DOCUMENTATIE

TEMA NUMARUL 1

Mudura Ana-Andreea

Grupa 30221

**CUPRINS**

**No table of contents entries found.**

**1.Obiectivul problemei....................................................................................... 3**

**2.Analiza calculatorului..................................................................................... 4**

**3.Proiectarea calculatorului.............................................................................. 7**

**4.Rezultate.......................................................................................................... 12**

**5.Concluzii.......................................................................................................... 13**

**6.Bibliografie...................................................................................................... 14**

**Obiectivul temei**

Obiectivul temei numărul 1 este implementarea unui calculator polinomial folosind paradigmele programării orientate pe obiect, folosirea unei interfețe grafice pentru utilizator si modelarea problemei sub forma unei arhitecturi Model-View-Controller

Se va prezenta:

* Modelarea MVC a calculatorului împreuna cu cazurile de utilizare a acestuia
* Paradigmele folosite in proiectarea temei
* Implementarea calculatorului
* Modalitatea de testare a aplicației cu ajutorul unui framework
* Concluzii

**Analiza calculatorului**

Calculatorul polinomial este modelat pe arhitectura Model-View-Controller care are ca si scop izolarea logicii computaționale de interfața cu utilizatorul, astfel rezultând o aplicație in care aspectul vizual si controlul propriu-zis sunt mai ușor accesibile si modificabile, fără a afecta comunicarea dintre cele doua.

Modelul – Polinomul

Polinomul manipulează operațiile logice utilizează informația primita de la interfața. Conține metode apelate de control împreuna cu informația care va urma sa fie folosita

View(Vizualizarea)

Vizualizarea conține interfața grafica cu utilizatorul, in cazul nostru se folosește biblioteca JAVAX SWING pentru proiectarea acesteia, împreuna cu definirea butoanelor de control(Action Listener) prin intermediul cărora se va manipula interfața de către client.

Controlul

Controlul are ca si conținut principal metodele evenimentelor care sunt împărțite pe cazuri de utilizare ca apoi rezultatul sa fie transmis înapoi către interfața grafica.

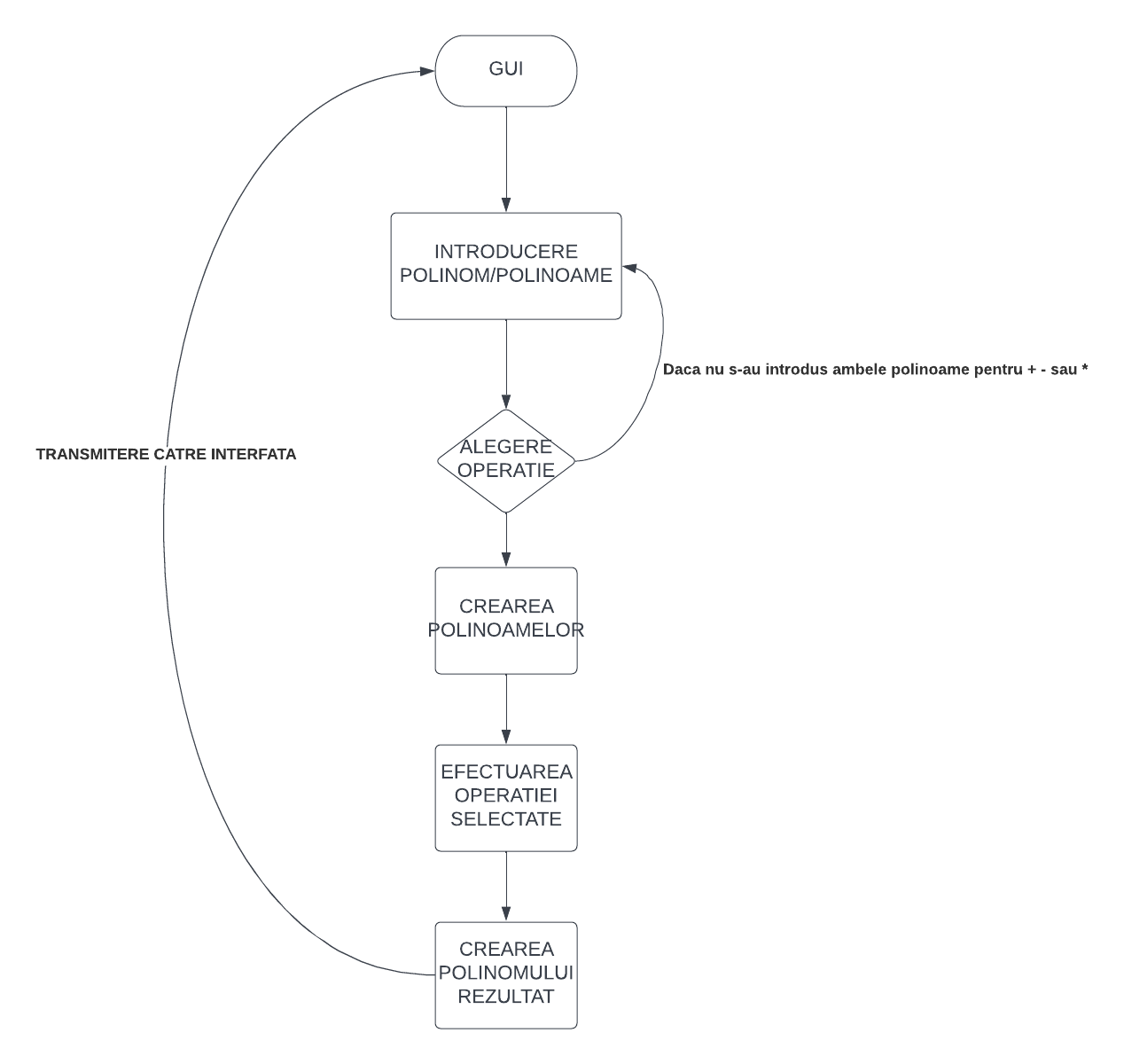
*USE-CASEURI*

Polinomul este format din unul sau mai multe monoame care constituie o lista care retine gradul si coeficientul fiecărui element.

Polinomul poate sa fie adunat, scăzut sau înmulțit cu un alt polinom sau poate sa fie derivat sau integrat cu ajutorul metodelor. In metodele dedicate este tratat fiecare caz in parte pentru fiecare operație selectata pentru amplificarea corectitudinii. Se tratează cazuri de null sau cazuri de introduceri incorecte a polinoamelor. Procesul de adunare, scădere si înmulțire necesita introducerea a doua polinoame in interfața, iar pentru procesul de derivare sau integrare exista un field dedicat pentru polinomul suspus operației cu scopul de a ușura utilizarea aplicației.

In diagrama următoare este prezentat un use case pentru operațiile executate de către calculator, în funcție de metodele apelate. Fiecare celula descrie fiecare proces apelat de la accesarea interfeței grafice cu utilizatorul pana la afișarea rezultatului in secțiunea dedicata din GUI.

**DIAGRAMA DE UTILIZARE A CALCULATOULUI POLINOMIAL**



**PROIECTAREA CALCULATORULUI**

Pentru realizarea calculatorului sunt folosite paradigmele programării orientate pe obiect care are ca si principii de baza in acest proiect încapsularea si polimorfismul.

Încapsularea (care mai poate fi interpretata ca si ascunderea de informații) asigura faptul ca obiectele încapsulate nu pot schimba starea interna a altor obiecte in mod direct(accesul se poate realiza doar prin intermediul metodelor proprii). In alta ordine de cuvinte, starea obiectului curent poate fi schimbata doar de metodele puse la dispoziție de obiectul respectiv.

Pentru "ascunderea" informațiilor, toate proprietățile polinomului, respectiv ale monomului au fost încapsulate. In program, informațiile despre fiecare obiect se pot obține prin utilizarea metodei de getter care are ca si scop returnarea stării curente a monomului/polinomului, iar pentru schimbarea stării curente se folosește metoda de setter.

Polimorfismul reprezintă atribuirea mai multor funcționalități unei metode, adica variabilele de intrare sunt diferite. In cazul calculatorului, funcția de add a colecțiilor (Linked List) are ca si corespondent funcția de add a celor doua polinoame. Funcția de add a colecțiilor are ca si intrare un monom, cea apelata de interfața un polinom.

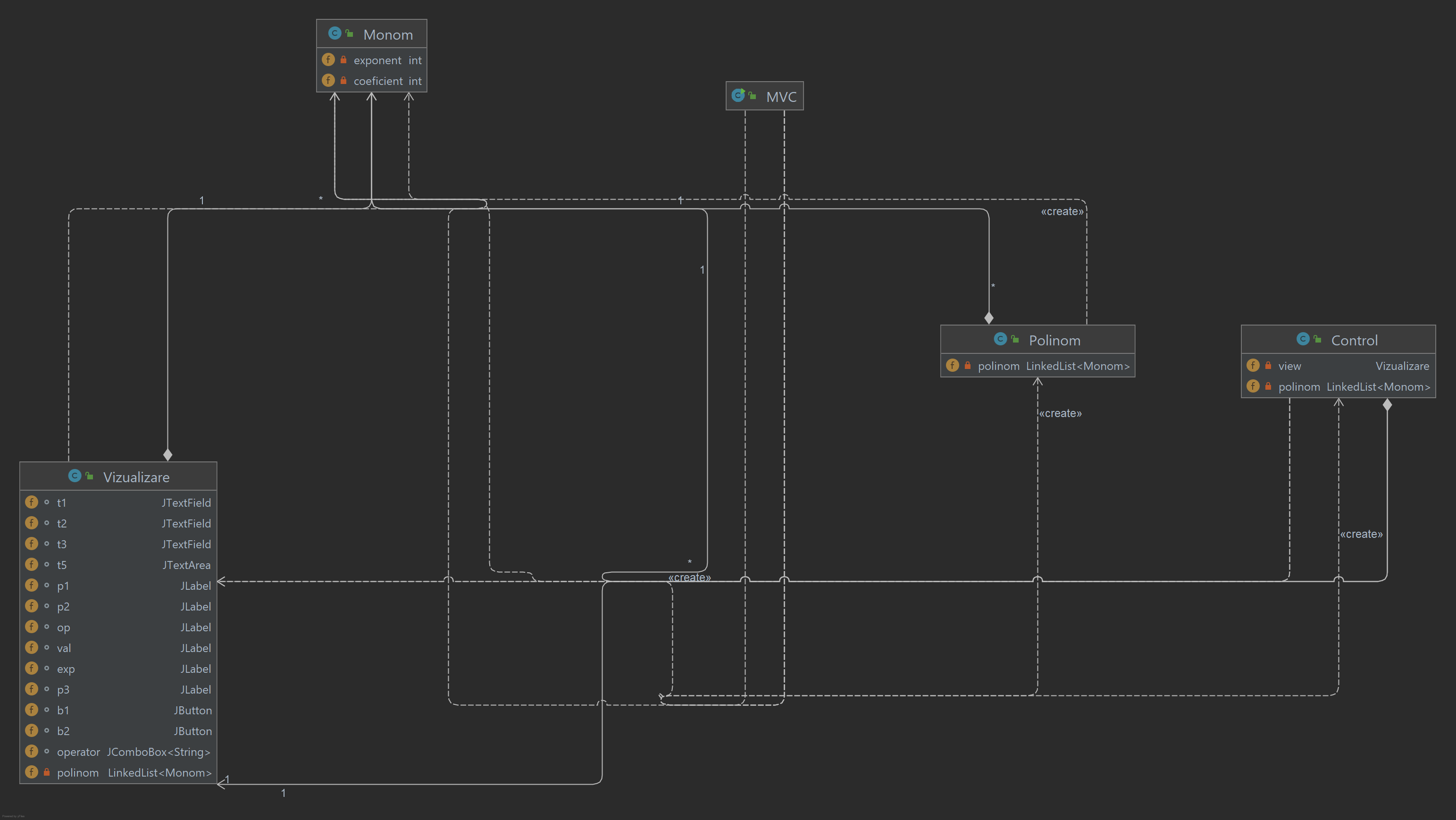
In diagrama UML următoare se pot observa relațiile dintre clase împreuna cu pachetele folosite si interfețele utilizate.

Intre Monom si Polinom exista o relație de dependenta( adică o relație de tip "folosește") din care se poate observa faptul ca o schimbare in unul dintre elemente poate cauza o schimbare in celălalt. Polinomul folosește gradul si coeficientul fiecărui Monom pentru efectuarea operațiilor. Exista relație de dependenta si intre Monom si Vizualizare, adică fiecare element din polinom trebuie transmis interfeței. Cea mai importanta relație de dependenta poate sa fie vizualizata intre MVC( clasa care combina fiecare element al arhitecturii pentru a fi rulat) si clasele Polinom-Monom-Control, adică programul principal se folosește de modelul propriu-zis si controlul modelului pentru a efectua operațiile.

Tot intre Monom si Polinom exista relația de compoziție adică poate fi interpretata ca si "un monom este parte a unui polinom". Compoziția vine sub forma unei agregări ("un polinom are mai multe monoame"), o formă specială de asociere care modelează relația parte–întreg între un agregat (întregul) și părțile sale. Relația se mai găsește intre Monom si Vizualizare("un monom poate fi vizualizat") si intre Control si Vizualizare("fiecare control al aplicației poate sa fie vizualizat").

Pe diagrama se mai pot observa si încapsulările folosite.

**DIAGRAMA UML A CLASELOR**



După cum am menționat mai sus, monomul încapsulează gradul si coeficientul sau. Fiecare polinom este format din mai multe monoame care fiecare are coeficientul si puterea sa, drept urmare structura de polinom am reprezentat-o cu o lista simplu înlănțuita, unde fiecare nod reprezintă un monom cu proprietățile sale. Coeficientul si puterea sunt reprezentate sub forma de numere întregi, iar informațiile preluate/afișate pe interfața grafica sub reprezentate sub forma de Stringuri.

Singura interfața folosita in proiectarea calculatorului este ActionListener, o interfața publica care primește ca argument un tip de date numit eveniment si exercita o acțiune(prin metoda actionPerformed) asupra unui obiect de tip component, dedicat interfețelor grafice. In proiect, fiecare buton deține metoda addListener(Compute – care la apăsarea lui se efectuează calculul ales – si Clear – butonul de revenire la starea inițială a interfeței).

**IMPLEMENTAREA CALCULATORULUI**

După cum am precizat in capitolele anterioare, calculatorul este proiectat pe arhitectura MVC, drept urmare avem 3 clase important: Modelul(Polinomul)+Monomul, Vizualizarea/View si Controlul. Fiecare clase conține metodele sale proprii care vor fi explicate mai jos.

1.*Monom*

In clasa Monom se regăsesc gradul si coeficientul încapsulate, împreuna cu setterii si getterii acestora.

In Monom exista una dintre cele doua metode toString care va folosita ulterior pentru modelarea polinomului pentru a fi transmis ulterior interfeței grafice. Metoda pregătește un String gol care in funcție de exponentul si coeficientul monomului modifica șirul de caractere. Pentru exponentul 0, se va transmite mai departe doar coeficientul, pentru exponentul 1 se va testa daca coeficientul este 0,1,-1 sau o valoare diferita de cele menționate anterior si in funcție de caz, se va modela Stringul cu “0”,“x”, “-x” sau “coef \* x”. Pentru exponenți > 1 se va testa din nou valoarea coeficientului: daca e 0, șirul de caractere va rămâne nul, daca e 1 sau –1 se va completa cu “x” respectiv “-x”, iar pentru valori diferite de cele menționate anterior “coeficient \* x ^ putere”.

O alta metoda importanta ce aparține clasei Monom este metoda de multiplicare a nodurilor din lista înlănțuita, adică din polinom. Diferit fata de adunare sau scădere, pentru a efectua înmulțirea fiecare monom din primul polinom trebuie înmulțit cu fiecare monom din cel de-al doilea polinom. Drept urmare, clasa Monom implementează o metoda care testează puterile si coeficienții fiecare monom, iar după cazuri se efectuează înmulțirea (polinomul 1 are grad 0 iar cel de al doilea are grad 1 sau invers – exponentul va rămâne 1; polinomul 1 are coeficient 1 si cel de al doilea are coeficientul 0 – coeficientul rămâne 1; înmulțire normala: gradele se aduna – coeficienții de înmulțesc).

2.*Polinom*

Clasa Polinom conține toate metodele computaționale ale operațiilor, metoda toString pentru transmiterea in GUI, metoda de creare a listei de monoame si lista propriu-zisa, bineînțeles încapsulata.

Metodele de adunare si scădere sunt împărțite in doua: metoda care parcurge polinomul si face adunarea/scăderea in funcție de egalitatea gradelor si metoda de adunare/scădere propriu-zisa, care parcurgând lista apelează metoda de efectuare a calcului. Metoda înmulțire este rupta in doua bucăți: pentru monom si pentru polinom. Metoda Monomului este descrisa mai sus in clasa aferenta, iar metoda clasei curente iterează pe fiecare monom in parte si verifica pozitivitatea si valoarea coeficientului pentru actualizarea polinomului rezultat.

Metodele de derivare si integrare sunt mai simple: se parcurge polinomul si se verifica gradul si coeficientul fiecărui monom, astfel pentru fiecare caz in parte se creste gradul si se împarte cu puterea actualizata( integrare) sau se scade gradul si se înmulțește coeficientul cu puterea anterioara.

Metoda toString a clasei polinom este asemănătoare cu cea a clasei Monom, in funcție de gradul si coeficientul Monomului curent este actualizat un sir de caractere ce va fi transmis pe interfața grafica.

Metoda de GetInput (preluarea intrării) de creare a listei este implementata cu ajutorul expresiilor regulate(REGEX) care definesc un șablon de căutare in șirul de caractere introdus din GUI si despart astfel stringul si îl pregătesc pentru modelarea sub forma de monom, apoi pentru introducerea lui in lista. Funcția recunoaște “+”, “-” pentru despărțirea monoamelor si “x^” pentru recunoașterea puterea si coeficientul acestuia.

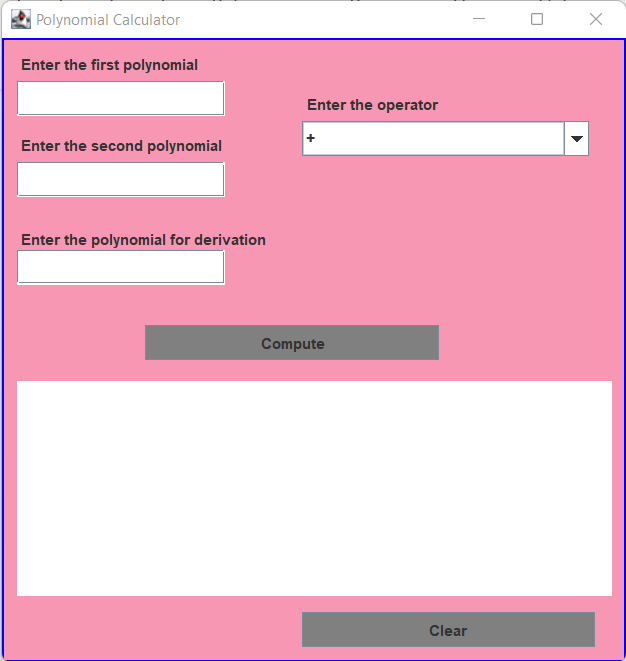
3.*Vizualizarea/View*

In clasa de Vizualizare este modelata interfața grafica si realizata cu biblioteca SWING.

Interfața este implementata sub forma unui JPanel in care exista butoanele de control ale calculatorului si zonele de introducere a polinoamelor calculate.

Pentru polinoame exista 3 JTextFielduri ( 2 pentru polinoame care vor urma sa fie supuse operațiilor de adunare, scădere sau înmulțire si 1 pentru operațiile de derivare si integrare) împreuna cu JLabel care indica atribuirea fiecărui spațiu de completat.

Operația poate sa fie selectata dintr-un JComboBox care împreuna cu butonul de Compute pornește aplicația, iar metodele de calul sunt apelate, iar rezultatul este afisat intr-o zona dedicata JTextArea.



4.*Controlul*

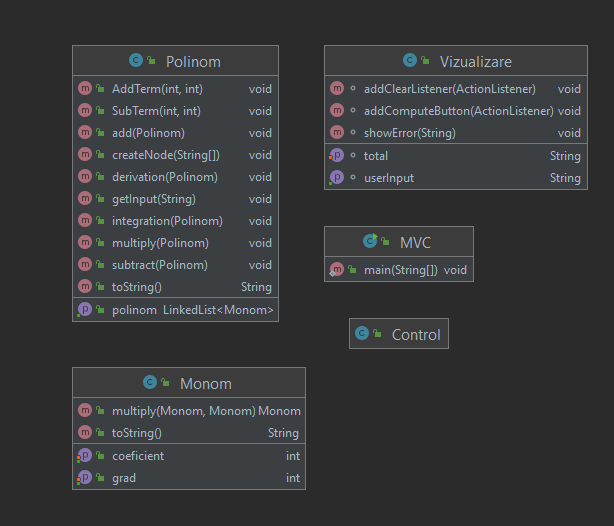
In clasa control sunt definite interfața grafica si lista înlănțuita, adică polinomul, încapsulate.

In interiorul clasei de control sunt definite doua clase interioare care reprezintă butonul de Compute si butonul de revenire in starea inițiala a interfeței, adică ștergerea conținutului de pe toate JTextFieldurile.

Metoda de ActionPerformed atribuie evenimentului acțiunea de computare a polinoamelor in urma apăsării butonului.

Daca este apăsat butonul de Compute se verifica daca cele doua polinoame au fost introduse( pentru adunare, scădere, înmulțire), astfel daca au fost introduse se va alege de pe JComboBox operația, iar in funcție de indexul selectat se va efectua operația dorita. Daca JTextFieldul unde se introduce polinomul pentru derivare/integrare nu este gol si restul sunt goale, atunci se va selecta operația de integrare si derivare si in funcție de index se va realiza calculul.

Daca este apăsat butonul de Clear atunci toate JTextFieldurile sunt șterse, inclusiv zona de rezultat a polinomului.



5.*MVC(clasa principala)*

Clasa principala a programului are metoda statica main unde sunt definite interfața si controlul, de unde interfața poate fi rulata, ii se pot modifica dimensiunile si se poate închide

**REZULTATE**

Pentru verificarea rezultatelor fără a apela interfața grafica am folosit framework-ul JUnit. In clasa de testare sunt definite doua polinoame pentru verificarea operațiilor, împreuna cu constructorul aferent care printează un mesaj pe consola înaintea începerii fiecărui test.

Pentru fiecare operație in parte am făcut doua tipuri de teste:

-adunare/scădere/înmulțire simpla: un test in care se verifica operații pe monoame cu grade diferite: 0, 1 sau 2 si folosirea funcției assertTrue in cazul in care calculul a fost efectuat corect.

-adunare/scădere/înmulțire cu un singur monom de gradul 0: se verifica corectitudinea operației cu termeni liberi ( nu conține "x" in șirul de caractere introdus de la tastatura )

-derivare: un test pe un singur polinom pentru verificarea corectitudinii calculului

-derivare: un test pentru verificarea derivării unei constante

-derivare: un test pe un singur polinom pentru verificarea corectitudinii calculului

-derivare: un test pentru verificarea integrării unei constante

**CONCLUZII**

Construirea unui calculatorul polinomial este un bun exercițiu pentru explorarea bibliotecii dedicate construirii interfețelor grafice si a metodelor butoanelor care preiau informația si o prelucrează in interiorul algoritmilor.

Polinomul reprezintă un exemplu bun si un exercițiu pentru manipularea colecțiilor si metodelor lor( colecția folosita in proiect: LinkedList).

Metoda de creare a nodurilor din lista este un exemplu rudimentar de manipulare a expresiilor regulate si a șirurilor de caractere pentru recunoaștere a unor șabloane.

*Îmbunătățiri pe viitor*

* Adăugarea operației de împărțire a polinoamelor si găsirea unui algoritm eficient
* Îmbunătățirea esteticii a interfeței grafice
* Folosirea unui algoritm de creare a nodurilor din lista mai scurt
* Dezvoltarea calculatorului polinomial pentru monoame cu coeficienți reali, care nu aparțin mulțimii numerelor reale si polinoame de mai multe variabile

**Bibliografie**

* Curs 12 *Programare orientata pe obiecte* – Diagrame UML de clase si obiecte –R. Brehar UTCN <https://moodle.cs.utcluj.ro/pluginfile.php/151029/mod_resource/content/1/POO06.pdf>
* Îndrumător de laborator *Programare orientata pe obiecte -* <https://moodle.cs.utcluj.ro/pluginfile.php/151021/mod_resource/content/1/10-GUI-I.pdf>
* Paradigmele programării orientate pe obiect <https://ro.wikipedia.org/wiki/Programare_orientat%C4%83_pe_obiecte>
* Model de referința pentru operații <https://stackoverflow.com/questions/49788045/polynomial-calculator-using-linked-list>
* Îndrumător de laborator -Testare Unitara – *Programare orientata pe obiecte* https://moodle.cs.utcluj.ro/pluginfile.php/151025/mod\_resource/content/2/12-Testarea\_Unitara.pdf