# Sistem de procesare a polinoamelor

# de o singura variabila

## Obiectivul temei

Scopul principal al temei este crearea unui sistem de procesare a polinoamelor de o singura variabila in vederea obtinerii unei aplicatii ce sa usureze calculele, oferind prin intermediul unei interfete grafice, posibilitatea realizarii automate a anumitor operatii pe polinoame. Astfel, se identifica urmatoarele obiective in vederea obtinerii unei aplicatii viabile si sustenabile:

|  |  |
| --- | --- |
| Obiective |  |
| Analiza cerintelor functionale  (Capitolul 2) | * Identificarea scenariilor de functionare din punct de vedere al utilizatorului si valorile de intrare introduse de acesta * Identificarea cerintelor functionale: operatiile pe date ce trebuie efectuate de catre sistem * Modelarea procesului de functionare al aplicatiei |
| Proiectarea  (Capitolul 3) | * Structurarea proiectului: impartirea sa in clase si pachete, diagrama UML * Definirea unor structuri de date ce sa faciliteze implementare precum si lizibilitatea codului * Analiza algoritmilor ce trebuie implementati in vederea cresterii performantei |
| Implementarea propriu-zisa (Capitolul 4) | * Punerea in practica a pasilor anterior * Urmarea pasilor de proiectare in mod sistemic pentru evitarea scrierii de cod redundant * Scrierea de cod lisibil si usor de interpretat prin limitarea liniilor de cod (maxim 30 de linii per metoda, respectiv maxim 300 de linii per clasa) |
| Analiza rezultatelor (Capitolul 5) | * Conceperea de teste menite sa verifice fiabilitatea si corectitudinea implementarii tuturor cerintelor funtionale * Interpretarea rezultatelor si descrierea eventualelor erori aparute sau identificate prin acestea |
| Dezvoltare  (Capitolul 6) | * Analiza retrospectiva a proiectului surprinzand dificultatile intalnite * Prezentarea posibilitatilor de dezvoltare ulterioare ale aplicatiei |

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

Functionalitatile identificate in urma analizei cerintelor stipulate de catre client sunt:

* ***Adunarea*** a doua polinoame introduse in interfata grafica si afisarea rezultatului operatiei;
* ***Scaderea*** celor doua polinoame introduse in interfata grafica si afisarea rezultatului operatiei intr-o forma usor de interpretat;
* ***Inmultirea*** a doua polinoame introduse in interfata grafica precum si afisarea rezultatului operatiei;
* ***Impartirea*** a doua polinoame introduse in interfata grafica si afisarea catului si restului rezultat in urma efectuarii operatiei;
* ***Derivarea*** unui polinom, introdus in interfata grafica si afisarea sa intr-o forma ce faciliteaza calculele urmatoare;
* ***Integrarea*** polinomului introdus in interfata grafica si afisarea sa intr-o forma usor de interpretat.

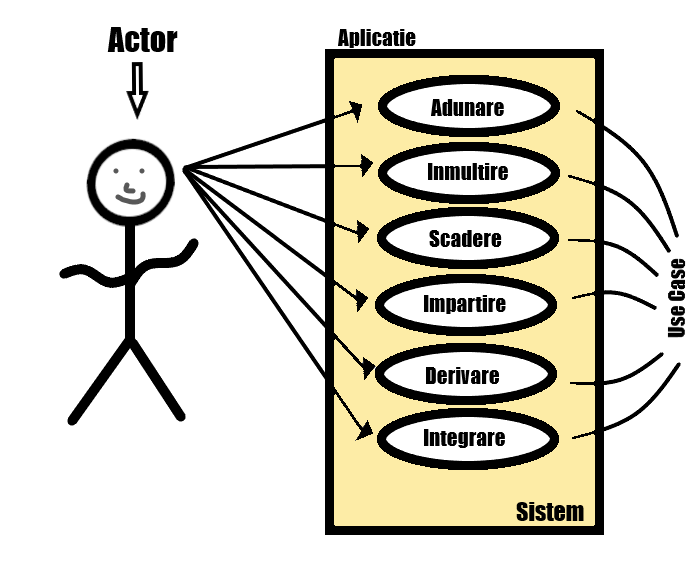


Fig 1. Diagrama cazurilor de utilizare

In cele ce urmeaza vor fi prezentate cazurile de utilizare ale aplicatiei:

*Actor principal: Utilizator al aplicatiei (Student, Elev, Profesor etc.)*

**Use Case: Adunare/Scadere/Inmultire**

Scenariu uzual reusit:

1. Actorul introduce in aplicatie polinoamele ce trebuie adunate/scazute/inmultite
2. Actorul apasa butonul „Calculeaza”
3. Aplicatia verifica daca polinoamele introduse sunt valide
4. Aplicatia proceseaza datele introduse si realizeaza operatiile
5. Aplicatia afiseaza rezultatul operatiei
6. Actorul preia rezultatul dat de aplicatie

Scenarii alternative:

1. Cel putin unul dintre polinoame este gresit
   * + Aplicatia ii transmite actorului ca input-ul este gresit
     + Scenariul se intoarce la pasul 1
2. Autorul nu introduce unul dintre polinoame
   * + Aplicatia interpreteaza polinomul ca fiind 0
     + Scenariul continua de la pasul 3

**Use Case: Impartire**

Scenariu uzual reusit:

1. Actorul introduce in aplicatie polinoamele ce trebuie impartite
2. Actorul apasa butonul „Calculeaza”
3. Aplicatia verifica daca polinoamele introduse sunt valide
4. Aplicatia verifica daca impartitoturul este egal cu 0
5. Aplicatia proceseaza datele introduse si realizeaza operatia de impartire
6. Aplicatia afiseaza catul si restul impartirii
7. Actorul preia rezultatul dat de aplicatie

Scenarii alternative:

1. Cel putin unul dintre polinoame este gresit
   * + Aplicatia ii transmite actorului ca input-ul este gresit
     + Scenariul se intoarce la pasul 1
2. Autorul nu introduce impartitorul ca fiind 0
   * + Aplicatia interpreteaza impartitorul ca fiind 0
     + Aplicatia ii transmite actorului ca input-ul este gresit
     + Scenariul continua de la pasul 1

**Use Case: Derivare/Integrare**

Scenariu uzual reusit:

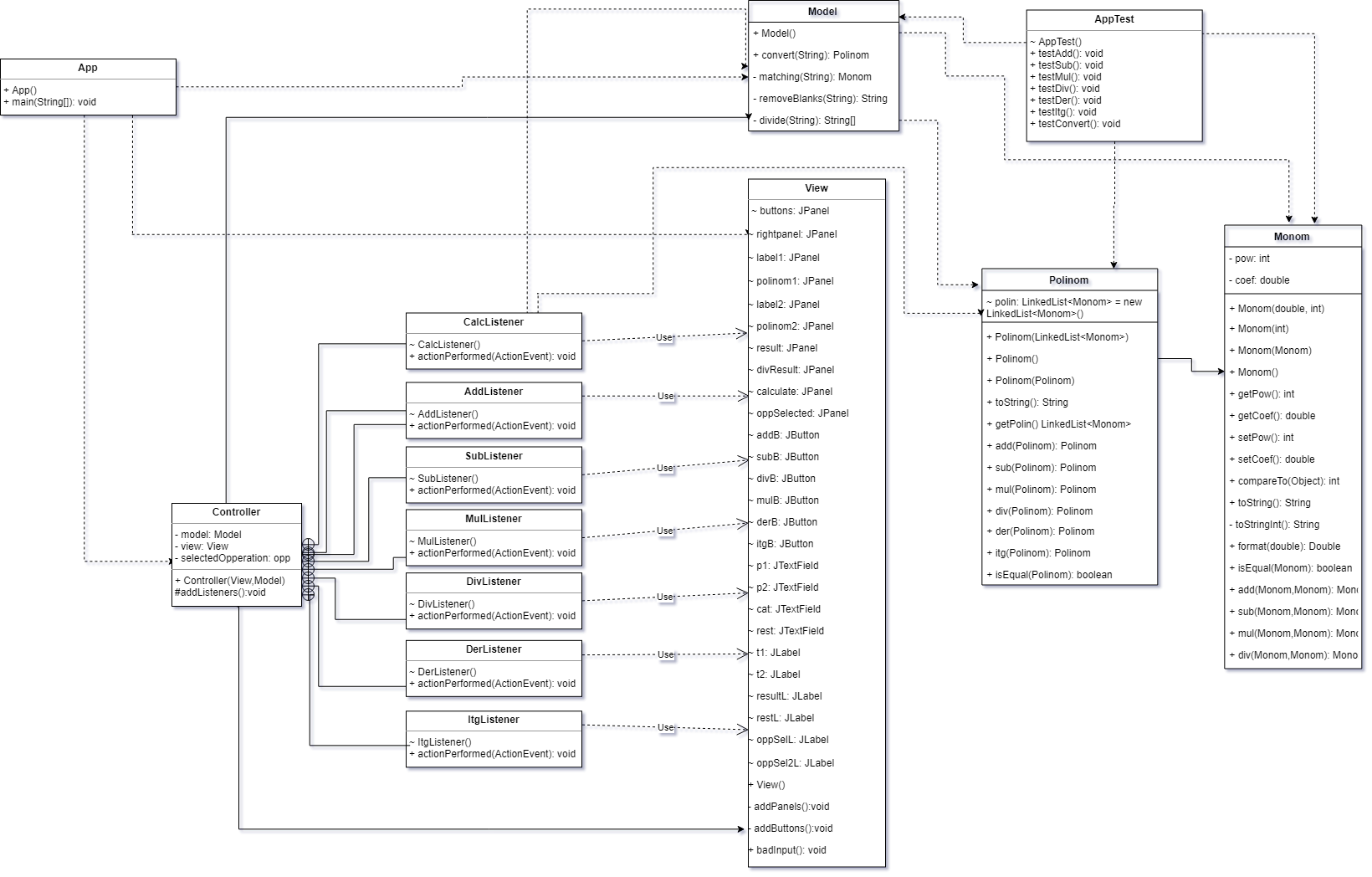
1. Actorul introduce in aplicatie polinomul de derivat/integrat
2. Actorul apasa butonul „Calculeaza”
3. Aplicatia verifica daca polinomul introdus este valid
4. Aplicatia proceseaza datele introduse si realizeaza operatia selectata
5. Aplicatia afiseaza rezultatul operatiei
6. Actorul preia rezultatul dat de aplicatie

Scenarii alternative:

1. Polinomul este gresit
   * + Aplicatia ii transmite actorului ca input-ul este gresit
     + Scenariul se intoarce la pasul 1
2. Polinomul introdus este 0
   * + Aplicatia afiseaza ca rezultat al derivarii/integrarii: 0
     + Scenariul se intoarce la pasul 1
3. Autorul nu introduce polinomul
   * + Aplicatia interpreteaza polinomul ca fiind 0
     + Scenariul continua cu scenariul b

## Proiectarea aplicatiei

Diagrama UML a proiectului:



Pentru usurarea operatiilor, am impartit polinomul in monoame, acestea fiind reprezentate prin coeficient si putere. In acest fel, polinomul propriu zis poate fi implementat cu ajutorul unei liste inlantuite.

Pentru ca descrierea interfetei grafice sa fie mai aerisita, se poate utiliza sistemul MVC ( Model View Controller ), impartindu-se astfel diferite atributii intre cele 3 clase. Modelul contine partea functionala, reprezentata de interpretarea input-ului venit din View. In View se defineste interfata grafica pe care utilizatorul o foloseste, iar Controllerul este insarcinat cu implementarea ascultatorilor pentru butoane si respectiv actiunile aferente in urma apasarii fiecaruia.

Lizibilitatea codului este un atu important, motiv pentru care in cadrul Controllerului, pentru selectia operatiei s-a utilizat enumeratia opp cu elementele in ordine: ***ADD***, ***SUB***, ***MUL***, ***DIV***, ***DER***, ***ITG*** .

## Implementarea

In cadrul proiectului se remarca prezenta urmatoarelor clase:

1. **Monom**

Clasa este implementata cu campurile coef de tipul double reprezentand coeficientul monomului si pow de tipul int reprezentand puterea monomului. Deasemenea clasa implementeaza interfata Comparable si functia aferenta compareTo care returneaza valori astfel incat lista de monoame sa fie ordonata in ordinea descrescatoarea puterilor.

Metodele format, toString si toStringInt se ocupa de afisarea monomului intr-o forma usor de interpretat.

Functiile add, sub, mul si div realizeaza operatiile adunare, scadere, inmultire si impartire pe monoame.

1. **Polinom**

Foloseste drept camp o lista inlantuita de monoame.

Metodele add/sub realizeaza suma/diferenta a doua polinoame si functioneaza pe principiul interclasarii, iterand pe fiecare dintre cele doua polinoame primite si adunand/scazand monoamele cu acelasi grad. In final functia returneaza un nou polinom ce reprezinta rezultatul adunarii/scaderii.

Metoda mul inmulteste fiecare monom din polinomul primit ca parametru cu monoamele obiectului cu care s-a apelat functia, rezultatul adunandu-se la variabila ce urmeaza a fi returnata. Procesul se repeta pana cand se parcurg toate monoamele polinomului primti ca parametru.

Metoda div imparte polinomul cu care s-a facut apelul la polinomul primit ca argument. Algoritmul gaseste monomul cu care trebuie inmultit impartitorul pentru a ajunge la acelasi grad cu deimpartitul. Dupa realizearea produsului dintre monom si impartitor, rezultatul se scade din deimpartit, iar monomul e adaugat la catul final. Procesul se incheie in momentul in care gradul impartitorului este mai mare decat gradul deimpartitului care devine astfel restul, returnandu-se catul.

Metodele der/itg deriveaza/integreaza polinomul primit ca parametru inmultind coeficientul fiecarui monom cu puterea acestuia si scazand apoi puterea cu unu in cazul derivarii, respectiv crescand puterea cu 1 si apoi impartind coeficientul la noua putere.

Metoda isEquals compara monom cu monom polinoamele si returneaza true daca sunt egale sau false daca sunt diferite.

1. **AppTest**

Testeaza operatiile implementate in clasa Polinom.

1. **Model**

Metoda convert are rolul de a transforma un string primit ca si paramentru intr-un polinom sau de a arunca o exceptie daca acest lucru nu este posibl. Pentru aceasta operatie, metoda apeleaza alte metode precum removeBlanks care elimina toate spatiile din string, divide, ce imparte stringul in substringuri delimitate de caracterul „+” si matching, ce utilizand Patterns incearca sa vada daca poate recunoaste un pattern in fiecare substring.

1. **Clasele Listener**

Implementeaza ascultatorii pentru fiecare buton si updateaza label-uri.

1. **Controller**

Contine drept campuri o variabila View si una Model, el facand legatura intre cele doua. Deasemenea contine si enumeratia opp, fiind totodata insarcinata cu adaugarea de ascultatori pe butoanele din view si selectarea operatiilor ce trebuie facute la apasarea butonului „Calculeaza”.

1. **App**

Contine main-ul necesar pentru lansarea aplicatiei.

1. **View**

Reprezinta implementarea interfetei grafice. Aceasta este alcatuita frame-ul impartit in doua mari parti. In partea stanga se afla 6 butoane utilizate pentru selectarea operatiei ce urmeaza a fi implementata, iar in partea stanga se gasesc text field-urile unde se vor introduce polinoamele, respectiv unde se va afisa rezultat in urma apasarii butonului Calculeaza. Fiecare buton in parte primeste un tip de ascultator personalizat intrucat functionalitatile difera radical, iar la apasarea butoanelor din partea stanga, panoul din dreapta se modifica eliminand campurile redundante (spre exemplu, derivarea se realizeaza pe un singur polinom, astfel text field-ul cu numarul 2 utilizat in adunare se ascunde).

Am ales aceasta interfata impartita in 2 bucati pentru a creste usurinta cu care aplicatia poate fi utlizata (cum am precizat mai sus, prin eliminarea campurilor redundante). In acest sens, la lansarea aplicatiei, se porneste un ghid de utilizare, iar in panoul din dreapta jos este indicata tot timpul operatia selectata.

## Rezultate

Pentru testarea operatiilor a fost implementata metoda isEqual ce returneaza „true”, daca polinoamele sunt egale, respectiv „false”, daca acestea difera. Verificarea incepe prin a sorta cele doua liste de monoame, astfel asigurandu-ne ca primul element din fiecare lista este monomul cu gradul cel mai mare al polinomului. Apoi din motive de performanta, se verifica daca gradele primelor monoame sunt diferite, moment in care metoda returneaza „false”. In caz contrar se incepe verificarea celor doua polinoame monom cu monom. Daca la cel putin una dintre aceste perechi gradul si coeficientul difera, metoda returneaza din nou „false”, iar daca verificarea este incununata de succes aceasta returneaza „true”.

Testarea operatiilor a fost efectuata prin metodele:

* testAdd – ce testeaza adunarea prin crearea a trei polinoame:

p1 = 5x^4-10x^3-x+5

p2 = 3x^5+10x^3-x+15

p3 = 3x^5+5x^4-2x+20

se poate observa ca suma primelor doua polinoame are ca rezultat al treilea polinom. Astfel prin intermediul metodei assertTrue se verifica daca rezultatul adunarii p1 si p2 este egal cu p3;

* testSub – ce testeaza scaderea prin crearea a trei polinoame:

p1 = 5x^4-10x^3-x+5

p2 = 3x^5+10x^3-x+15

p3 = -3x^5+5x^4-20x^3-10

se poate observa ca diferenta primelor doua polinoame are ca rezultat al treilea polinom. Astfel prin intermediul metodei assertTrue se verifica daca rezultatul scaderii p1 si p2 este egal cu p3;

* testMul – ce testeaza inmultirea prin crearea a trei polinoame:

p1 = 5x^4-10x^3+5

p2 = x^2+2x

p3 = 5x^2+5x^6-20x^4+10x

se poate observa ca produsul primelor doua polinoame are ca rezultat al treilea polinom. Astfel prin intermediul metodei assertTrue se verifica daca rezultatul inmultirii p1 si p2 este egal cu p3;

* testDiv – ce testeaza inpartirea prin crearea a patru polinoame:

p1 = 5x^4-10x^3+5

p2 = x^2+2x

p3 = 5x^2+40-20x

p4 = 4-80x

se poate observa ca raportul primelor doua polinoame are ca si cat al treilea polinom, iar rest al patrulea. Astfel prin intermediul metodei assertTrue se verifica daca rezultatul impartirii p1 si p2 are catul egal cu p3 si restul egal cu p4;

* testDer – ce testeaza derivarea prin crearea a doua polinoame:

p1 = 5x^4-10x^3+5x+8

p2 = 20x^3-30x^2+5

se poate observa ca derivarea primului polinom are ca si rezultat al doilea polinom. Astfel prin intermediul metodei assertTrue se verifica daca rezultatul derivarii p1 este egal cu p2;

* testItg – ce testeaza integrarea prin crearea a doua polinoame:

p1 = 5x^4-10x^3+5x+8

p2 = x^5-2.5x^4+5x

se poate observa ca integrarea primului polinom are ca si rezultat al doilea polinom. Astfel prin intermediul metodei assertTrue se verifica daca rezultatul integrarii p1 este egal cu p2;

* testConvert – ce testeaza convertirea prin crearea unui polinom si a unui string:

p = -15x^6+12x^5+x^4+2x^2-10x^3+5x-44

s = -15\*x^6+ 12x^5+x^4+2x^2-10x^3+5x-44

se poate observa ca polinomul este echivalent cu valoarea stringului. Astfel prin intermediul metodei assertTrue se verifica daca prin convertirea stringului s rezulta polinomul p.

## Concluzii

In urma realizarii acestui proiect am invat importanta separarii clare a obiectelor de care am nevoie pentru creearea proiectului. Am inteles ca este nevoie de o analiza amanuntita inainte de inceperea scrierii codului pentru a ajunge de la inceput la cea mai buna solutie de implementare si pentru a nu fi nevoit sa modific codul radical in timpul scrierii lui. De asemnea, 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.

Posibilitati de dezvoltare ulterioara:

* Modificarea labelurilor in cazul impartirii si scaderii pentru o mai usoara utilizare
* Permiterea utilizarii drept variabila si a altor litere nu doar „x”
* Implementarea de operatii pe mai multe variabile
* Implementarea unui sistem care sa calculeze si sa afiseze graficul functiilor

## Referinte

<https://www.tutorialspoint.com/java/java_regular_expressions.htm>

<https://regexr.com/>