MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

# CALCULATOR DE POLINOAME

Alin Matean

Facultatea de Automatica si Calculatoare

Grupa 30228

# Cuprins

# Obiectivul temei

Obiectivul acestei teme este proiectarea și implementarea unui calculator de polinoame. Acest calculator trebuie să dispună de o interfață grafică cu ajutorul unde utilizatorul sa introducă polinoamele dorite și să poată vedea rezultatul mai multor operații și anume: adunarea, scăderea, înmulțirea, impărțirea, integrarea și derivarea.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## Analiza problemei

În matematică, un polinom este o expresie construită dintr-una sau mai multe variabile și constante($PX=an*Xn+an-1*Xn-1+\cdots+a1*X+a0$), folosind doar operații de adunare, scădere, înmulțire și ridicare la putere constantă pozitivă întregă: $x^2 - 9x + 3$ este un polinom. Polinoamele sunt constituite din mai multe expresii numite monoame. Monomul este alcătuit dintr-o constantă înmulțită cu o variabilă, ridicată la o anumită putere. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile în acel monom. Un polinom construit cu o singură variabilă se numește univariant: $-3x^4$.

## Modelare

Modelarea problemei presupune interpretarea celor două polinoame și a operației alese, iar mai apoi furnizarea rezultatului așteptat și corect. Polinoamele introduse trebuie sa aibă coeficienți numere întregi și trebuie sa fie de o singură variabilă și anume:X/x.

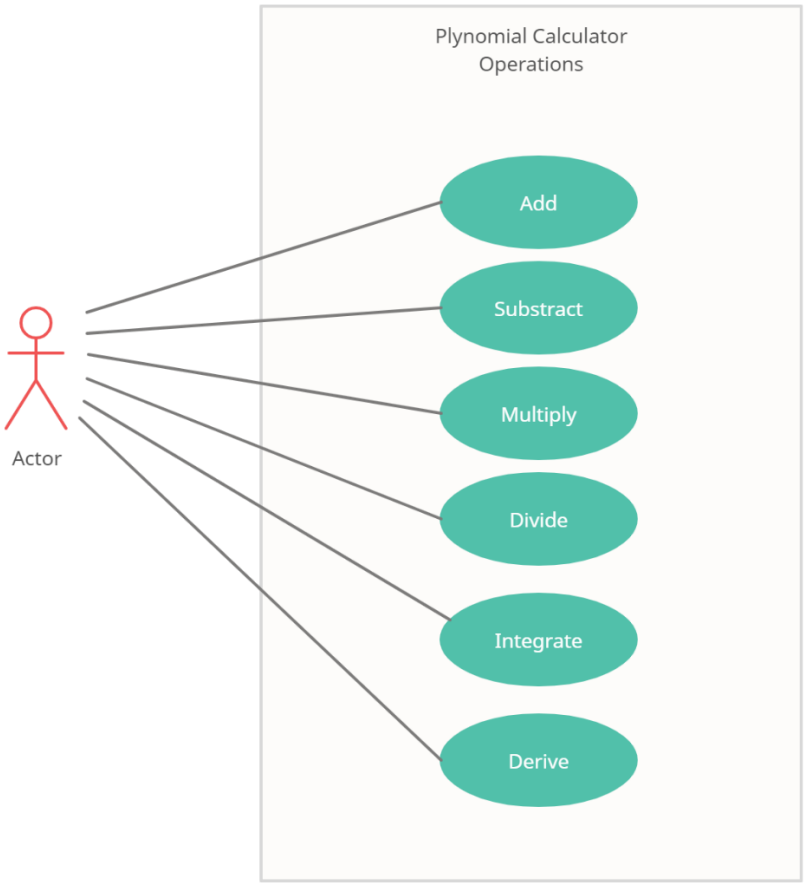
Operatia de adunare a polinoamelor presupune adunarea coeficientilor monoamelor care au acelasi exponent și formarea unui singur polinom. Operatia de scadere se realizeaza la fel, adică se scad coeficienții monoamelor din fiecare polinom care au același exponent. Operația de înmulțire presupune înmulțirea fiecărui monom din primul polinom, cu fiecare monom din al doilea polinom, iar mai apor reducerea termenilor asemenea, adica cu același grad. Împărțirea este o operație mai delicată, necesită mai mulți pași decat restul operațiilor, dar ca o verificare folosim formula ,,d = î*c + r", adica deîmpărțitul este egal cu împărțitorul ori câtul plus restul. În cazul integrării se aplica o împărțire la nivelul coeficientului, adica coeficientul inițial/exponent+1, și o adunare cu 1 la

nivelul exponentului. Operația inversă integrării este derivarea, care face o înmulțire intre exponent si coeficient setând astefl coeficientul și o scădere cu 1 in cazul exponentului.

# Scenarii

Scenariul de utilizare a aplicației presupune interacțiunea cu interfața grafică creată.Astfel, utilizatorul poate să introducă, in doua câmpuri rezervate, câte un polinom, iar mai apoi să aleagă operația care se va aplica pe cele două polinoame sau doar pe primul, cum e în cazul integrării și derivării. Pentru ca polinoul să fie valid, el trebuie să conțină doar constante(0-9), variabila x sau X(care va fi interpretată la fel) și exponentul introdus prin "^". Utilizatorul, dupa introducerea polinoamelor și alegerea operației, va primi într-un câmp rezervat, rezultatul calculului.

# Cazuri de utilizare – diagrama Use-case



Scenariul principal de utilizare cu succes al calculatorului:

-utilizatorul introduce polinoamele

-alege operația

-verifica rezultatul

-pentru a introduce polinoame noi, poate folosi butonul de „Clear”

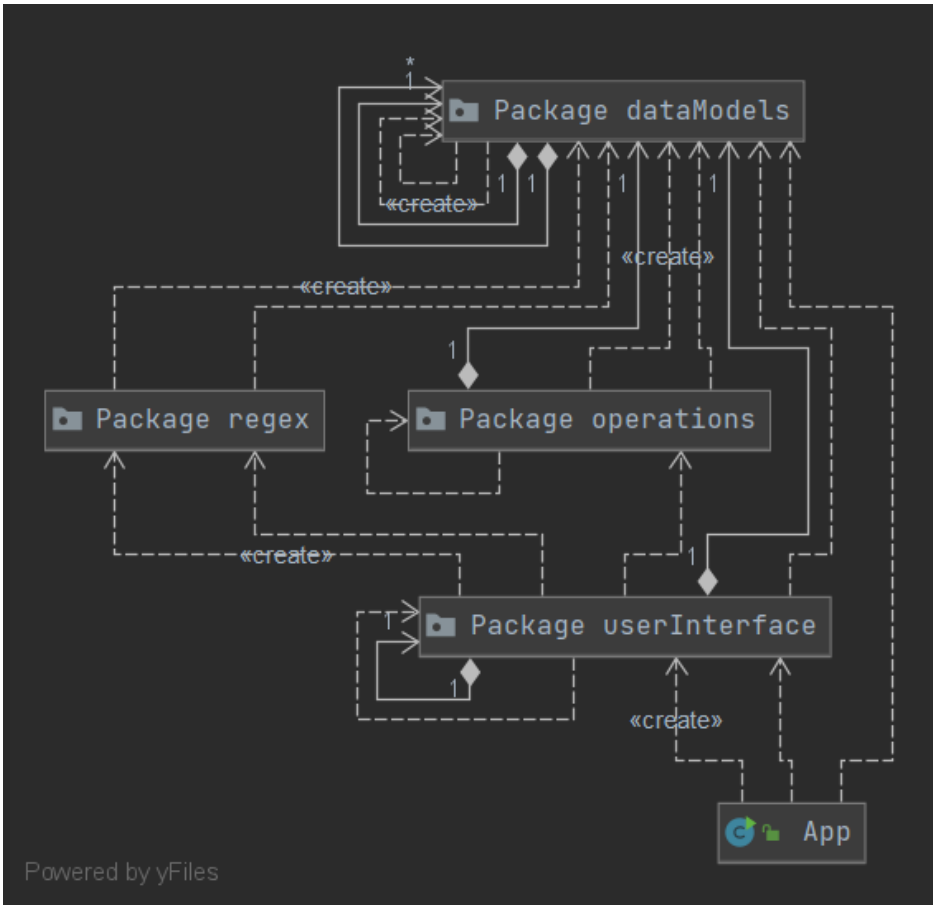
-operația de derivare și cea de integrare se vor aplica asupra primului polinom

În cazul în care unul din cele doua polinoame introduse conține și alte caractere pe langă cele prezetate mai sus, nu va fi validat și se va afișa un mesaj de eroare referitor la polinomul invalid intrdous. Astfel ne aflăm în scenariul care nu se va încheia cu succes.

# Proiectare

## Decizii de proiectare

Aplicația folosește tiparul de proiecatre MVC(Model - View - Controller). Acest model ne ajută să stabilim o ordine in cadrul pachetului principal care conține aplcația.




În cadrul pachetului “src” unde se află tot codul necesar funcționării și testării aplcației se găsesc două directoare “main” și “test”. În directorul main găsim folder-ul “java” care conține pachetul prinipal și anume “tuc.tp.tema1”, în care se află de fapt tot conținutul pentru funcționarea corectă a calculatorului.

Acest pachet conține la rândul lui alte 4 pachete și o clasă separata “App” unde se află metoda “main” care pornește aplicația. Cele 4 pachete sunt:

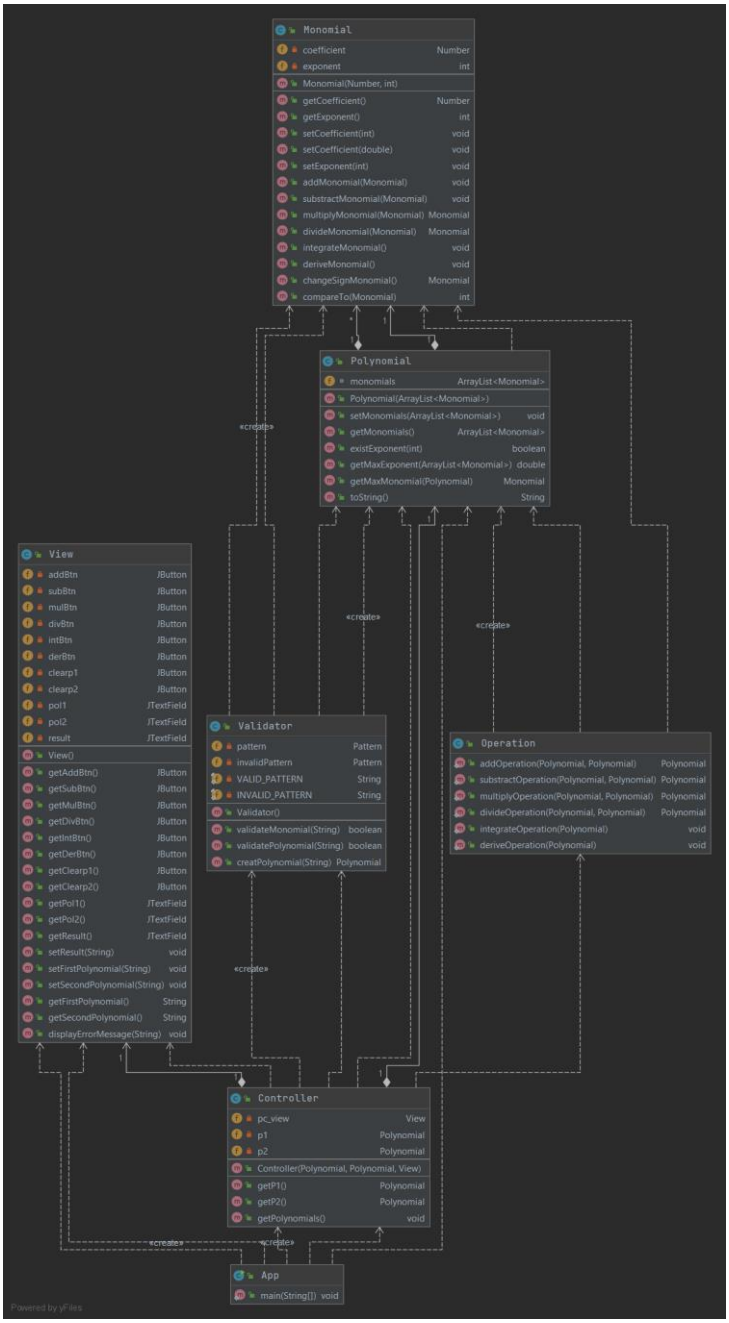
- dataModels care e constituit din două clase “Monomial” și “Polynomial” ce reprezintă modelul aplcației
- operations este constituit din clasa “Operation” unde sunt implementate operatiile pe modele
- regex este pachetul care conține clasa Validator necesar validării polinoamelor introduse
- userInterface conține clasele „Controller” si „View” folosite pentru realizarea interfeței grafice

În imagine poate observa diagrama UML de pachete a proiectului, diagrama pachetului “tuc.tp.tema1”.

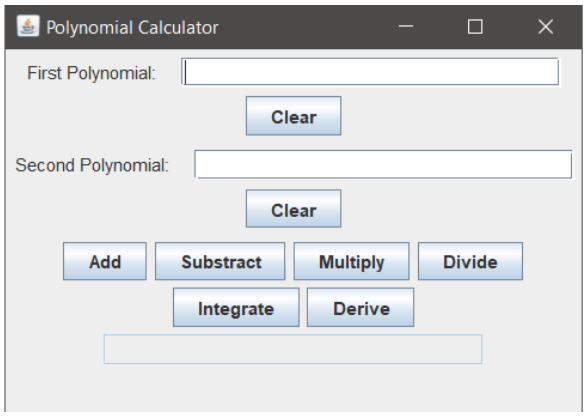
# Diagrama UML de clase

## Clasele proiectate:

- Monomial – pachetul dataModels
- Polynomial – pachetul dataModels
- Operation – pachetul Operations
- Validator – pachetul regex
- Controller – pachetul userInterface
- View – pachetul userInterface
- App – nu este inclusă in alt pachet separat, doar in “tuc.tp.tema1”, care le conține pe toate



## Interfața grafică cu utilizatorul




Utilizatorul va introduce în primele doua câmpuri câte o expresie polinomială, va alege operația și va primi rezultatul in câmpul de jos. Polinoamele introduse ar trebui să fie de forma: aX^n + bX^(n-1) + …+zX^0, unde a, b,…,z sunt constante întregi, iar n este un întreg pozitiv. În cadrul introducerii polinoamelor, pot să lipsească X^0 si să apară doar constanta, iar în loc de X^1 să se scrie doar X, cum de altfel vor aparea și la rezultat. Operația de integrare și operația de deriare se vor aplica asupra primului polinom.

# Implementare

Aplicația "Polynomial Calculator" este dezvoltată folosită limbajul de programare Java, utilizând paradigma de programare MVC(Model – View - Controller). Mai jos voi prezenta fiecare clasa folosită, cu metodele corespunzătoare.

## Clase

### Pachetul dataModels:

- Monomial
  - în această clasă avem două varibie instanță corespunzatoare fiecărui obiect nou instanțiat și anume *coefficient* de tipul Number și *exponent* de tipul int.
  - constructorul acestei clase instanțiază un nou obiect cât timp *coefficient* e diferit de 0, nu ar avea sens sa se creeze un Monom(0, 0).
  - Avem un getter și un setter pentru *exponent* și 2 getteri și un setter pentru *coefficient,* asta deoarece uneori de dorim valoare de tipul int, alteori cea de tipul double.
  - Avem 6 metode care implementează cele 6 operații cerute: addMonomial adună doua monoame dacă acestea au același exponent, același lucru de întâmplă și la substractMonomial, dar evident se scad. Metoda multiplyMonomial realizează înmulțirea monoamelor la nivelul coeficienților și adunarea la nivelul exponenților, operația inversă fiind divideMonomial.

IntegrateMonomial și deriveMonomial fac operația de integrare și derivare corespunzătoare din matematică.

- Am mai implementat o metodă changeSignMonomial, folositoare ulterior, care practic schimbă semnul polinomului.
- Deoarece clasa Monomial implementează interfața *Comparable,* am suprascris metoda compareTo, astefel încât să le pot ordona ulterior în ordine descrescătoare a exponenților(util la afișarea polinoamelor)

➢ Polynomial

- Un polinom este reprezentat ca o listă de monoame -> *ArrayList* în implementare. Astfel constructorul calsei inițializează variabila instanță monomials și formează astfel polinomul ca o lista de monoame
- Am facut o metodă *toString* pentru a afișa polinomul ca o expresie(ca in matematică), am avut grija să se afișeze ca și double in cazul în care s-a efectuat o împărțire a coeficienților și rezultatul nu a fost de tip întreg.
- Metoda *existExponent* verifică daca în lista obiectului cu care este chemată metoda există exponentul trimis ca și parametru -> se va evalua la true/false
- Metoda *getMaxExponent* ne returneaza exponentul cel mai mare dîntr-un polinom
- Metoda *getMaxMonomials* folosește cele două metode de mai sus pentru a extrage monomul de grad maxim, util în operația de împărțire

## Pachetul operations

➢ Operation

- Clasa Operation este o clasă mai special, conține doar metode statice și nu conține variabile obiect. Aici sunt implementate toate operațiile
- Prima metoda *addOperation* primește ca argumente două polinoame și returneaza un rezultat de același tip: Polynomial. Ca și implementare: parcurg lista de monoame a fiecărui polinom și aplic adunarea de la monom, care se va realiza doar în cazul în care cele două monoame la care ne aflăm la un moment dar în parcurgere au același grad. Rezultatul acestei adunări se va stoca în primul polinom. Astfel acum primul polinom conține monoamele inițiale adunate cu cele care au același grad din polinomul2. Aceste monoame le stocăm într-o nouă listă, pe care o parurgem și la care trebuie s adăugam eventualele monoame din polinomul2 care nu aveau un corespondent ca și grad în polinomul1 și nu s-au adunat. Lista nou rezultată conține acum toate monoamele celor două adunate. Returnăm un nou polinom format din această listă.
- Metoda *substractOperation* este similară cu cea prezentată mai sus, aici se scad monoamele din primul polinom cu monoamele din al doilea de același grad. Dupa ce am realizat operația trebuie să verificam daca în polinomul2 au rămas monoame ce nu aveau corespondent și să le adăugăm în lista nouă, dar cu semn schimbat! La final cu ajutorul unui iterator parcurgem această listă și verificăm dacă nu cumva avem polinoame care s-au redus și au ajuns să aiba coeficientul 0, pe care le eliminăm ulterior.
- Metoda *multiplyOperation* la fel returnează un nou polinom rezultat din înmulțirea celor două primite ca și argumente. Parcurgem listele monoamelor și de data aceasta înmulțim fiecare monom din prima lista cu fiecare din a doua. Adăugăm fiecare nou monom într-o listă pe care o parcurgem la fiecare pas, dacă găsim un nou monom cu același grad adunăm acest monom nou rezultat, dacă nu îl adăugăm ca nou element în listă. La final returnăm un nou polinom
- Metoda *divideOperation* este cea mai complexă. Ne salvăm inițial cele două polinoame în variabile auxiliare. Sortăm listele din fiecare poliom astfel încât să obținem un polinom sortat

descrescător în funcție de exponent. Prima dată verificăm dacă cele două polinoame au un singur monom, iar dacă e așa facem doar o impartire a celor două monoame și returnăm rezultatul de tip polinom cu un singur termen. Dacă polinoamele sunt expresii mai complexe, la fiecare pas verificăm gradul primului polinom să fie mai mare sau egal cu gradul celui de-al doilea, facem împărțirea monomului de grad maxim din primul polinom cu monomul de grad maxim din al doilea, rezultatul îl adăugăm la lista finală, dar îl înmulțim și cu al doilea polinom pentru a genera descăzutul. Facem scăderea dintre primul polinom și descăzut, iar rezultatul va deveni noul deîmpărțit, adica va deveni polinomul1, dupa care reluăm pașii. Când nu mai este satisfăcută condiția inițială dintre gradul deîmpărțitului și cel al împărțitorului, am terminat împărțirea și polinomul cu care am rămas va reprezenta restul, pe care eu am ales să-l pun în polynomial1, adică să-l suprascriu. Rezultatul va fi un nou polinom format din lista la care am adăugat pe rând câte un monom rezultat din împărțirea monoamelor de grad maxim.

- Metoda *integrateOperation* este scurtă și parcurge lista de monoame a polinomului, la fiecare pas realizează integrarea monomului, după implementarea din clasa Monomial.
- Metoda *deriveOperation* parcurge lista monomului și se calculează derivata fiecărui monom conform implementării din clasa respectivă, după cu ajutorul unui iterator vom elimina monoamele care au ajuns eventual la exponent negativ.

## Pachetul regex

- Validator
  - Această clasă se ocupă cu validarea datelor introduse de utilizator folosind pachetele din java.util: regex.Pattern si regex.Matcher.
  - Avem doua constante declarate static final, una cu un Pattern valid și alta cu Pattern invalid.
  - În constrcutorul clasei compilăm punem rezultatul compilării acestor pattern-uri în două variabile obiect.
  - In metoda booleana returnăm validateMonomial returnăm true dacă s-a găsit un pattern-match valid, altfel false
  - Metoda validatePolynomial face validarea polinomului în funcție de validarea fiecărui monom
  - Metoda createPolynomial este cea care ne creeaza efectiv polinoamele, în funcție de verificarea grupurilor componente de exemplu dacă se intrduce “x^3-2x^2”, sau doar “x” sau doar “3”, astfel verific fiecare tip de monom și creez în final un polinom. Pentru ca polinomul dat să fie creat cu succes trebuie sa nu existe alte caratere în afară de: “x,X.orice cifră și ^”.Mai mult de atât, pentru ca rezultatul să fie cel așteptat se recomandă introducerea unui polinom care să aibă sens, cu exponent numar mai mare decât zero și coeficient întreg.

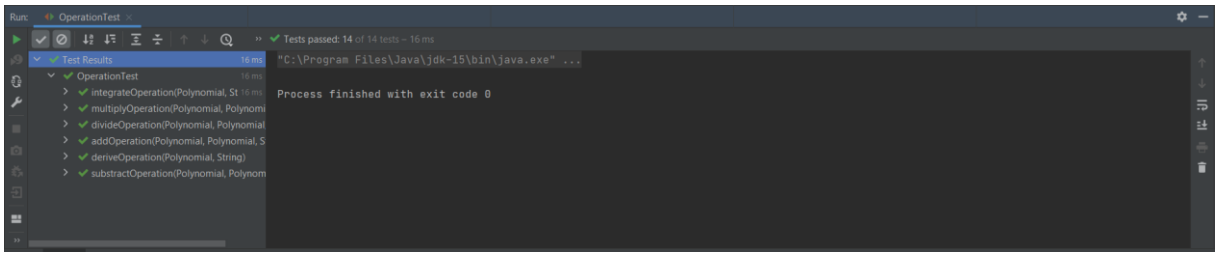
## Pachetul userInterface

- Controller
  - Constructorul aceste clase inițializează cele două polinoame și view-ul
  - Tot aici, adaug cate un ActionListener pentru fiecare buton creat în clasa View și cu care va avea de interacționat utilizatorul. La fiecare buton, îmi iau polinoamele dacă au fost validate cu success, realizez operația dorită și setez rezultatul în câmpul result
  - Metoda *getPolynomials()* ia textul introdus de utilizator și le validează ca mai apoi să continue utilizarea calculatorului.
- View

- Această este partea vizuală a aplicației, aici am declarat butoanele, câmpurile și etichetele de care va avea nevoie utilizatorul
- În constructorul clasei, creez frame-ul necesar, adaug toate componentele și cel mai important, setez conținutul ca fiind vizibil.
- Am mai declrat metode get si set utilizate in clasa Controller, pentru a putea avea access la butoane și la câmpuri.

# Rezultate

Cu ajutorul Junit am testat aplcația. Am testat operațiile din clasa monom și operațiile cu polinoame implementate în clasa Operations.


Clasele MonomialTest și OperationTest se află în folder-ul de test al proiectului. MonomialTest conține teste simple @Test, iar OperationTest conține teste parametrizate. Testele sunt verificate cu ajutorul aserțiunilor.

# Concluzii

## Dezvoltări ulterioare

Acest calculator momentan poate realiza 6 operații pe polinoame cu exponent întreg și coeficient întreg

Posibile dezvoltări ulterioare ar fi:

- Realizarea operațiilor cu polinoame de mai multe variabile, nu doar de una

- Implementarea unei metode care să afle rădăcinile polinomului
- Calcularea derivatelor parțiale, dacă vom implementa polinoame cu mai multe variabile
- Calcularea valorii unui polinom într-un anumit punct
- Implementarea derivatelor de ordin mai mare de 1

## Concluzii

Cu ajutorul acestui proiect, am aprofundat modelul architectural MVC. Am învățat să folosesc pattern matching, Regex mai exact, utilizat pentru prima dată de mine. Am înțeles mai bine cum se realizează structurarea proiectului în pachete. De asemenea, am învățat să utilizez testele parametrizate, foarte utile de altfel.

# Bibliografie

https://users.utcluj.ro/~igiosan/teaching_poo.html

https://regex101.com/

https://app.creately.com/

https://www.youtube.com/watch?v=rhzKDrUiJVk

https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression