MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Calculator de polinoame

-Documentație-

Student:
Pop Ruxandra Maria

Univeristatea Tehnică din Cluj-Napoca
Facultatea de Automatică și Calculatoare
Secția Calculatoare și Tehnologia Informației
Anul 2, Grupa 30226

Cuprins

1.Obiectivul temei

1.1.Obiectiv principal

Obiectivul principal al acestei teme de laborator a fost să propunem, să proiectam și să implementam un sistem de procesare a polinoamelor.Mai ușor de zis, un calculator de polinoame ,respectiv de 2 polinoame ,care poate realiza diverse operații ,cum ar fi:adunare,scadere,înmulțire,împarțire,derivare și întegrare,utilizându-se o interfață grafică. Fiind un sistem de calcul,prezintă totodată și date de intrare (introduse de la tastatură) și date de iesire (polinomul rezultat după efectuarea calculelor),care pot fi vizualizate de către utlizator ,sub o formă ușor de interpretat.

1.2.Obiective secundare

Reprezintă pasii care trebuie urmați pentru atingerea obiectivului principal.

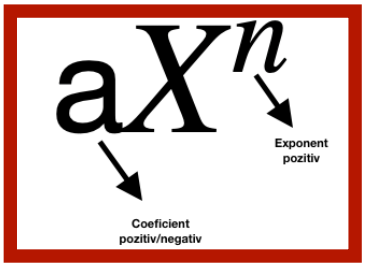
Obiectiv Secundar	Descriere	Capitol
Dezvoltarea de use case-uri și scenarii	Am analizat cine ar putea să se folosească acest sistem de calcul,și care ar fi modurile sale de utilizare .	**2**
Alegerea structurilor de date	Am ales ca structură de date un ArrayList de monom și polinom.	**3**
Impărțirea pe clase	Am ales să impart programul în multiple clase cu nume sugestive astfel încât să fie cât mai ușor de înțeles.Clasele sunt: Main, Regex, Monom, Polinom și clasele pentru realizarea modelului MVC.	**3**
Dezvoltarea algoritmilor	A reprezentat cea mai dificilă parte a proiectului.Pentru a realiza funcționarea corectă a celor 6 operați am incercat multiple variante de proiectare și implementare ,până am ajuns la varianta cea mai corectă și eficienta din punctul meu de vedere.	**3**
Implementarea soluției	Am folosit paradigma OOP,cu multiple clase și metode,care vor fi prezentate mai jos.	**4**
Testarea	Am introdus multiple date de intrare pentru cele 2 polinoame ,încercând să acopăr cât mai multe cazuri posibile.Totodată am realizat și o testare cu JUnit pentru a fi sigură de corectitudinea operațiilor.	**5**

2.Analiza problemei

Pentru a înțelege problema și scopul ei ,este necesar să definim câteva aspecte teoretice importante legate de polinom.

Polinomul este o expresie formată dintr-o multitudine de monoame.

Monomul *are următoarea reprezentare:*



Prin urmare un ***polinom*** *de grad n se scrie în următoarea formă canonică:*

$$P(x)=a_0X^n+a_1X^{n-1}+....+a_{n-1}X+a_n \text{ ,unde } a_0 \neq 0$$

Proprietăți elementare ale polinoamelor:

- ***Suma*** *a două polinoame este un polinom.*
- ***Produsul*** *a două polinoame este un polinom.*
- ***Derivata*** *a două polinoame este un polinom.*
- ***Primitiva*** *a două polinoame este un polinom.*

Pentru a implementa un calculator ,există o varietate de soluții și metode ,în ceea ce privește introducerea datelor ,aspectul polinomului rezultat în urma operației ,precum și aspectul interfeței grafice care este una de tipul "User Friendly" , ușor de înțeles și de utlizat de către orice utilizator.

În ceea ce priveste interfața grafică se poate observa că introducerea polinoamelor de catre utlizator se face destul de simplu prin intermediul celor două TextField-uri . Pentru acest lucru am folosit Regex-uri ,care îmi permit să separ din string-ul dat ca input de catre utilizator ,prin interfața grafică ,monoamele , a identifica exponenții și coeficienții fiecarui monom.
Prin intermediul acestui Regex ,utilizatori pot introduce polinoame de orice grad și dimensiune ,calculatorul fiind capabil să le citească și să le interpreteze.

Desi această metodă pare una destul de revoluționară și eficientă ,poate reprezenta și unele dezavantaje :cum ar fi să apară unele erori ,atunci când nu se respectă instrucțiiunile de scriere, întâlnite la nivelul interfeței grafice .De

altfel,utilizatori sunt atenționați în momentul în care introduc un polinom invalid ,printr-o eroare care le va comunica să verifice înstrucțiunile.

Verificarea instrucțiunilor se face prin apăsarea butonului ***Instr*** *,întâlnit în partea din dreapta a interfeței.*

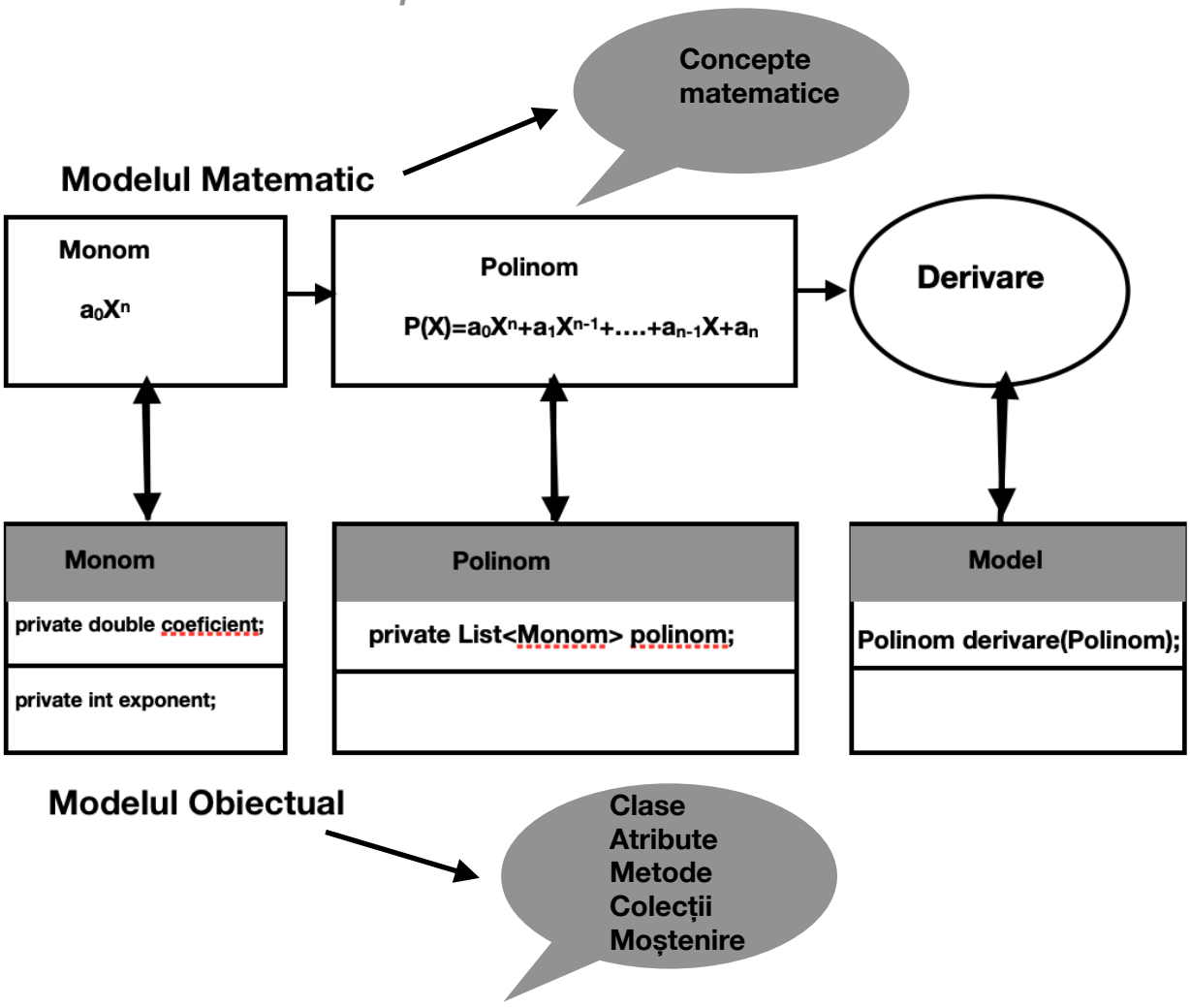
Modelare

Modelarea problemei ne ajută să înțelegem mai bine sistemul pe care dorim să-l dezvoltăm, să găsim un corespondent între realitate și software.

În cazul problemei de față ,modelarea este destul de simplu de realizat ,deoarece domeniul problemei este unul matematic ,iar intre programare și matematică există o conexiune apropiată,strânsă.

Astfel,în cadrul calculatorului software, am transpus din modelul matematic ,monomul și polinomul descrise mai sus , în clase cu nume simbolice ,și cu relații matematice similare. Astfel ,polinomul conține o listă de monoame ,după cum am precizat și mai sus; operațiile au fost implementate prin metode care respectă principiile matematice .

Denumirea sugestivă și intuitivă a claselor ,inspirate din domeniul matematic, ajută la o mai bună înțelegere și interpretare a lor ,precum și la definirea atributelor corespunzătoare.



Cerințe de funcționare

Identificarea cerințelor de funcționare constituie un element critic în dezvoltarea unui sistem software.

Înainte să ne apucăm de proiectarea și implementarea propriu zisă a sistemului , este necesar să cunoaștem ce cerințe trebuie acesta să indeplinească ,pentru a descoperii ce trebuie să facă sistemul ,cum trebuie să funcționeze ,pentru a stii ce obiective dorim să atingem la sfarșitul implementari.

În continuare voi prezenta ,sub forma unui tabel ,cerințele sistemului și constrângerile,pe care le-am identificat în urma analizei problemei.

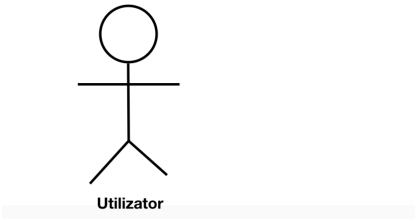
	Descriere
Cerințe funcționale	-Calculatorul de polinoame trebuie să permită utilizatorilor să introducă polinoame -Calculatorul de polinoame trebuie să permită utilizatorilor să selecteze operația matematică dorită -Calculatorul de polinoame trebuie să adune,să scadă,să înmultească,să determine impărțirea a două polinoame , precum și derivata ,primitiva unui polinom ,prin apăsarea butonului specific pentru fiecare operație -Calculatorul de polinoame trebuie să anunțe utilizatorul în cazul unei erori ,care poate fi scrierea greșită a polinomului/polinoamelor, sau impărțirea la 0 -Calculatorul de polinoame trebuie să prezinte un buton prin care utilizatorul poate reseta calculatorul ,precum și un buton prin care utlizatorul poate consulta instrucțiunile de funcționare ale calculatorului
Cerințe non-funcționale	-Calculatorul de polinoame trebuie să fie intuitiv și ușor de folosit -Calculatorul de polinoame trebuie să afiseze rezultatul într-un mod cât mai interactiv și ușor de citit și înțeles -Calculatorul de polinoame trebuie să permită alegerea tipului de operație (binare sau unare)
Constrângeri	-Exponentul fiecarui monom trebuie să fie număr intreg și pozitiv -Coeficienții monoamelor să fie numere intregi negative sau pozitive -La scrierea polinoamelor trebuie să se respecte un format (Pattern)

Diagrama de use case *(în cazul operatiei de scadere)*

Diagrama de use case prezintă o colecție de cazuri de utilizare și actori.

În cazul acestei teme :

- ***actorul este utilizatorul,cel care foloseste calculatorul.***



- ***cazurile de utilizare în cazurile operației de adunare sunt:***
- ***În caz de succes:***

-utilizatorul introduce cele 2 polinoame în interfața grafică.
-utilizatorul face click pe butonul ***“+“.***
-calculatorul polinomial efectuează adunarea celor 2 polinoame și se afisează rezultatul în textField-ul Result .

- ***În caz de nesucces:***

-utilizatorul introduce 2 polinoame invalide (din punct de vedere al formatului scrieri,nu respectă indicațiile din Instr)
-calculatorul va genera o eroare
-se revine la primul pas din cele de succes

- ***cazurile de utilizare în cazurile operației de scadere sunt:***
- ***În caz de succes:***

-utilizatorul introduce cele 2 polinoame în interfața grafică.
-utilizatorul face click pe butonul ***“-“.***
-calculatorul polinomial efectuează scaderea celor 2 polinoame și se afisează rezultatul în textField-ul Result .

- ***În caz de nesucces:***

-utilizatorul introduce 2 polinoame invalide(din punct de vedere al formatului scrieri,nu respectă indicațiile din Instr)
-calculatorul va genera o eroare
-se revine la primul pas din cele de succes

cazurile de utilizare în cazurile operației de înmulțire sunt:

- ***În caz de succes:***

-utilizatorul introduce cele 2 polinoame în interfața grafică.

-utilizatorul face click pe butonul ““.*

-calculatorul polinomial efectuează înmulțirea celor 2 polinoame și se afisează rezultatul în textField-ul Result .

- ***În caz de nesucces:***

-utilizatorul introduce 2 polinoame invalide(din punct de vedere al formatului scrieri,nu respectă indicațiile din Instr)

-calculatorul va genera o eroare

-se revine la primul pas din cele de succes

- ***cazurile de utilizare în cazurile operației de împarțire sunt:***

- ***În caz de succes:***

-utilizatorul introduce cele 2 polinoame în interfața grafică.

-utilizatorul face click pe butonul “/“.

-calculatorul polinomial efectuează împarțirea celor 2 polinoame și se afisează rezultatul în interfață.Se vor afișa două polinoame Cât și Rest .

- ***În caz de nesucces:***

-utilizatorul introduce 2 polinoame invalide (din punct de vedere al formatului scrieri,nu respectă indicațiile din Instr)

-utilizatorul introduce polinomul 2 ca fiind 0

-calculatorul va genera o eroare

-se revine la primul pas din cele de succes

- ***cazurile de utilizare în cazul operației de derivare sunt:***

- ***În caz de succes:***

-utilizatorul introduce un polinom în interfața grafică.

-utilizatorul face click pe butonul "δ".

-calculatorul polinomial efectuează operația de derivare a polinomului și se afisează rezultatul în textField-ul Result .

- ***În caz de nesucces:***

-utilizatorul introduce un polinom invalid (din punct de vedere al formatului scrieri,nu respectă indicațiile din Instr)
-calculatorul va genera o eroare
-se revine la primul pas din cele de succes

- ***cazurile de utilizare în cazul operației de integrare sunt:***

- ***În caz de succes:***

-utilizatorul introduce un polinom în interfața grafică.

-utilizatorul face click pe butonul "∫".

-calculatorul polinomial efectuează operația de integrare a polinomului și se afisează rezultatul în textField-ul Result .

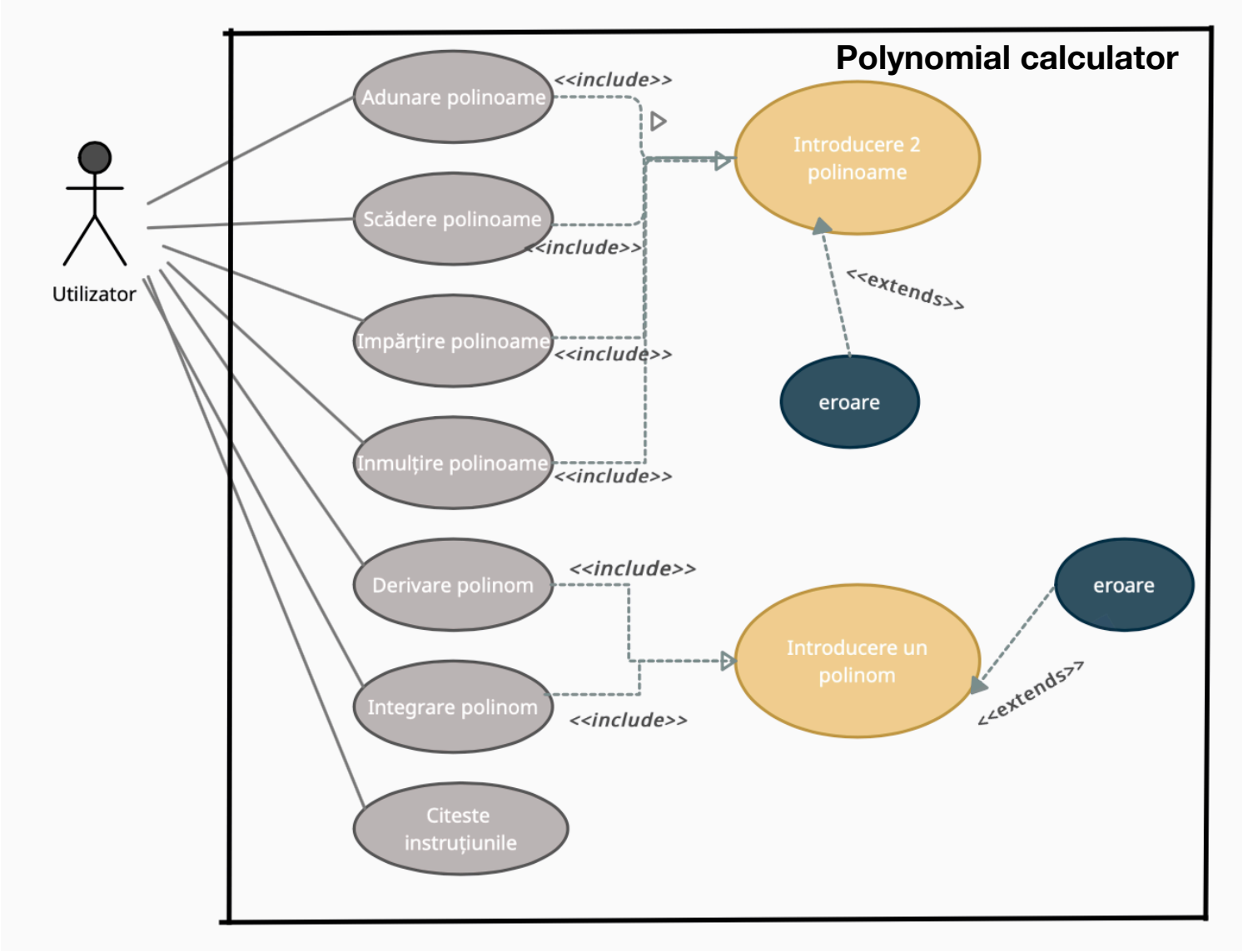
- ***În caz de nesucces:***

-utilizatorul introduce un polinom invalid (din punct de vedere al formatului scrieri,nu respectă indicațiile din Instr)
-calculatorul va genera o eroare
-se revine la primul pas din cele de succes

cazurile de utilizare în cazul citirii instrucțiiunilor sunt:

- ***În caz de succes:***

-utilizatorul selectează butonul Instr. din interfață .După care apasă ok pentru a reveni la calculator.



După cum se poaate observă diagrama conține actorii sistemului, cazurile de utilizare și relațile dintre ei.

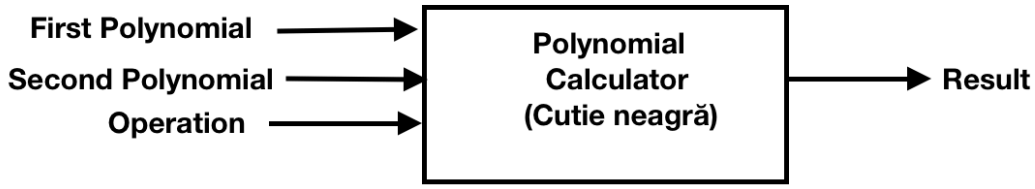
3.Proiectare

3.1.Etapa de design

Prezintă mai multe nivele:

- Design level 1

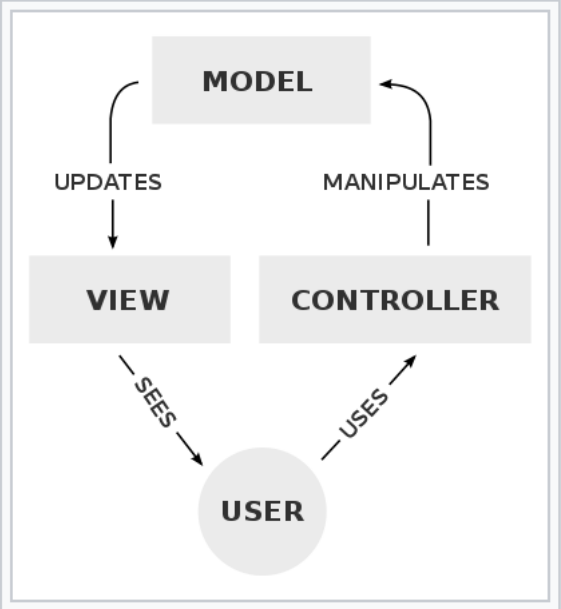
În prima fază,am realizat o schemă bloc a sistemului de calcul,pentru a avea o privire de asamblu asupra proiectului.



După cum se observă ,asupra cutiei negre se transmit date de intrare (introduse de către utilizator),iar prin intermediul operațiilor(metodelor) care se află în interiorul ei, se va genera un rezultat.Această cutie neagră reprezintă o imagine globală a sistemului care urmează să fie implementat.
În acest moment avem o viziune mai clară a ceea ce avem de construit, și anume un calculator care efectuează diferite operații pe doua polinoame, și care trebuie sa genereze acelasi răspuns mereu ,dacă datele de intrare nu se schimbă niciodată.

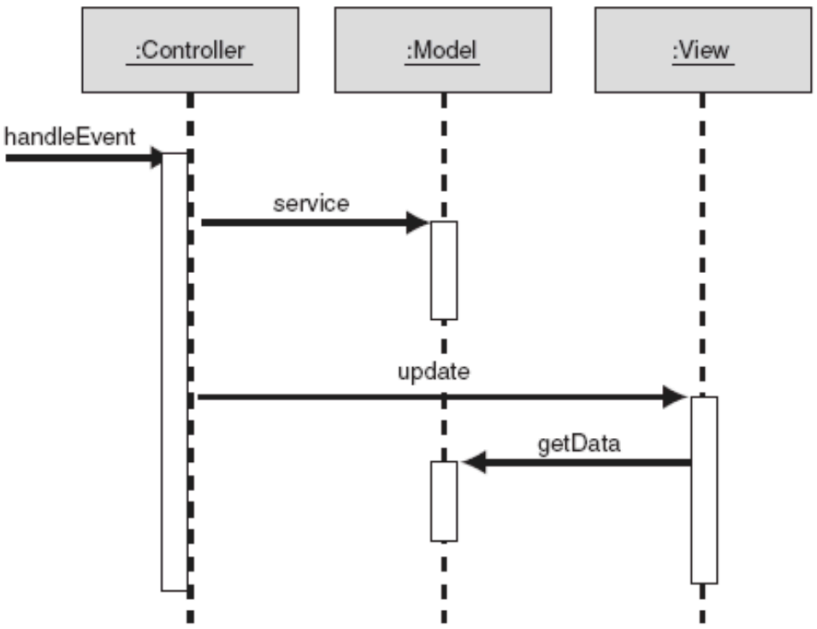
- Design level 2

În a doua etapă ,am împărțit calculatorul în 3 subsisteme care reprezintă modelul arhitectural MVC(Model View Controller).Acesta ne permite să dezvoltăm,să implementăm și să testăm fiecare parte a programului independent ,menținând codul organizat. Însemnând un cod mai eficient și o modalitate mai bună de reutilizare a modelului.



! Între aceste 3 subsisteme există o relație de legatură strânsă.

View-ul transmite evenimentele care se petrec la nivelul interfeței grafice (ex: alegerea tipului de operație care se va efectua) ,controller-ului care le interprezetează,și care interacționează cu model-ul.Model-ul actualizează noile date , după care acestea vor fi preluate de către view ,pentru a face modificările necesare la nivelul interfeței.

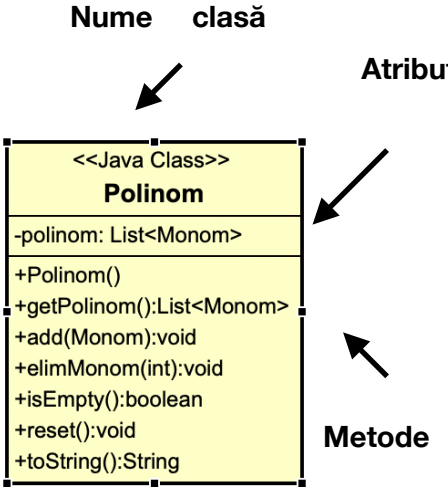


Explicarea modului de funcționare pe un exemplu concret.

Utilizatorul selectează tipul de operație dorit.În acest moment View transmite un semnal către Controller care va permite efectuarea acestor tipuri de operații.
După care utilizatorul introduce polinomul,și apașă butonul "δ".Acum View va transmite din nou un semnal către Controller ,acestă interpretând faptul ca utilizatorul dorește să efectueaze operația de derivare și transmite mai departe , către Model, polinomul citit din interfață și ii comunică faptul că dorește derivarea lui.Modelul efectuează operația de derivare și setează valoarea rezultatului ,pe care View o preia cu scopul de o afișa pe interfață in field-ul Result.

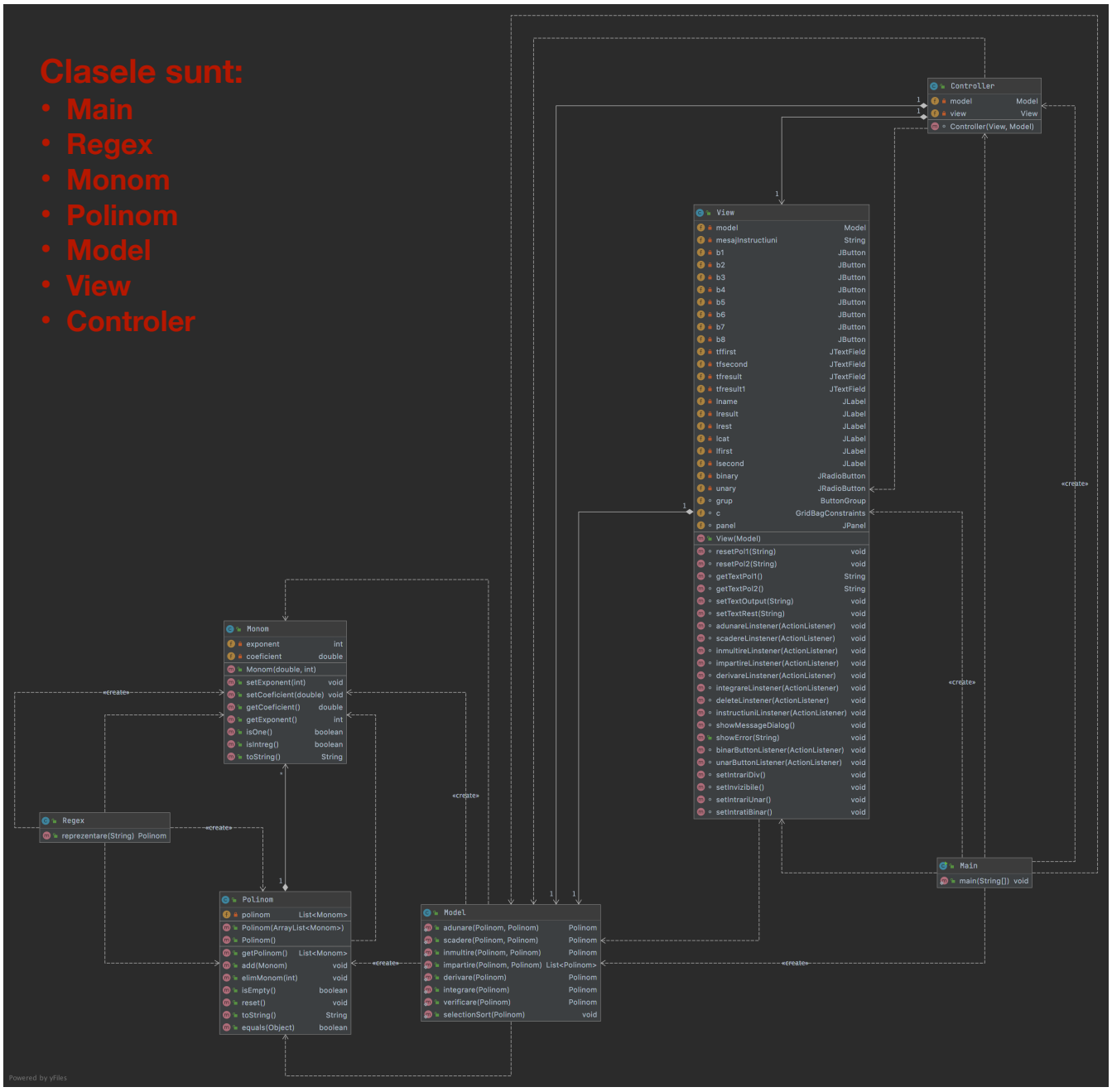
- Design level 3

In ceea de a 3 etapă, se definesc clasele ,metodele și cum interacționează între ele. Clasele și metodele au nume sugestive și simbolice inspirate din domeniul matematic,după cum am precizat și mai sus.
Având toate aceste detalii am realizat diagrama UML.



UML-Unified Modeling Language.

Reprezintă un set de clase,interfețe,colaborări și alte relații.



3.2.Structuri de date

Am decis ca în acest proiect să folosesc ca și structuri de date colecțile.

Colecția : orice clasă care păstrează obiecte și implementează interfața Collection.

Și anume am folosit pentru a reprezenta polinomul ,un ArrayList de monoame ,ceea ce mi-a ușurat destul de mult partea de implementarea a operațiilor de calcul, deoarece această colecție implemetează toate metodele din interfața List.

Pentru a o folosi am importat java.util.ArrayList.

```
public class Polinom {
    private List<Monom> polinom;

    public Polinom( )
    {
        polinom = new ArrayList<Monom>( );
    }
```

3.3.Algortimi folosiți

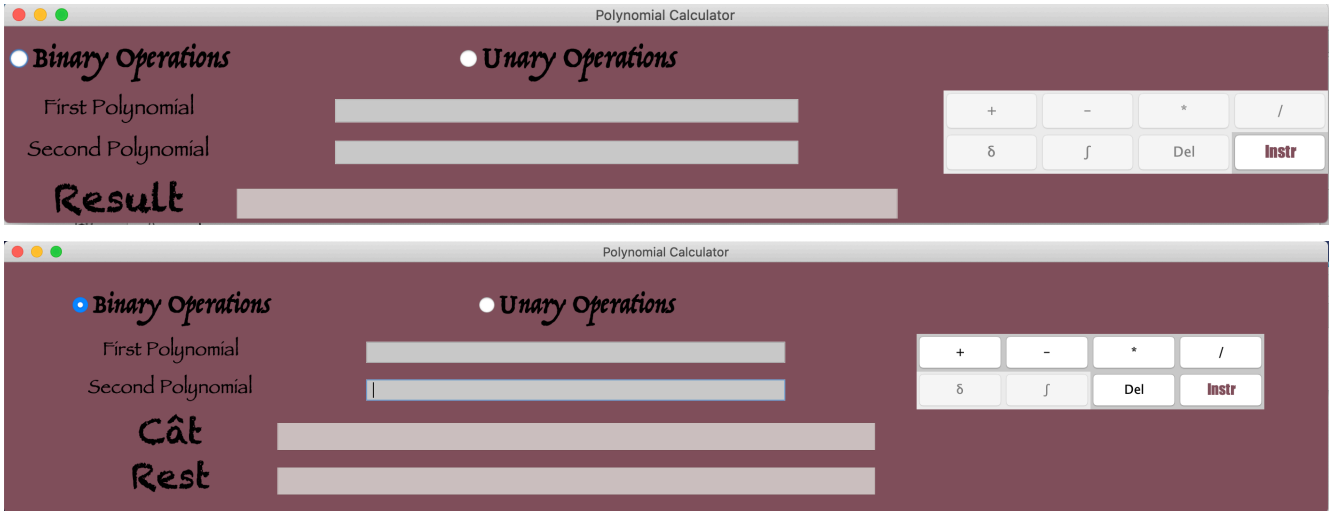
Operațiile de adunare,scădere,impărtire sunt inspirate după algortimul de interclasare.Iar celelalte operații (înmultire,derivare,integrare) ,se fac prin intermediul parcugeri polinomului/polinoamelor cu ajutorul for-each-ului, neavând la bază un algoritm fundamental. Am folosit și algoritmul selection sort , pentru a-mi ordona polinomul în ordinea descrescătoare în funcție de exponentul fiecărui monom.Puteam folosi și Collections.sort() ,dar pe moment uitaserăm și nu am mai modificat ulterior.:)

```
public static  void selectionSort(Polinom rezp1){
        for(int i=0;i<rezp1.getPolinom().size()-1;i++){
            for(int j=1+i;j<rezp1.getPolinom().size();j++){
                if(rezp1.getPolinom().get(i).getExponent()<rezp1.getPolinom().get(j).getExponent()){
                    int m1=rezp1.getPolinom().get(i).getExponent();
                    double m2=rezp1.getPolinom().get(i).getCoeficient();
                    rezp1.getPolinom().get(i).setCoeficient(rezp1.getPolinom().get(j).getCoeficient());
                    rezp1.getPolinom().get(i).setExponent(rezp1.getPolinom().get(j).getExponent());
                    rezp1.getPolinom().get(j).setExponent(m1);
                    rezp1.getPolinom().get(j).setCoeficient(m2);
                }
```

3.4.Interfața grafică *(Graphical User Interface)*

Reprezintă un mecanism prietenos pentru interacțiunea utilizatorului cu programul.

Folosim o interfață User-Friendly pentru a permite utilizatoriilor să se simta mai familiarizați cu programul chiar înainte de a-l fi utilizat. Printr-o interfață utilizatorul poate înțelege mai bine cum funcționează programul ,poate învăța mai repede modul de utilizare a acestui calculator.



Interfața grafică cuprinde următoarele componente:

Panou *- care cuprinde toate elementele și a cărui titlu este setat ca fiind "Polynomial Calculator".*

Butoane *- sunt in număr de 8 ,din care 6 sunt reprezentate de cele 6 operații, și 2 de operațiile de stergere și pentru afișarea instrucțiiunilor.*

-2 butoane de tip Radio, folosite pentru alegerea tipului de operație .
Binary- pentru operații efectuate pe ambele polinoame.
Unary- pentru operații efectuate pe un singur polinom.

TextField-uri *-spațiile în care se pot introduce date de la tastatură sau se pot afisa rezultatele.*

-sunt in numar de 4 :
-pentru First Polynomial.
-pentru Second Polynomial.
-pentru Result/Cât.
-pentru Rest.

Label *- este o etichetă .Sunt in numar de 6.*

!În main se instantiază interfața grafică , deci de fiecare dată când este rulat main se formează o noua interfață.

4.Implementare

Clasele folosite sunt destul de ușor de înteles și interpretat.

1.Clasa Main

Este clasa în care spunem programului ce să execute.Sunt declarate 3 obiecte unul de tip Model ,unul de tip View care are ca argument un model ,și unul de tip Controller care are ca argumente un view și un model.Care vor determina deschiderea interfeței grafice unde se intamplă toate interacțiunile utilizatorului cu sistemul de calcul.

```
public class Main {
    public static void main(String[] args) {

        Model model = new Model();
        View view = new View(model);
        Controller controller = new Controller(view, model);
        view.setVisible(true);


    }
}
```

view.setVisible(true); — va face interfața vizibilă

2.Clasa Regex

Preia din datele de tip String introduse de catre utilizator ,date utile și igonoră restul textului. Avem un Pattern (tiparul după care ar trebui să fie introduse polinoamele) pentru forma polinomului și unul pentru forma monomului.Dacă acest tip de reprezentare nu este respectat la introducerea celor doua polinoame ,vor apărea erori .Mai multe informații despre cum ar trebui scrise polinoamele , se pot găsi la apăsarea butonului Instr intâlnit în cadrul interfeței.

```
public class Regex {

    public Polinom reprezentare(String t) {

        String polinomFormat = "([+-]?[0-9]*[Xx]?\\^?[0-9]*)";
        String monomFormat = "([+-]?[0-9]*)([Xx]?)\\^?([0-9]*)";
        Pattern patern = Pattern.compile(polinomFormat);
        Matcher mat1 = patern.matcher(t);
        Polinom p = new Polinom();
```

M-am inspirat pentru realizarea ei de pe stackoverflow

3.Clasa Monom

Prezintă două atribute ,și anume coeficient care este de tip double și exponent care este de tip int .Acestea vor fi instanțiate prin constructorul Monom. Pentru fiecare dintre aceste două atribute există câte două metode ,o metodă de tip set ,și una de tip get.Mai există și metodele de tip boolean care mă ajută în realizarea metodei toString() care oferă o reprezentare cât mai corectă și exactă a fiecarui monom. Metoda toString() prezentă aici , este construită în funcție atât de valoriile exponentului ,cât si de cele ale coeficientului , pentru ca polinomul rezultat , să arate cât mai realist în interfață.

```
public class Monom {

    private int exponent;
    private double coeficient;

    public Monom(double coeficient, int exponent) {
        this.exponent = exponent;
        this.coeficient = coeficient;
    }
```

4.Clasa Polinom

Prezintă un singur atribut ,o colecție de tip ArrayList de Monoame.Constructorul instanțiază colecția.
Avem o metodă de get care va returna polinomul format.
O metodă de add care va adauga câte un nou monom în listă(în polinom).
O metodă de eliminare a unui monom ,care ne va fi de folos în implementarea operației de impartire. De asemenea este prezentă și metodă toString() ,care va ajuta la afisarea polinomului în urma efectuari operatiilor.
Totodată aici am suprascris și metoda equals(Object a) ,necesară în realizarea testelor cu JUnit , mai multe detalii despre această metodă se găsesc în capitolul 6.

```
public class Polinom {
    private List<Monom> polinom = new ArrayList<>();

    public Polinom(ArrayList<Monom> polimon) {

        this.polinom = polimon;
    }

    public Polinom() {

        polinom = new ArrayList<Monom>();
    }
```

Pentru realizarea interfeței am abordat modelul MVC(model view controller) .

5.Clasa Model

În această clasă am ales să implementez cele 6 tipuri de operații necesare realizării calculatorului.

- ***Metoda adunare(Polinom a,Polinom b)** - realizează adunarea a doi polinomi .Argumentele sunt Polinom a,Polinom b.*

Parcurgem fiecare polinom ,fiecare monom din polinomul a și fiecare monom din polinomul b.Dacă intălnim monoame ce au același exponent, se formează un nou monom (rezm) prin adunarea coeficienților și păstrarea exponentului . Acest nou monom se adaugă intr-un polinom (rezp) care la sfarșitul metodei va fi returnat.Dacă se întâlnesc monoame cu grad diferit ,în rezp se adaugă monomul cu grad cel mai mare .La final după ce unul din contori a ajuns la dimeniunea maximă a polinomul ,primul while se oprește. Celelate while-uri au ca scop adăugarea in rezp a acelor monoame care nu au reusit sa fie adăugate prin intermediul primului while. La sfarsitul funcției aplic pe polinomul rezultat funcția verificare() // acestă funcție va fi explicată mai jos.

- ***Metoda scadere(Polinom a ,Polinom b)*** *-realizează scaderea a doi polinomi. Argumentele sunt Polinom a,Polinom b.*

Aceasta metodă este asemânătoare cu metoda de adunare() descrisă mai sus ,doar că in primul while in loc să se efectueaze adunarea coeficienților celor două monoame ,se efectuează scăderea lor.Astfel noul monom obținut în while va fi format din scăderea coeficienților , și păstrarea exponentul . La sfarșitul funcției aplic pe polinomul rezultat funcția verificare().

- ***Metoda inmultire(Polinom a,Polinom b)****-realizează înmulțirea a doi polinomi.Argumentele sunt Polinom a,Polinom b.*

Parcurgem fiecare polinom , și înmulțim fiecare monom cu fiecare monom ,adică înmulțim coeficienții și adunăm exponenții intr-un nou monom rezm ,pe care îl vom aduga intr-un polinom provizoriu p1(pentru a avea monoamele in ordinea descrescatoare a gradului) .Înainte sa se ajungă din nou la primul for ,în rezp se face adunarea cu polinomul provizoriu. La sfarșitul funcției aplic pe polinomul rezultat funcția verificare().

- .***Metoda impartire(Polinom a,Polinom b)****-realizează împarțirea a doi polinomi.Argumentele sunt Polinom a,Polinom b.*

Aceasta metodă realizează împarțirea clasică a celor doi polinomi .În polinomul a păstrez restul împarțiri la fiecare pas , până când nu mai conține nici un monom sau până când gradul lui este mai mic decât gradul polinomul b.În rezp se pastrează câtul operației.Aici nu abordez cazul în care gradul polinomului b este mai mare decât a polinomului a ,deoarece am ales să fac acest lucru separat în Controller ,pentru a-mi fi mai usor la afișarea în interfață a rezultatului. De asemenea cazul în care polinomul b este 0 , se abordează tot in Controller.După cum se observă această metodă returnează o listă de polinoame ,care va avea mereu dimensiunea 2 având pe prima poziție restul iar pe cea de a doua câtul.Acest lucru face ca afisarea in înterfață sa fie mai ușor de efectuat. (Singura metodă carea returneaza o listă de polinoame)

- ***Metoda derivare(Polinom a)****-efectuează derivarea unui polinom.Are un singur argument ,deoarece derivarea se face pe un singur polinom ,în cazul nostru pe Polinom a.*

Se parcurge polinomul,pentru fiecare monom setăm coeficientul ca fiind produsul dintre vechiul coeficient și exponentul lui , iar exponentul îl decrementăm cu 1.Si acest polinom se adaugă în rezp . La sfarsitul funcției aplic pe polinomul rezultat funcția verificare().

- ***Metoda integrare(Polinom a)-****efectuează integrarea unui polinom.Si această metodă prezintă doar un polinom,Polinom a.*

Se parcurge polinomul,pentru fiecare monom setăm coeficientul ca fiind impărțirea dintre vechiul coeficient si exponentul lui adunat cu 1 , iar exponentul il incrementăm cu 1 .Si acest polinom se adaugă in rezp . La sfarsitul funcției aplic pe polinomul rezultat funcția verificare().

- ***Metoda verificare(Polinom rezp) -*** *prezintă doar un argument, Polinom rezp. Aceasta metodă are ca scop verificarea încă o dată a polinomului final .Verificarea constă în faptul că se parcurge incă o dată polinomul pentru a ne asigura că nu mai există monoame de același grad .Se ia primul monom din polinom (m1), și se compară exponentul său cu exponentelele celorlalte monoame prezente în polinom.Dacă se găseste încă un monom cu același exponent ca m1 ,se adună coeficienții lor intr-o variabila numită "coeficient" și se elimină monomul găsit. După ce se verifică m1 cu fiecare monom ,în rezp1 se adugă un monom format din coeficientul calculat anterior și exponentul lui m1.Și tot asa până când se ajunge la finalul polinomului rezp. După terminarea while-ului aplic un selection sort pe noul polinom format , pentru ai ordona monoamele în ordinea descrescatoare a gradului.*

- ***Metoda selectinSort(Polinom rezp1) -*** *este algoritmul fundamental de selectionSort care are ca scop ordonarea monoamelor unui polinom în ordinea descrescatoare a gradelor .În loc de această metodă puteam folosi Colections.sort() deoarece polinomul este o colecție de monoame****.***

6.Clasa view

Este folosită pentru implementarea interfeței grafice,ceea ce vedem noi utilizatori.Prezintă butoanele pentru cele 6 operații plus 2 butoane suplimentare pentru afisarea instrucțiunilor și pentru resetarea calculatorului. Totodată prezintă două butoane de tip JRadioButton pentru a alege ce tip de operație dorim să efectuam (binary-pe doua polinoame sau unary- pe un singur polinom).Mai intălnim aici și numeroase textField-uri care ne permit scrierea polinoamelor/polinomului și afișarea rezultatelor ,precum și etichete JLabel care ne îndrumă unde ar trebui scris fiecare polinom,unde va fi afisat rezultatul operațiilor.Am folosit și GridBagConstraints care mă ajută la aranjarea mai eficientă a componentelor în interfața grafică .Aici se gasesc diferite metode care fac legătura între butoane și controller,precum și metode de afisare ale erorilor , de setare a textField-urilor,de setare a vizibilitați unor butoane la anumite perioade de timp.

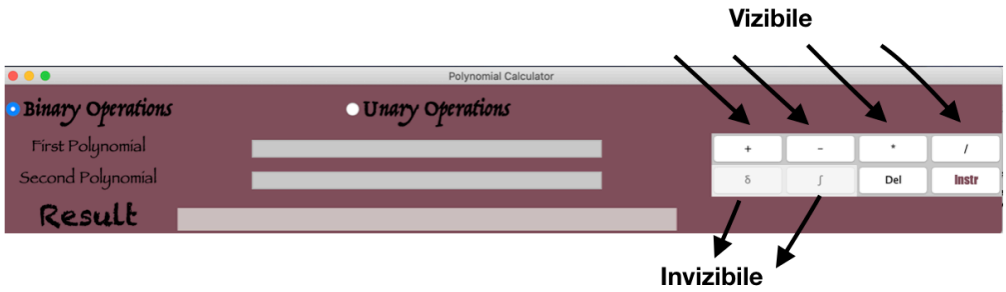
7.Clasa Controller

Are 2 atribute , un model și un view
Traduce interacțiunile utilizatorului cu view-ul, în acțiuni pe care le va efectua modelul.
Fiecare metodă întâlnită aici corespunde fiecarui buton prevăzut în interfață ,și descrie comportamentul programului în urma apăsări unui anumit buton .Astfel aici se găsesc metode pentru fiecare buton .În fiecare metodă se preiau polinomul/ polinoamele ca și un string și se convertesc in Pattern-ul corespunzător nouă prin intermediul metodei din clasa Regex , și anume metoda reprezentare(string input), după care se vor efectuă operația dorită din model.Dacă polinomul citit nu respectă structura Pattern-ului se va returna o eroare.

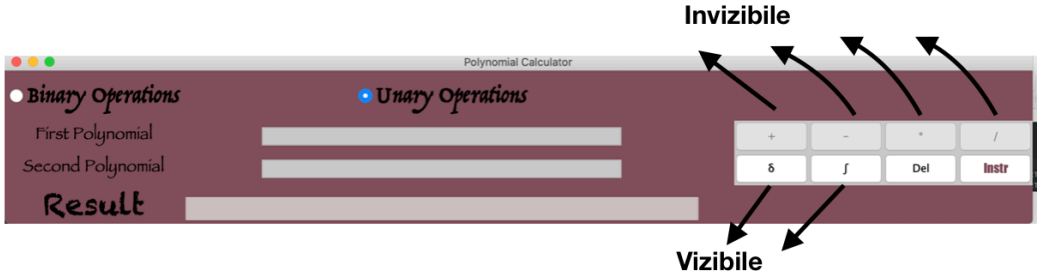
```
Regex r = new Regex();
Polinom a = new Polinom();
Polinom b = new Polinom();
try {
    a = r.reprezentare(view.getTextPol1());
    b = r.reprezentare(view.getTextPol2());
```

Sunt prezente și două metode pentru butoane de tip JRadio ,aceste metode au ca scop setarea vizibilitați/invizibilității anumitor componente din interfață
La început ,când se deschide aplicația ,câmpurile de scriere sunt blocate,până la apăsarea unuia dintre cele două butoane JRadio.

Dacă se apasă butonul “Binary Operations” , se vor putea adăuga cele două polinoame, și se va putea apasă unul dintre cele 4 butoane specifice pentru operațiile binare.



Dacă se apasă butonul “Unary Operations” , se va putea adăuga doar primul polinom, și se va putea apasă unul dintre cele 2 butoane specifice pentru operațiile unare.



! Vizibile=se pot apasă și executa operațiile.
Învizibile =nu se pot apasă și nu se pot executa operațiile.

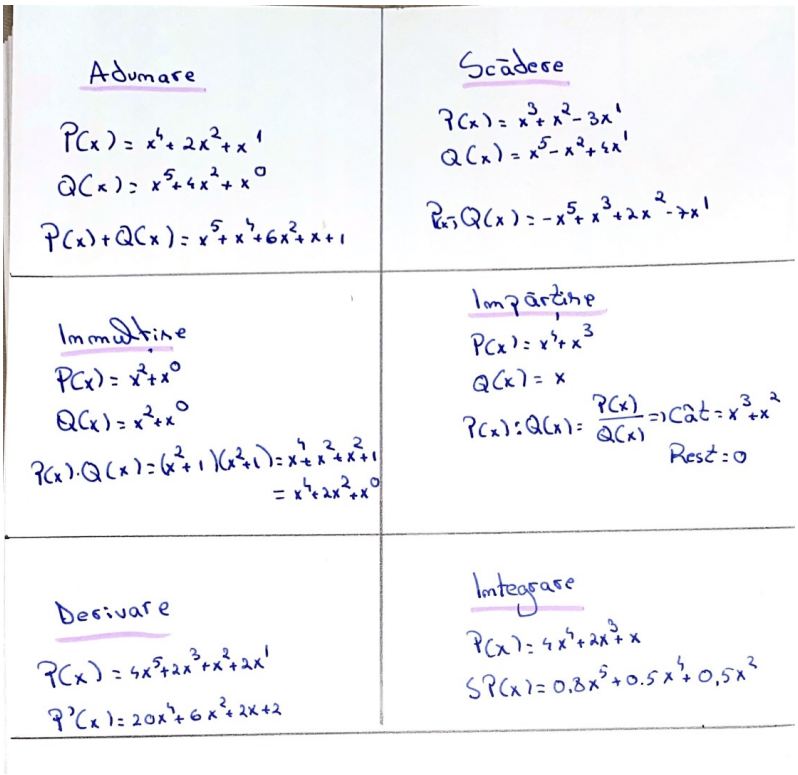
8.TesteUnitare() - aici se vor efectua testele de verificare ale sistemului de calcul.Are 6 metode, una pentru fiecare operație implementată.Mai multe detalii despre această clasă vor fi prezentate în capitolul următor.

5.Rezultate

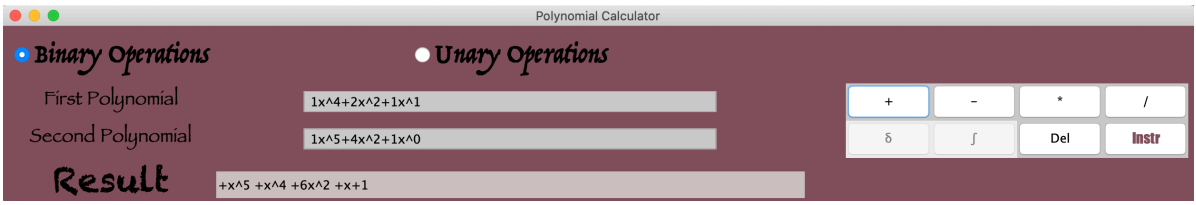
Pentru a testa corectitudinea programului am utilizat două metode:

1.Prima metodă a fost introducerea unor date de la tastatură ,în repetate rânduri pentru a acoperi o varietate cât mai mare de cazuri.
Am calculat pe foaie diferite operații ,pentru diferite polinoame, după care am întrodus polinoamele în calculator ,și am comparat rezultatul de pe foaie cu rezultatul din interfață /consolă. Am testat programul prin această metodă , atât înainte de a implementa interfața grafică , cât și după aia. Am tot efectuat această metodă până am fost convinsă că programul funcționează corect.

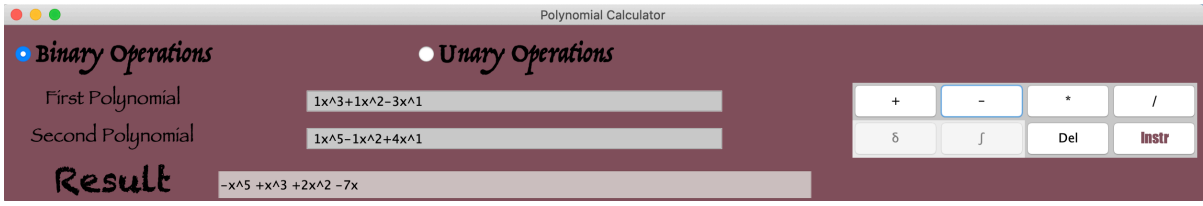
Câteva dintre exemplere pe care le-am introdus.



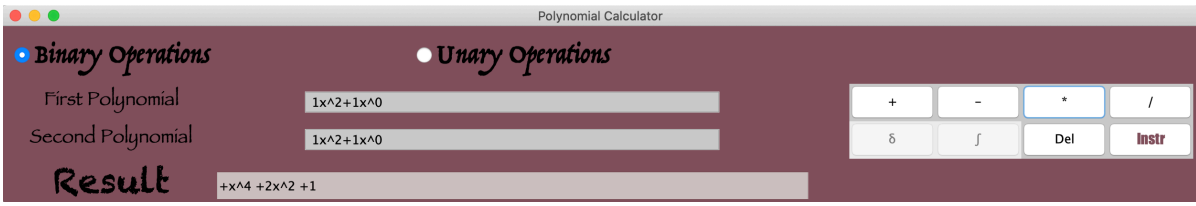
Adunare



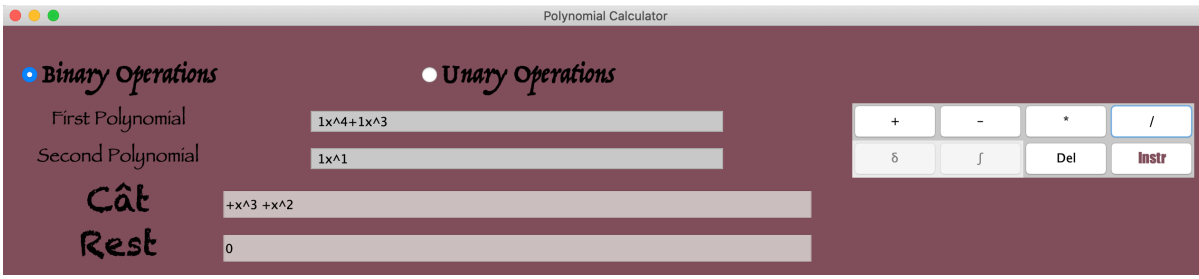
Scadere



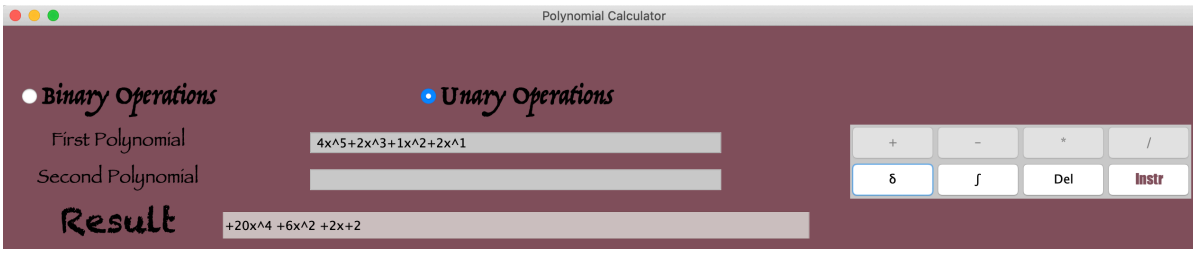
Inmulțire



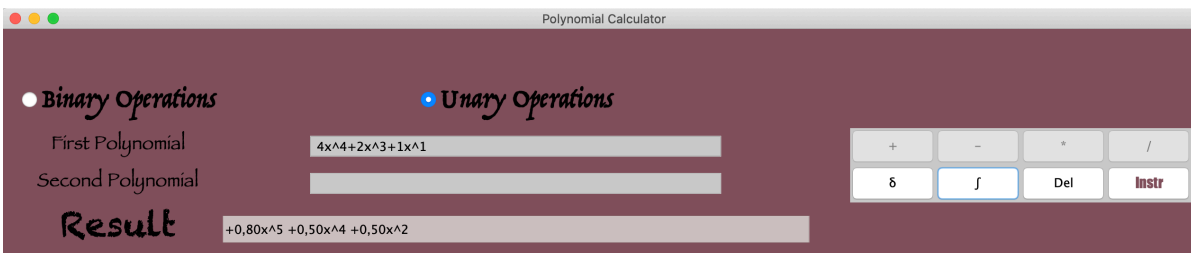
Impărțire



Derivare



Integrare



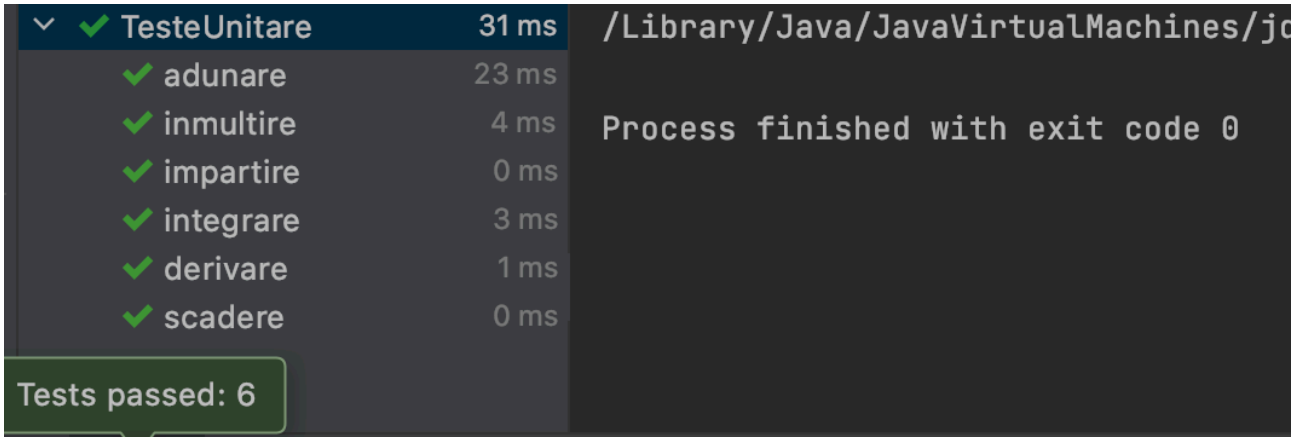
2.A doua metodă a fost testarea cu JUnit.
În această metodă am folosit Assertions , mai exact ***assertEquals.***
Pentru fiecare operație binară am instanțiat câte 3 polinoame(înafară de împărțire unde am instanțiat 4 , deoarece a trebuit să compar și câtul și restul).
Pentru fiecare operație unară am instanțiat câte un polinom.
Polinomul p1 și p2 conștituie polinoamele pe care se vor efectua fiecare operațiie ,iar p3 este rezultatul acestei operații.
Aceste polinoame sunt exemplele din prezentarea suport pentru temă.
Polinoamele le-am inițializat print-ul ArrayLIst de monoame după cum se poate observa.

```
Polinom p1= new Polinom(new ArrayList<Monom>(Arrays.asList(
Polinom p2= new Polinom(new ArrayList<Monom>(Arrays.asList(
Polinom p3= new Polinom(new ArrayList<Monom>(Arrays.asList(
```

Pentru ca metoda assertEquals sa funcționeze corect , am dat ca argumente reprezentarea polinoamelor sub formă de string;

```
assertEquals(p3.toString(),Model.adunare(p1, p2).toString());
```

După cum se poate observa toate testele sunt trecute cu succes.



În varianta finală a proiectului nu au mai aparut erori sau greseli de calcule.Am rezolvat toate erorile găsite și întâmpinate de a lungul proiectului

6.Concluzie

În concluzie, consider că acest proiect mi-a aprofundat cunostințele dobândite până acuma ,în ceea ce priveste limbajul de programare Java și paradigma OPP,precum și capacitatea de a crea o interfață mult mai complexă din punct de vedere al numarului de componente utilizate.

Ca și dezvoltare ulterioară, s-ar putea adaugă în interfața grafică mai multe butoane care să servească la scriere polinomului , la stergerea doar unui caracter, și nu în ultimul rând s-ar putea găsi un Pattern pentru scrierea polinomului ,mult mai eficient ,în sensul să nu trebuiască scris și puterea 0 și 1 în căzul monomului,sau coeficientul 1.

7.Bibliografie

1.https://ro.wikipedia.org
2.https://stackoverflow.com
3.https://www.tutorialspoint.com/index.htm
4.https://www.youtube.com