TEHNICI DE PROGRAMARE

TEMA 1

## Student : Belascu Dorin

## Grupa : 302210

## Profesor supervizor : Teodor Petrican

Cuprins:

1. **Obiectivul temei ..............................................................................3**
2. **Analiza problemei, modelare, scenarii, cazuri de utilizare.................4**
   1. Analiza problemei.......................................................................4
   2. Modelare....................................................................................5
   3. Scenarii, Cazuri de utilizare.........................................................5
3. **Proiectare.........................................................................................6**
4. Diagrama UML............................................................................6
5. Designul Clasei............................................................................7
6. Algoritmi....................................................................................10
7. Interfata Utilizator.....................................................................12
8. **Implementare si testare...................................................................13**
9. **Rezultate, concluzii, dezvoltari ulterioare.........................................14**
10. Lucruri noi invatate......................................................................14
11. Dezvoltari ulterioare....................................................................15
12. **Bibliografie.......................................................................................16**

# Obiectivul temei

**(RO) Laborator – Tema 1**

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

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## 2.1 Analiza problemei

In matematica, un polinom este o expresie construita dintr-una sau mai multe variabile si constante, folosind doar operatii de adunare, scadere, inmultire si ridicare la putere constanta pozitiva intreaga.

Polinoamele sunt construite din termeni numiti monoame care sunt alcatuite dintr-o constanta (numita coeficient) inmultita cu una sau mai multe variabile. Fiecare variabila poate avea un exponent constant intreg pozitiv. Exponentul unei variabile dintr-un monom este egal cu gradul acelei variabile in acel monom. Coeficientul unui monom poate fi numar intreg sau numar fractionar.

Calcularea polinoamelor este foarte utila in multe domenii(economie, inginerie, statistica, etc). O interfata usor de utilizat chiar si de cei care nu au cunostiinte in cadrul calculatoarelor este de mare ajutor in marirea productivitatii.

Am ales sa implementez unele dintre cele mai des utilizate operatii polinomiale: adunare, scadere, inmultire, impartire, derivare si integrare.

## 2.2 Modelare

Principala entitate de modelat este evident Polinomul. Clasa Polinom include pe langa un vector de Monoame(fiecare monom fiind caracterizat de un coeficient real si de o putere intreaga) toate operatiile care se pot face intre doua polinoame, ele fiind strict legate de entitatea Polinom. Fiecare dintre operatii returneaza un nou polinom, polinom rezultat. Pentru partea de GUI s-a folosit JavaFX impreuna cu Scene Builder creandu-se astfel toate butoanele, campurile de text de care a fost nevoie pentru o folosire usoara si placuta. Cat despre testare s-a creat un Junit Test Case ce cuprinde un test pentru fiecare dintre operatiile prezentate.

## 2.3 Scenarii, Cazuri de utilizare

Aplicatia prezentata incarca doua polinoame din oficiu la deschidere dar permite si utilizatorului sa introduca de la tastatura 2 polinoame si apoi posibilitatea de a alege ce operatie sa se execute.

Daca de exemplu polinoamele introduse sunt :

P1 : 12x^4-6x^3+2x+7

P2 : 3x^2+2

Rezultatele vor fi :

1. Adunarea : +9.00x^0 +2.00x^1 +3.00x^2 -6.00x^3 +12.00x^4
2. Scaderea : -5.00x^0 -2.00x^1 +3.00x^2 +6.00x^3 -12.00x^4
3. Impartire : Cat: -2.67x^0 -2.00x^1 +4.00x^2

Rest: +12.33x^0 +6.00x^1

1. Inmultire: +14.00x^0 +4.00x^1 +21.00x^2 -6.00x^3 +24.00x^4 -18.00x^5 +36.00x^6
2. Derivare (P1) : +2.00x^0 -18.00x^2 +48.00x^3
3. Integrare(P1) : +7.00x^1 +1.00x^2 -1.50x^4 +2.40x^5

Se pot introduce orice polinoame care respecta formatul ca si in exemplul de la deschiderea aplicatiei. Polinomul este construit automat cu ajutorul clasei java Regex. Orice polinom rezultat poate fi folosit ca polinom de intrare. In cadrul casutei ce este folosita pentru rest, valoarea este pastrata pana la introducerea altor polinoame la intrare pentru a se face impartirea.

# Proiectare

## 3.1 Diagrame UML

## 3.2 Designul Claselor

Exista doar 2 clase care se ocupa de partea de functionalitate : Polinom si Monom. Restul claselor au rolul de a testa(OperatiiTest.java) sau de a oferi utilizatorului o modalitate cat mai placuta de utilizare(TestController.java, Main.java, TestController.java ).De asemenea apare si fisierul test.fxml scris cu ajutorul aplicatiei Scene Builder si este responsabil pentru descrierea fizica a interfetei.

**a) Clasa Polinom**

Atribute :

Exista un singur atribut, un ArrayList ce contine toate monoamele componente in ordine crescatoare dupa putere, ordine data de modul de adaugare a unui nou monom. Atributul este private, astfel realizandu-se incapsularea.

Constructori :

Exista un singur constructor, fara parametrii, care nu face decat sa instantieze un nou obiect ce va detine un nou ArrayList.

Metode :

Ar fi de amintit cateva metode de tip get care ajuta la mentinerea incapsularii si ofera informatii legate de starea listei de monoame. GetMonom(int i) -> returneaza monomul de pe pozitia i, getLungimePolinom() -> returneaza numarul de monoame din polinom(utilizat pentru parcurgeri) , getGradPolinom() -> folosit in cadrul impartirii, pentru comparari de grade.

Exista si metode de populare ale listei de monoame(in cazul in care se incearca adaugarea unui monom a carui putere deja exista se va face adunare intre coeficienti ) : adaugaMonom() -> adauga un nou monom listei, preiaRegex(String s) -> primeste ca si parametru un string ce este preluat

folosind clasa java Regex astfel incat sa se potriveasca cu sablonul de stocare al polinomului.

Tot aici se gasesc si toate metodele raspunzatoare pentru executia operatiilor dorite. Acestea sunt :

- aduna(Polinom p, boolean completeazaAdaugarea) -> polinomul curent se aduna la polinomul primit ca si parametru si se returneaza un nou polinom egal cu rezultatul adunarii. Variabila completeazaAdunarea are rolul de a face diferenta intre cei 2 pasi ai adunarii(in primul pas-false se aduna monoamele cu puteri comune si cele care exista doar in polinomul gazda iar in al doilea pas-true se aduna si monoamele care nu sunt comune si se afla doar in polinomul primit ca si parametru).

- scade (Polinom p1) -> Se scade din polinomul curent polinomul primit ca si parametru.Pentru aceasta operatie se foloseste adunarea cu minus polinomului primit ca si parametru.

- inmulteste(Polinom p) -> se returneaza un nou polinom egal cu inmultirea polinomului curent si cel primit ca si parametru.

-imparte(Polinom p) -> se imparte polinomul cu grad mai mare la polinomul cu grad mai mic.In cazul in care polinoamele sunt egale se imparte polinomul curent la polinomul primit ca si parametru.Se returneaza doua polinoame noi unul fiind egal cu restul si unul cu catul

- deriveaza(int n) -> se deriveaza de n ori polinomul curent dupa care se curata toate monoamele ale caror coeficienti sunt 0 sau puterea mai mica decat 0(curataPolinom());

- integreaza(int n) -> se integreaza de n ori polinomul curent

Amintim si metodele private care se ocupa de modularizarea functionarii celorlalte metode, de „bucataria programului”. Operatii de baza la nivel de monom : adunaMonom, inmultesteMonom, imparteMonom. Verificarea existentei unui monom si adaugarea in mod sortat au aplicatii in cadrul adaugarii unui nou monom.

Apar si 2 metode de afisare, afiseaza() si functia toString() folosita pentru formarea unui String cu continutul listei pentru umplerea zonelor de text din interfata.

**b) Clasa Monom**

Atribute :

Aici se memoreaza informatiile cu referire la fiecare dintre monoame. Coeficientul poate sa fie orice numar real(tip float) iar puterea numar intreg. Ambele sunt de tip private astfel incurajandu-se incapsularea.

Constructori :

Exista un singur constructor cu 2 parametrii, pentru atributele private ale clasei. Puterea este preluata si stocata ca si double dar la afisare se transforma in intreg.

Metode :

Apar metode de tip get si set pentru modificarea atributelor. Pentru afisare apare metoda toString() care foloseste metoda statica format() a clasei String pentru afisarea coeficientului doar cu doua zecimale.

**c) Clasa Main**

Aici se lanseaza partea de GUI incarcandu-se scena pentru interpretarea fxml-ului. Tot aici se seteaza si standardul raspunzator pentru afisarea float-ului cu punct, nu cu virgula.

**d) Clasa TestController**

In aceasta clasa se preiau evenimentele ce apar in interfata si in functie de ce este selectat de la buton se preiau polinoamele introduse in campul de text, sunt transformate si stocate in variabile de tip polinom ce sunt create si se executa operatia selecata.

Atribute : Atribute folosite pentru logica de prelucrare a informatiilor : polinom1, polinom2, polinomCat, polinomRest, rezultatImpartire si atribute folosite pentru a se face legatura cu fisierul fxml : textFieldP1, textFieldP2, textFieldCat, textFieldRest(pentru fiecare dintre acestea exista un element fxml cu fx:id-ul cu denumire identica cu numele atributului in clasa).

Metode : Apar metode pentru fiecare dintre operatiile ce se pot face.(adunare, scadere, inmultire, impartire, derivare, integrare) si o metoda executaOperatie ce se ocupa de gestiunea operatiilor folosind cat mai putin cod.

**e) Clasa TestOperatii**

Este un fisier Junit Test Case si contine cate un test pentru fiecare dintre operatii. Pentru operatii se folosesc polinoamele ce apar din oficiu la deschiderea aplicatiei.

## 3.3. Algoritmi

**Preluare Regex**

Una dintre cele mai dificile parti ale implementarii a fost preluarea stringului si stocarea lui ca si polinom utilizand clasa java Regex astfel inca introdusul polinomului sa fie cat mai flexibil. Expresia formatului pentru preluare este : ([+-]?\\d\*\\.?\\d\*)[xX](\\^(\\d+))?|([+-]?\\d+\\.?\\d\*);

Fiecare dintre parantezele rotunde reprezinta o anumita grupare a carei valoarea va putea fi preluata separat. In stanga parantezelor patrate se gaseste descrierea zonei de semn (plus sau minus) si coeficientul care poate fi separat de punct, fiind de tip float. Daca exista semnul „^” trebuie neaparat sa existe si „x” sau „X”,lucru sugerat de parantezele patrate. „|” reprezinta sau-ul din logica booleana si cu ajutorul lui se trateaza cazul in care avem doar termenul liber, fara x la o putere anume. Se recunoaste doar o singura variabila pentru polinom, aceasta fiind „x” sau „X”. O varianta corecta de introducere a unui polinom este: 12x^4-6x^3+2x+7. Tot aici exista si o parte de if-else-uri care, in functie de existenta sau nu a unei grupari preiau corespunzator.

**Impartirea**

Metoda de impartire primeste ca si parametru un al doilea polinom care poate fi atat deimpartit cat si impartitor in functie de gradul lui fata de gradul curent. In cazul in care gradul polinomului primit este mai mare decat gradul polinomului curent se interschimba astfel incat mereu se va imparti un polinom cu grad mai mare la un polinom cu grad mai mic. Algoritmul are la baza metoda clasica de impartire a doua polinoame. Se imparte monomul dominant al deimpartitului la monomul dominant al impartitorului si se memoreaza rezultatul. Rezultatul memorat se inmulteste cu impartitorul pentru ca mai apoi noua valoare sa fie scazuta din impartit. Catul va fi rezultatul sumei monoamelor rezutate dupa impartirea monoamelor dominante. Restul va fi egal cu valoarea deimpartitului dupa executarea impartirilor. Se va returna un vector de 2 Polinoame pentru cat si rest.

**Adunarea**

Fiind implementat in interiorul clasei polinom, algoritmul returneaza un nou polinom egal cu suma dintre polinomul curent si cel primit ca si parametru.

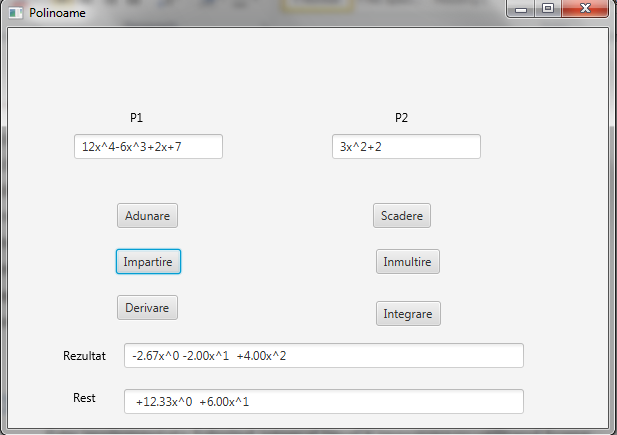
Dupa modul de implementare sunt 2 pasi a adunarii folosinduse un apel recursiv al functiei si o variabila de tip boolean(completeazaAdunarea) pentru a se decide care pas al adunarii este.

In primul pas se selecteaza toate monoamele din polinomul gazda si in cazul existentei unui alt monom cu putere identica in polinomul primit ca si parametru se aduna coeficientii acestora,in cazul in care monomul gazda nu exista si in polinomul parametru se adauga asa cum este.

Al doilea pas consta in adaugarea monoamelor existente doar in polinomul parametru la rezultat.

Dupa finalizarea celui de al doilea pas se returneaza polinomul rezultat.

## 3.4 Interfata utilizator

 Este implementata folosind integral JavaFX impreuna cu utilitarul Scene Builder pentru generarea fisierului fxml in urma unor operatii de tip „drag an drop”.

Contine 6 butoane pentru fiecare operatie in parte si 4 campuri de text(P1,P2 -> polinoamele introduse, Rezultat -> rezultatul dupa efectuarea operatiilor,catul in cazul impartirii, Rest -> restul impartirii).

La apasarea oricarui buton se vor prelua polinoamele din campurile P1 si P2 si se va aplica asupra lor operatia selectata rezultatul fiind afisat in campul Rezultat.

De precizat ca orice polinom produs la iesire poate fi folosit ca si polinom de intrare. Despre functionalitate se gasesc mai multe in cadrul capitolului 3.Proiectare in subcapitolul 3.2 Designul Claselor.

# Implementare si Testare

Primul si al doilea polinom trebuie introdusi ca si o suma de monoame, care sa respecte urmatoarea forma :

semn + coeficient + variabila + “^” + putere

* Semn : + sau –
* Coeficient : orice numar real (ex : 2.3, 1.43)
* Variabila : trebuie mereu sa fie „x” sau „X”
* Puterea : poate sa fie orice numar intreg pozitiv(ex 4, 5, 52)

Desi aplicatia recunoaste diferite tipuri de specificare a unui polinom de intrare, exista totusi anumite reguli de respectat :

* In cazul introducerii unui polinom vid pentru inmultire se va returna polinomul vid ca si rezultat iar in cazul impartirii se va returna polinomul mai mare(deimpartitul -> la impartire mereu se imparte polinomul mai mare la polinomul mai mic)
* Daca se introduc pentru putere numere intregi negative ele vor fi preluate ca si numere intregi pozitive
* Desi spatiul alocat in interfata pentru introducerea polinoamelor este mai mult decat suficient exista o limita in special in cadrul inmultirii, doua polinoame foarte mari pot returna un polinom suficient de mare incat sa nu incapa in campul de text pentru rezultat.
* Puterea va fi introdusa doar folosind simbolul „^”, de exemplu x la patrat va fi : x^2, scrierea puteriilor 0 sau 1 este optionala.
* Nu se permite introducerea oricarui alt exceptand cifrele pentru coeficient si putere, x sau „X” si semnul „^” pentru putere.

Cateva exemple de polinoame valide:

3x + 5x^2 – 9x^3

2x^9 + 1x^9 – 3x^1 +4.4x – 5

4.2x^2 – 3x^4 + 3.5x^2

Se observa ca ordinea de introducere nu este importanta(din punct de vedere a puteriilor monoamelor).De asemenea in cazul introducerii in cadrul aceluiasi polinom a doua monoame cu grad identic se face adunarea.

Aplicatia a fost testata pe o gama larga de exemple pentru a asigura buna functionare la general. Exista si un fisier Junit test case unde se gasesc teste pentru fiecare dintre operatii.

# Rezultate, concluzii, dezvoltari ulterioare

Dupa ducerea la capat a aceste aplicatii mi-am dat seama ca si daca este vorba de o problema relativ usor de rezolvat exista o multime de cazuri de luat in considerare, cazuri ce pot sa produca o multime de probleme care nu sunt atat de evidente la inceput.

## 5.1 Lucruri noi invatate

Am invatat cum sa folosesc JavaFX impreuna cu Scene Builder, un designer java de GUI foarte puternic si usor de utilizat. Folosirea lui mi-a usurat foarte mult munca pentru partea de interfata fiind posibil sa vad in timp real efectul vizual al prelucrarii interfetei. Pana acum am folosit doar prelucrarea interfetei grafice intr-un mod manual, scriind cod dar avand acum program de tip „drag and drop” mi s-a permis sa ma concentrez mult mai mult pe partea de problema adevarata, prelucrarea polinoamelor.

Folosirea clasei java Regex pentru a prelua polinomul din cadrul unui string a fost de asemenea o provocare care o sa-mi fie de mare ajutor pe viitor. Cea mai grea partea de aici a fost crearea sablonului asa incat sa se acopere fiecare dintre cazurile posibile.

De asemenea am invatat cum sa scriu o intreaga documentatie pentru un proiect, acest lucru fiind foarte important pentru a prezenta un proiect asa incat oricine va urma sa-l foloseasca sa poata sa afle tot de ce are nevoie.

## Dezvoltari ulterioare

Sunt o multime de lucruri ce se pot schimba in cadrul aplicatiei in functie de contextul de folosire pentru a imbunatatii functionalitatea sau partea de interfata cu utilizatorul.

**Posibile imbunatatiri de functionalitate :**

* Permiterea introducerii polinoamelor de mai multe variabile
* Posibilitatea introducerii unor polinoame cu puteri de tip real
* Adaugarea mai multor operatii posibile ca de exemplu : transformata Fourier, valoarea intr-un punct, gasirea radacinilor, ridicarea la putere, etc
* Prinderea tuturor exceptiilor care pot sa apara si returnarea unui mesaj care sa descrie exceptia aparuta
* Posibilitatea de alegere in cadrul scaderii, derivarii, integrarii polinoamelor asupra carora sa se execute operatiile.

**Posibile imbunatatiri de interfata**

* Desenarea polinomului rezultat(si rest in cazul impartirii)
* Punerea la dispozitie a mai mult spatiu pentru afisarea rezultatului
* Posibilitatea de a executa derivare sau integrare asupra aceluiasi polinom de mai multe ori, schimbandu-se automat continutul de intrare
* Adaugarea de butoane de curatare a campurilor
* Posibilitatea alegerii dintre mai multe exemple deja scrise pentru testare pe interfata

# Bibliografie

<http://stackoverflow.com/>

<https://www.youtube.com/>

<https://www.youtube.com/results?search_query=using+javafx+scene+builder+with+eclipse>

<http://docs.oracle.com/javase/7/docs/api/>