled "POLINOMUL REZULTAT:" and contains a text box with the value  $+3x^6+7x^2-8x^1$ . To the right of this section is an orange button labeled "Stergere date".

In poza de mai sus este introdus si un exemplu pentru a se clarifica din nou modul in care aplicatia asteapta introducerea datelor. Polinomul se introduce sub forma „ $ax^b+cx^d-ex^f$ ”, variabila polinomului fiind x. Dintre cele patru operatii una sub alta se alege una pentru a se afisa rezultatul, timp in care derivarea si integrarea se pot face separat, pentru fiecare polinom. Datele vor fi sterse daca se apasa butonul portocaliu, afisandu-se instructiuni unde sa se introduca datele din nou. In momentul in care nu se respecta forma prestabila sau cel putin unul dintre polinome lipseste se va afisa o fereasta cu mesajul de eroare corespunzator:

PRIMUL POLINOM (P1):	Derivare(P1)	Integrare(P1)
<input type="text"/>	<input type="text"/>	<input type="text"/>
AL DOILEA POLINOM (P2):	Derivare(P2)	Integrare(P2)
3x^3+2x^0	<input type="text"/>	<input type="text"/>

Adunare(P1+P2)

Scadere(P1-P2)

Inmultire(P1\*P2)

Impartire(P1/P2)

POLINOMUL REZULTAT:

Stergere date

Message

**i** Primul polinom lipseste

OK

PRIMUL POLINOM (P1):	Derivare(P1)	Integrare(P1)
2(x)^6+4*x^4	<input type="text"/>	<input type="text"/>
AL DOILEA POLINOM (P2):	Derivare(P2)	Integrare(P2)
3x	<input type="text"/>	<input type="text"/>

Adunare(P1+P2)

Scadere(P1-P2)

Inmultire(P1\*P2)

Impartire(P1/P2)

Message

**i** Forma incorecta de introducere a datelor

OK

## 5. Rezultate

În urma algoritmilor implementați și folosind structura prezentată s-a obținut o aplicație care își însușește operațiile cerute. Sistemul de procese al polinoamelor este capabil să primească doi polinomi și să rezolve operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare.

## 6. Concluzii, ce s-a învățat din tema, dezvoltări ulterioare

În urma dezvoltării acestui proiect, am recapitulat conceptele de OOP, bineînțeles și operațiile pe polinoame. Ca primă temă, am învățat să lucrez mai bine cu interfața și aranjarea în panouri. Îmbunătățirile ulterioare care cred că s-ar putea aduce ar fi un aspect mai estetic și mai prietenos din punct de vedere al interfeței cu utilizatorul și adăugarea mai multor operații de efectuat pe polinoame.

## 7. Bibliografie

<https://sites.google.com/site/ursuleacv/knowledge/principiileprogramariiorientatepeobiectecestepoo/claseobiecteincapsulareamostenireapolimorfismul>

<http://andrei.clubcisco.ro/cursuri/2poo/co/Curs2.PDF>