DOCUMENTATIE

TEMA 1

Polynomial Calculator

NUME STUDENT: ATITIENEI STEFAN COSTIN
GRUPA: 30227

CUPRINS

1. Obiectivul temei:

Obiectivul principal al acestei teme este obtinerea unui calculator pentru polinoame, care poate realiza calcule precum adunarea, scaderea, inmultirea, impartirea dintre doua polinoame si derivarea si integrarea unui polinom. Utilizatorul va folosi interfata grafica pentru a scrie doua polinoame de o singura variabila, cu coeficienti si puteri numere intregi, si va putea selecta operatia dorinta prin apasarea unor butoane.
Obiectivele secundare sunt urmatoarele :

- Analiza problemei. In capitolul 2 va fi detaliat acest obiectiv, unde se va vorbi despre modelarea problemei, scenariile de utilizare posibile si cazurile de utilizare. Identificarea cerintelor este de asemenea o etapa importanta in analiza problemei deoarece reduce semnificativ sansele de a schimba sistemul dupa dezvoltarea acestuia. Cerintele se vor clasifica in functie de prioritati si vor fi analizate. Pe parcursul proiectarii pot aparea noi cerinte care vor fi adaugate ulterior.
- Proiectarea sistemului: In capitolul 3, unde se va vorbi despre algoritmi si clasele folosite
- Implementarea sistemului: In capitolul 4, unde se va detalia implementarea claselor
- Testarea: In capitolul 5, unde se va specifica modul de testare a claselor

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

- **Analiza problemei :**

Procesul de dezvoltare al unei aplicatii porneste de la niste nevoi care sunt transformate in cerinte. Analiza problemei consta in identificarea acestor cerinte, care sunt de mai multe tipuri: functionale, non-functionale si constrangeri.
Cerintele functionale sunt urmatoarele :

- Calculatorul de polinoame trebuie sa ii ofere utilizatorului o interfata prin care sa poata comunica cu algoritmul, deoarece nu toti sunt specialisti in informatica si nu sunt familiari cu interactiunea printr-o consola.
- Calculatorul de polinoame trebuie sa permita utilizatorului sa insereze niste polinoame pentru care doreste sa aplice niste operatii
- Calculatorul de polinoame trebuie sa permita utilizatorului de asemenea sa selecteze operatia dorita : adunare, scadere, inmultire, impartire, derivare, integrare
- Calculatorul de polinoame trebuie sa realizeze corect calculele conform cerintelor utilizatorului si sa le afiseze acestuia tot prin intermediul interfetei grafice
- Aplicatia trebuie sa previna utilizatorul prin anumite mesaje in cazul inserarii gresite a unor polinoame sau in cazul selectarii gresite a unor operatii: de exemplu cazul in care utilizatorul uita sa insereze un polinom sau incearca sa imparta un polinom cu 0
- Calculatorul de polinoame trebuie sa afiseze rezultatul in functie de variabila pe care o foloseste utilizatorul in inserarea polinoamelor

Cerintele non-functionale sunt urmatoarele :

- Aplicatia ar trebuie sa aiba o interfata grafica cat mai simpla, intuitiva si usor de folosit de cat mai multe persoane. De exemplu un font marit pentru persoanele mai in varsta
- Sistemul construit ar trebui sa fie cat mai eficient din punctul de vedere al performantei, vitezei si eficientei, pentru a-i asigura utilizatorului o interactiune cat mai placuta
- Scalabilitatea este de asemenea o alta cerinta non-functionala

- **Modelarea :**

  In matematica, un polinom este alcatuit din mai multe monoame, iar un monom este alcatuit dintr-un coeficient, care poate fi un numar intreg in cazul nostru, si puterea unei variabile, care de asemenea poate fi un numar intreg. Astfel, vom avea o clasa pentru polinom, care contine o lista de monoame, care la randul lor vor avea o clasa. Celelalte clase vor face parte din structura MVC. Avand in vedere aceste aspecte, in capitolele urmatoare se vor descrie algoritmii folositi pentru calculele cu polinoame.
  Pe baza acestor cerinte se realizeaza arhitectura intregului sistem.

- **Scenarii:**

  Scenariul 1: Utilizatorul incearca sa adune 2 polinoame
  Actor principal : Utilizatorul
  Scenariu care se termina cu succes :

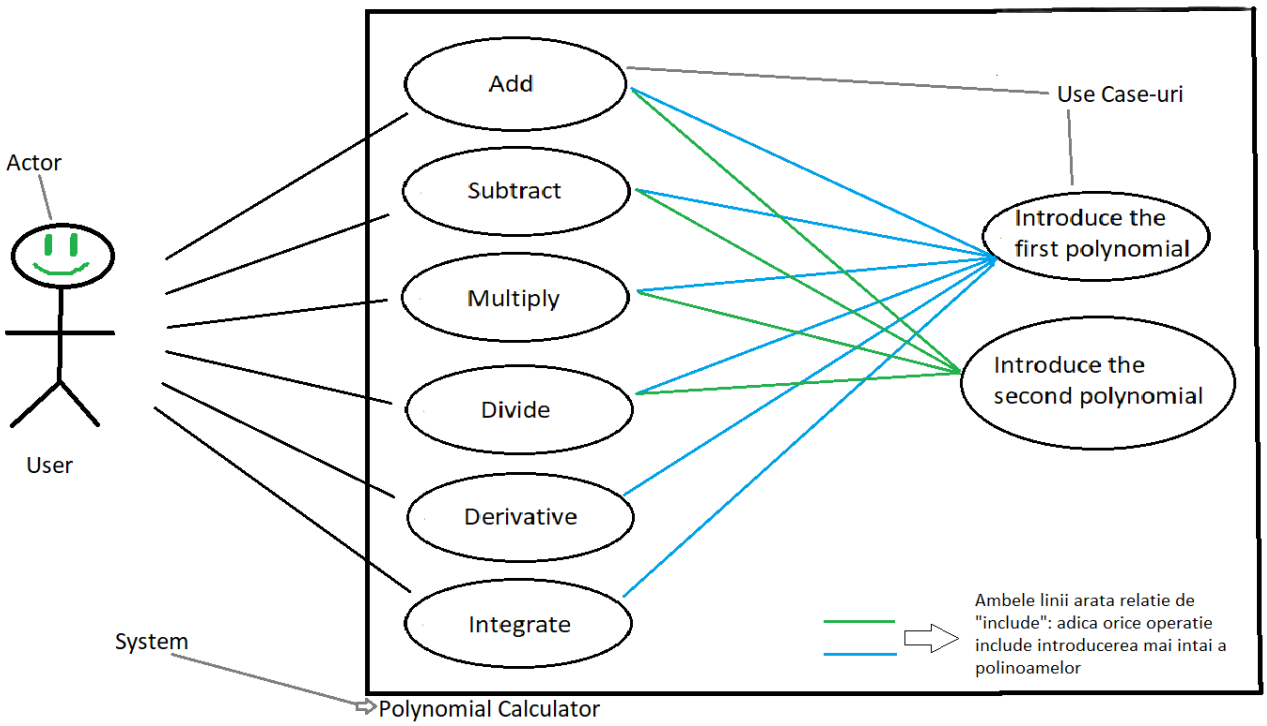
1) Utilizatorul insereaza doua polinoame in mod corect in interfata grafica
2) Sistemul verifica daca polinoamele au fost inserate corect
3) Utilizatorul apasa butonul corect pentru adunare
4) Algoritmul preia informatiile inserate de utilizator si afiseaza rezultatul corect in interfata grafica

   Exemplu :
   Polinom 1 : x^3+2x-5
   Polinom 2 : x^-2-7x^2+9x^3-12
   Rezultat : 10x^3-7x^2+2x-17+x^-2

  Scenariu alternativ :

- Utilizatorul uita sa insereze unul dintre polinoame : Apare un mesaj in care este avertizat de acest lucru. Scenariul contiuna apoi cu pasul 1.
  Un alt scenariu ar fi cand utilizatorul incearca sa impartaun polinom cu polinomul ‘0’, caz in care de asemenea apare un mesaj de atentionare si se continua cu scenariul 1.
  Pentru toate operatiile se respecta aceeasi pasi pentru scenariu.

- **Cazuri de utilizare:**



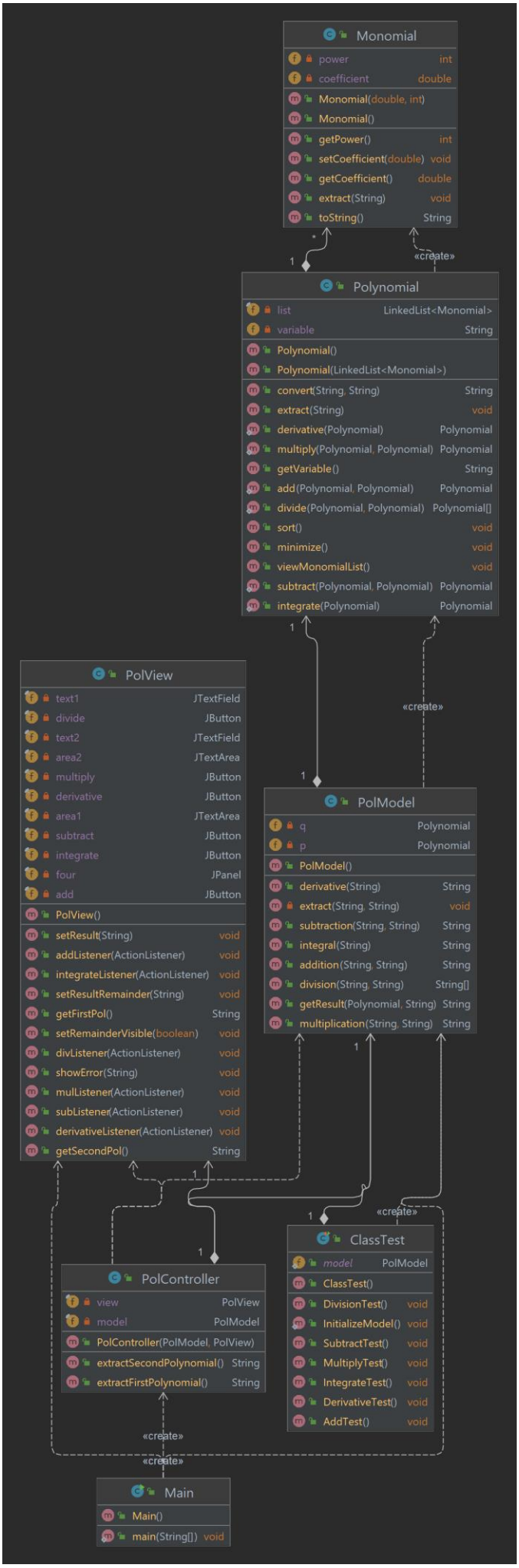
4. Proiectare:

4.1 Proiectarea OOP a aplicatiei :

Am ales sa folosesc o abordare orientata pe obiecte. Am format cate o clasa pentru polinom, monom, model, view si controlar. Am folosit abordarea cu MVC, in care clasa “View” consta in formarea interfetei grafice, clasa “Model” este alcatuita din algoritmii folositi pentru obtinerea rezultatului, iar clasa “Controller” este cea care face legatura dintre celelalte doua, adica preia informatii din clasa “View” (informatiile introduse de utilizator), le transforma cu ajutorul algoritmilor din “Model”, dupa care rezultatul obtinut din clasa “Model” il trimite clasei “View”, care il va afisa pe interfata grafica pentru ca utilizatorul sa il poata vedea.
Clasa “Monomial” contine atributele "coefficient" si "power", care sunt specifice unui monom din matematica. Clasa “Polynomial” contine o lista de monoame si contine si variabila in care este scris acesta : "variable".

4.2 Diagrama UML :

Aceasta evidentiaza relatiile de agregare, compozitie, dependenta, asociere, generalizare si realizare dintre clasele folosite. De asemenea se observa lista cu atributele folosite, metodele, si vizibilitatea acestora (daca sunt declarate public sau private).


Monomial
power int
coefficient double
Monomial(double, int)
Monomial()
getPower() int
setCoefficient(double) void
getCoefficient() double
extract(String) void
toString() String
Polynomial
list LinkedList<Monomial>
variable String
Polynomial()
Polynomial(LinkedList<Monomial>)
convert(String, String) String
extract(String) void
derivative(Polynomial) Polynomial
multiply(Polynomial, Polynomial) Polynomial
getVariable() String
add(Polynomial, Polynomial) Polynomial
divide(Polynomial, Polynomial) Polynomial[]
sort() void
minimize() void
viewMonomialList() void
subtract(Polynomial, Polynomial) Polynomial
integrate(Polynomial) Polynomial
PolView
text1 JTextField
divide JButton
text2 JTextField
area2 JTextArea
multiply JButton
derivative JButton
area1 JTextArea
subtract JButton
integrate JButton
four JPanel
add JButton
PolView()
setResult(String) void
addListener(ActionListener) void
integrateListener(ActionListener) void
setResultRemainder(String) void
getFirstPol() String
setRemainderVisible(boolean) void
divListener(ActionListener) void
showError(String) void
mulListener(ActionListener) void
subListener(ActionListener) void
derivativeListener(ActionListener) void
getSecondPol() String
PolModel
q Polynomial
p Polynomial
PolModel()
derivative(String) String
extract(String, String) void
subtraction(String, String) String
integral(String) String
addition(String, String) String
division(String, String) String[]
getResult(Polynomial, String) String
multiplication(String, String) String
PolController
view PolView
model PolModel
PolController(PolModel, PolView)
extractSecondPolynomial() String
extractFirstPolynomial() String
ClassTest
model PolModel
ClassTest()
DivisionTest() void
InitializeModel() void
SubtractTest() void
MultiplyTest() void
IntegrateTest() void
DerivativeTest() void
AddTest() void
Main
Main()
main(String[]) void
«create»
«create»
«create»
«create»
«create»
1
*
1
1
1
1
1
1
1
1

4.3 Structurile de date folosite :

Singura structura de date folosita este LinkedList<Monomial>. Am folosit-o pentru a adauga monoamele rezultate in urma despartirii unui polinom in monoame. Asupra acestei structuri de date se va folosi o metoda de sortare, care a fost implementata intr-o clasa separata care implementeaza interfata Comparator. In aceasta metoda se va face posibila ordonarea in functie de puterea fiecarui monom, descrescator. Astfel, datorita acestei metode de sortare, lista cu monoame din fiecare polinom va fi ordonata descrescator dupa puterea acestora.

4.4 Interfetele definite :

O interfata folosita este ActionListener. Aceasta este folosita pentru implementarea metodei "actionPerformed", care contine actiunea unui anumit buton in momentul in care este apasat. Functioneaza ca un "ascultator", reactionand la actiunile utilizatorilor conform cerintelor acestora.

O alta interfata este Comparator, folosita pentru a putea sorta lista de monoame ce alcatuieste un polinom. Pentru cazul in care utilizatorul nu introduce din prima polinomul descrescator in functie de puterea monoamelor, sistemul va ordona polinomul cu ajutorul metodei "sort" din clasa Collections, implementata intr-o clasa care implementeaza la randul ei interfata Comparator.

4.5 Algoritmi folositi :

Utilizatorul introduce doua polinoame, sau unul, in functie de operatia pe care vrea sa o realizeze. Sistemul trebuie sa preia datele introduse de utilizator (adica polinoamele) si sa le transforme in obiecte instante ale clasei polinom. Pentru acest lucru am folosit un regex, cu ajutorul caruia am descompus string-ul introdus de utilizator ca polinom, in mai multe stringuri de tip monom. Mai departe, tot cu ajutorul unui regex, am despartit stringul unui monom intr-un coeficient si o putere care sa caracterizeze monomul respectiv si am instantiat un obiect de tipul clasei "Monomial" cu aceste caracteristici. Astfel, am obtinut un monom pe care l-am introdus in lista de monoame a polinomului. Se repeta acest proces pentru fiecare monom al polinomului si in final se obtine un obiect de tipul clasei "Polynomial" care contine o lista de obiecte de tipul clasei "Monomial". Mai departe am avut nevoie de algoritmii pentru operatiile pe polinoame, descrisi mai jos:

1. <u>Adunare:</u>
   Pentru a aduna doua polinoame in matematica, se aduna coeficientii monoamelor cu aceeasi putere si rezulta astfel un polinom ca suma a celor doua. Aceeasi tehnica este folosita si de algoritm. Se parcurge lista de monoame a fiecarui polinom si se aduna intr-un rezultat toate monoamele cu puteri egale, sau daca nu exista puteri egale, se trec cu acelasi coeficient. Aceasta tehnica de adunare am aplicat-o cu ajutorul algoritmului de interclasare a doua liste: se parcurg in acelasi timp termenii celor doua liste si se compara puterile monoamelor. Daca sunt egale, se aduna coeficientii si se trece in ambele liste la urmatorul monom, altfel se adauga monomul cu putere mai mica la rezultat si se trece doar in acea lista la urmatorul monom. La final trebuie sa fie parcurse in intregime listele ambelor polinoame.
2. <u>Scadere:</u>
   Pentru a scadea doua polinoame se urmareste acelasi algoritm ca la adunare, singura diferenta fiind faptul ca se scad coeficientii monoamelor cu aceeasi putere, nu se mai aduna. Rezultatul operatiei va fi de asemenea tot un polinom.

3. Inmultire:
   La inmultirea dintre doua polinoame, este necesara parcurgerea listei primului polinom, iar pentru fiecare monom din aceasta, se parcurge intreaga lista din al doilea polinom si se inmultesc coeficientii, respectiv se aduna puterile variabilelor. Astfel, va rezulta cate un polinom partial in urma inmultirii fiecarui monom din primul polinom cu toate monoamele din al doilea. Aceste rezultate se vor aduna in final, urmand sa se formeze astfel si rezultatul final al inmultirii. Pentru aceasta tehnica de inmultire am parcurs in mod clasic listele polinoamelor cu "nested loops". Pentru aceasta operatie m-am folosit de operatia de adunare a doua polinoame, pentru a forma rezultatul final.
4. Impartire:
   Aceasta este operatia cea mai dificila dintre toate. Rezultatul aceste operatii va fi alcatuit din doua polinoame: unul fiind catul, iar celalalt restul.
   Daca impartim de exemplu polinomul P la polinomul Q, vom face urmatorii pasi:
- se ordoneaza polinoamele descrescator dupa puterea monoamelor cu ajutorul metodei "sort"
- se imparte primul monom al polinomului P cu primul monom al polinomului Q, astfel rezultand primul monom din cat
- se inmulteste catul obtinut pana in prezent cu Q si se scade rezultatul din P, obtinand astfel restul impartirii
- se repeta pasii considerand noul P ca fiind restul, pana cand puterea primului monom din P este mai mica decat puterea primului monom din Q
- ultimul P va reprezenta restul
  In capitolul de implementare este o poza cu pseudocodul dupa care am construit acest algoritm.
5. Derivare:
   Operatia de derivate necesita doar un singur polinom introdus de utilizator. Algoritmul este urmatorul:
- se parcurge lista monoamelor polinomului introdus
- se inmulteste fiecare coeficient al unui monom cu puterea acestuia
- se scade cu 1 puterea fiecarui monom (daca are puterea diferita de 0, adica daca exista variabila pentru acel monom)
6. Integrare:
   Pentru integrare este de asemenea nevoie de un singur polinom introdus de la tastatura. Algoritmul urmeaza urmatorii pasi:
- se parcurge lista monoamelor polinoului introdus
- si imparte fiecare coeficient al unui monom cu puterea acestuia (daca puterea e diferita de 0)
- puterea fiecarui monom va creste cu 1 (se va tine cont de situatia in care integram x^-1, care are ca rezultat "lnx")

5. Implementare:

1) Interfata Utilizator:

Am construit interfata utilizator cu ajutorul claselor si implicit metodelor din pachetul Java.Swing. Niste clase folosite sunt JButton, JTextArea, JLabel si altele.

Cu ajutorul acestora am format niste obiecte (butoane de exemplu) pe care le-am introdus pe un panel cu un anumit format, iar panel-ul pe o fereastrea care apare mereu la lansarea programului. Fereastra are un titlu "Polynomial Calculator", se inchide la apasarea butonului de "X" si se mareste/micsoreaza in functie de preferinte. Fereastra contine pe prima linie titlul aplicatiei, dupa care urmeaza doua zone de text in care se introduce primul, respectiv cel de-al doilea polinom. Apoi sub acestea se gaseste rezultatul care va fi contruit si afisat ulterior. Utilizatorul va putea scrie doar in zonele specifice introducerii de polinoame, nu si in zonele specifice afisarii rezultatului. Sub aceste zone de text se afla 6 butoane, cate 3 pe o coloana, care reprezinta operatiile ce pot fi selectate. In cazul selectarii operatiei de impartire, va mai aparea o zona de text sub zona rezultatului, care va contine restul impartirii, rezultatul fiind astfel considerat de fapt catul operatiei. Pentru o interfata mai "user-firendly" am si colorat fundalul si butoanele.
Toate aceste lucruri au fost facut in clasa "PolView", in constructorul ei, deoarece ele trebuie sa apara de fiecare data cand se construieste o astfel de fereastra. Pe langa acestea, clasa mai contine si anteturi de metode precum: obtinerea polinoamelor introduse, setarea rezultatului sau diverse actiuni care sa aiba loc in momentul apasarii unui buton. Aceste metode vor fi de fapt implementate in clasa 'PolController".

2) **PolController:**
Se foloseste de un obiect de tipul clasei View si unul de tipul clasei Model pentru a putea implementa ascultatorii butoanelor din interfata. Aici sunt implementate metodele definite in clasa "PolView". Pentru fiecare metoda (implementata intr-o clasa diferita din interiorul acestei clase), se foloseste o structura de tipul "try..catch.." pentru a ma asigura ca utilizatorul introduce ceva in campul pentru polinoame. In caz negativ, se afiseaza un mesaj utilizatorului si se asteapta introducerea corecta. In caz pozitiv, urmeaza extragerea polinomului introdus (ca String), apelarea metodei din clasa "PolModel" care sa imi formeze rezultatul in functie de operatia dorita, dupa care se apeleaza metoda de afisare a rezultatului definita in clasa "PolView".
La clasa in care este ascultatorul pentru operatia de impartire, se activeaza si zona de text pentru afisarea restului in momentul in care se afiseaza rezultatul, iar in restul ascultatorilor se ascunde acea zona de text deoarece nu se doreste afisarea acestuia pentru ca nu exista pentru alte operatii.

3) **PolModel:**
Are ca variabile instanta doua obiecte de tipul clasei "Polynomial"
Contine urmatoarele metode:

- extract: Primeste un String "s" pe care il transforma intr-un polinom de tipul clasei "Polynmial" prin apelarea unei metode din acea clasa. Dupa aceea sorteaza polinomul descrescator dupa puterea monoamelor, iar la final minimizeaza polinomul: adica aduna monoamele cu aceeasi putere (daca exista)
- getResult: transforma un obiect de tipul Polynomial intr-un String. Metoda e folosita pentru afisarea rezultatului
- addition: metoda folosita pentru adunare
se apeleaza metoda de adunare construita in clasa "Polynomial" si se intoarce rezultatul sub forma de string (adica dupa ce e convertit) inapoi la "PolController"

Toate celelalte metode din clase functioneaza dupa acelasi principiu ca si metoda pentru adunare, cu mici diferente precum: la integrare se adauga constanta "C" la rezultat, iar la impartire si intorc doua stringuri ca rezultat.

4) **Polynomial:**

Este clasa care contine cele mai importante metode si cei mai importanti algoritmi.
Are ca atribute o lista inlantuita de monoame si o variabila care contine variabila in care este scris polinomul respectiv
Metode:
-minimize: minimizeaza un polinom (reduce monoamele cu aceeasi putere la un singur monom)
-extract: foloseste un regex pentru a extrage monoamele si a le adauga ulterior in lista
-add: am folosit algoritmul de interclasare pentru aceasta operatie
-subtract: am folosit algoritmul de interclasare pentru aceasta operatie
-multiply: folosesc adunari repetate pe polinoame temporare obtinute in urma inmultirii a cate un monom dintr-un polinom cu toate monoamele din celalalt polinom
-divide: Implementarea am facut-o cu ajutorul urmatorului pseudocod:

```
function n / d is
    require d ≠ 0
    q ← 0
    r ← n             // At each step n = d × q + r

    while r ≠ 0 and degree(r) ≥ degree(d) do
        t ← lead(r) / lead(d)       // Divide the leading terms
        q ← q + t
        r ← r − t × d

    return (q, r)
```

-derivative: folosesc un for..each pentru parcurgerea listei de monoame
-integrate: folosesc un for..each pentru parcurgerea listei de monoame. Pentru calculul coeficientului, acesta fiind real si avand multe zecimale, se va calcula astfel incat sa se afiseze doar cu doua zecimale cu formula:
(Nr real * 100) /100
-convert: contine parametrul t, care reprezinta variabila in care se va scrie polinomul si parametrul "integrate", care arata daca este vorba despre integrare, pentru a testa si cazul in care integrala(x^-1)=lnx. Aceasta metoda transforma un polinom intr-un string (afisarea polinomului sub forma de string)

5) **Monomial:**
   Contine o variabila double pentru coeficient si una intreaga pentru putere. Contine de asemenea o metoda pentru extragerea unui monom dintr-un string (extragerea coeficientului si puterii) tot cu ajutorul unui regex. In rest contine doar getter-e si setter-e.

6. Rezultate:

Pentru aceasta parte a aplicatiei am format o clasa numita ClassTest in care am importat pachetul Junit. Am construit cate o clasa de test pentru fiecare operatie si am testat corectitudinea acestora. Rezultatul a fost corect pentru toate operatiile. Nu am folosit metode cu mai mult de 30 de linii si nici clase cu peste 300.

7. Concluzii:

In concluzie, consider ca aceasta tema m-a ajutat in consolidarea cunostintelor despre Java si programarea orientata pe obiecte. S-a simtit lipsa de obisnuita in a lucra cu aceste concepte si de aceea timpul terminarii acestui proiect a fost destul de indelungat.
Posibilele dezvoltari ulterioare ar putea fi urmatoarele:

- Implementarea operatiilor pe mai mult de doua polinoame
- Implementarea derivarii de grad mai mare decat 1
- Posibilitatea introducerii unor polinoame de mai multe variabile
- Atentionarea utilizatorului in cazul introducerii unor polinoame gresite ca sintaxa

8. Bibliografie:

Pentru operatii:

- Suportul de Prezentare oferit pentru aceasta tema

Pentru informatii legate de Java si programarea orientata pe obiecte:

- Cursurile si laboratoarele de la POO
- https://docs.oracle.com/javase/7/docs/api/

Pentru testarea operatiilor:

- https://www.symbolab.com/solver/polynomial-long-division-calculator