DOCUMENTATIE

TEMA *1*

NUME STUDENT: Ionas Andreea-Georgiana
GRUPA: 30227

CUPRINS

1. Obiectivul temei

(i) Principalul obiectiv al acestei teme este realizarea unui calculator de polinoame, care sa implementeze operatiile de : adunare, scadere, inmultire, impartire, derivare si integrare a polinoamelor dar si realizarea unei interfete grafice ”user-friendly”.

(ii) Obiectivele secundare:

Realizarea interfetei grafice.	Realizam o interfata usor de utilizat, intuitiva si care sa fie ”appealing” pentru user.
Stabilirea unor structuri de date pe care sa le folosim in proiect.	Implementam clasele ”Monomial” si ”Polynomial” si determinam ce atribute trebuie sa posede.
Implementarea operatiilor pe polinoame.	Implementam pe rand operatiile care trebuie realizate de calculatorul nostru.
Implementarea structurii interne a interfetei grafice.	Ne asiguram ca interfata grafica functioneaza in mod adecvat.
Creearea unui ”pattern” pentru validarea input-ului.	Cream un regex pentru a indentifica un tipar in input-ul dat de utilizator si de a identifica, corect, coeficientii si puterile.

*Toate obiectivele secundare vor fi detaliate in capitolul 4.

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

Cerinte functionale:

- *Calculatorul trebuie sa efectueze operatiile de adunare, scadere, inmultire, impartire, derivare si inetgrare pentru doua polinoame date.*
- *Calculatorul terbuie sa accepte doua polinoame introduse de utilizator.*
- *Trebuie sa validam cele doua polinoame introduse de utilizator.*
- *In functie de butonul apasat, calculatorul trebuie sa efectueze o operatie.*
- *In cazul operatiilor de derivare si inetgrare, calculatorul efectueaza aceste operatii asupra primului polinom.*

Cerinte non functionale:

- *In cazul introducerii unor input-uri gresite, calculatorul va afisa un mesaj de eroare.*
- *Calculatorul ofera raspunsul la operatia aleasa intr-un timp relativ scurt.*
- *Interfata este intuitiva, usor de inteles si utilizat.*

Descriere use-case :
Use case : adunare a doua polinoame
Actor principal : User
Pasii pentru succes:

1) *User-ul introduce primul polinom.*
2) *User-ul introduce cel de al doilea polinom.*

3) *User-ul apasa buton-ul care contine operatia de adunare ”Add”.*
4) *Calculatorul valideaza input-ul primit.*
5) *Calculatorul genereaza rezultatul adunarii celor doua polinoame.*

Scenariu alternativ:

a) *User-ul introduce un input invalid.*
   1) *Calculatorul returneaza un mesaj de eroare.*
   2) *Scenariul se reia de la inceput.*

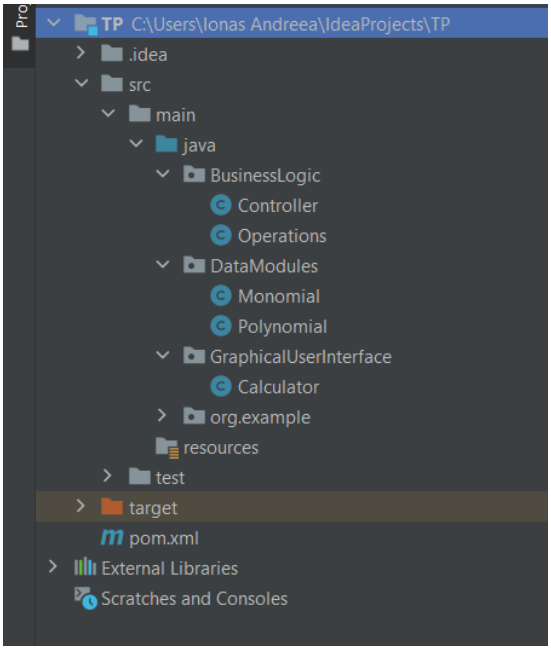
b) *User-ul apasa buton-ul altei operatii.*
   1) *Calculatorul returneaza rezultatul.*
   2) *User-ul apasa butonul operatiei dorite.*
   3) *Calculatorul genereaza rezultatul corespunzator.*

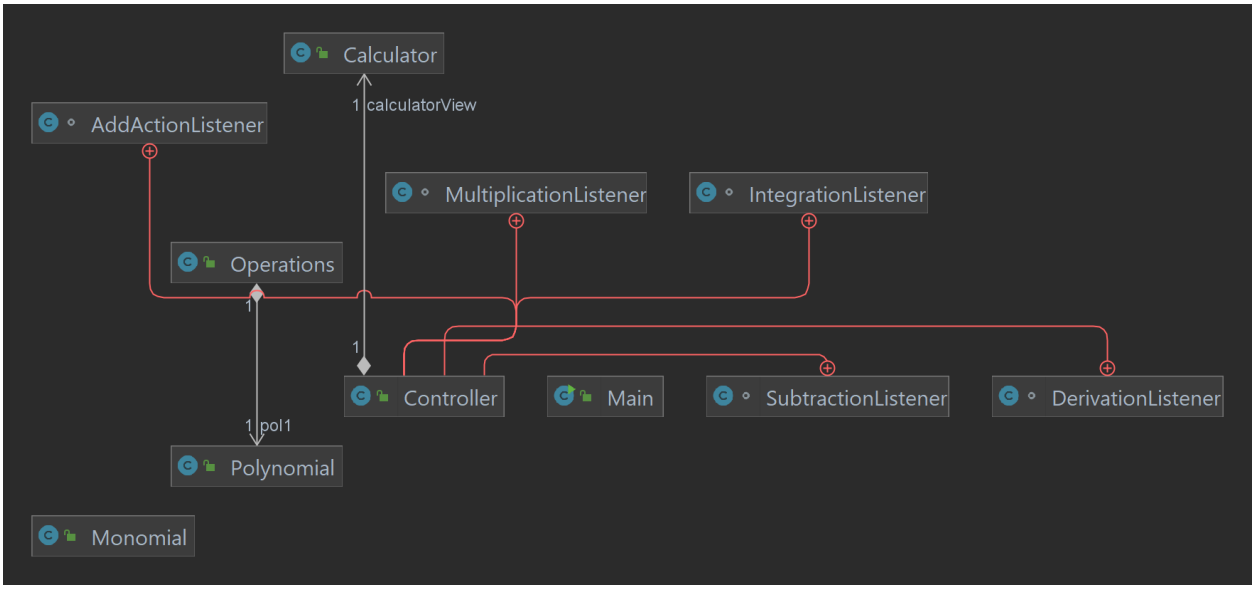
3. Proiectare

Proiectarea OOP a aplicatiei:

Divizarea proiectului in mai multe pachete si clase:

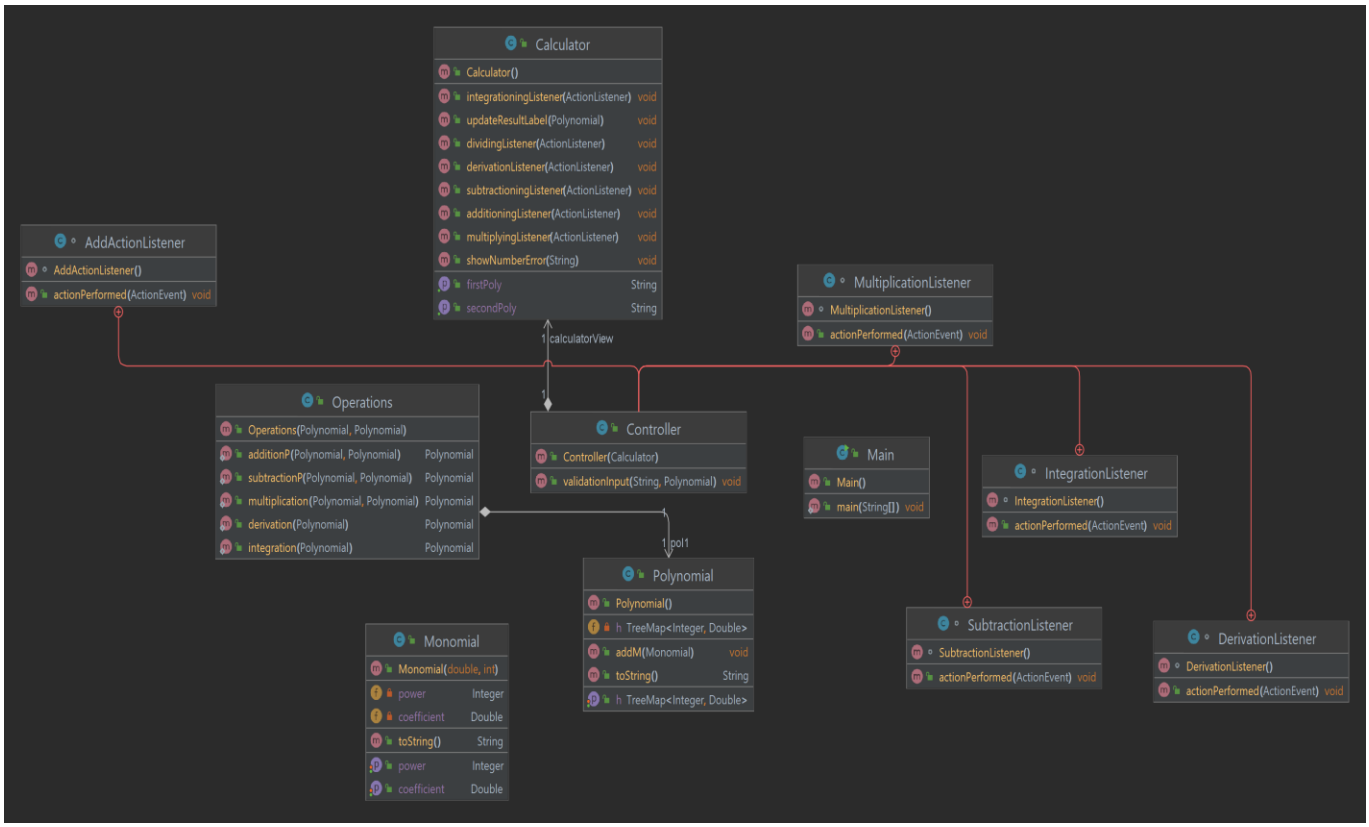
- Pachetul BusinessLogic in care avem clasele Controller si Operations.
- Pachetul DataModules in care avem clasele Monomial si Polynomial (structurile principale folosite in acest proiect).





Dezvoltarea interfetei grafice care este usor de utilizat: se introduc polinoamele in Text Box-ul corespunzator, se apasa unul dintre butoane iar rezultatul este afisat.

Diagrama UML de clase si pachete:



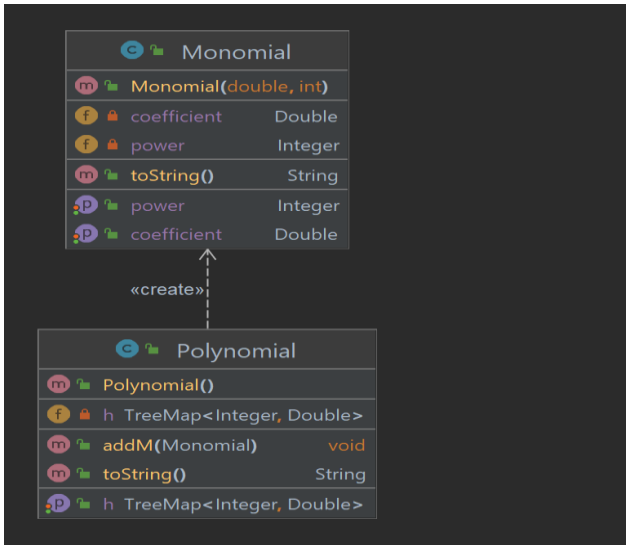
In ceea ce priveste structurile de date folosite avem ”Monomial” si ”Polynomial” ca structuri de baza. Informatii mai in detaliu legate de clase si structuri de baza se gasesc la capitolul 4.Implementare.

4. Implementare

Pentru implementarea acestei aplicatii s-au folosit urmatoarele tehnologii: limbajul de programare Java si Swing pentru realizarea interfetei grafice.

Pentru inceput s-au implementat clasele “Monomial” si “Polynomial”. In clasa “Monomial” s-au declarant atributele “coefficient” si “power” si s-a definit metoda toString si constructorul clasei, pe langa geters si seters.

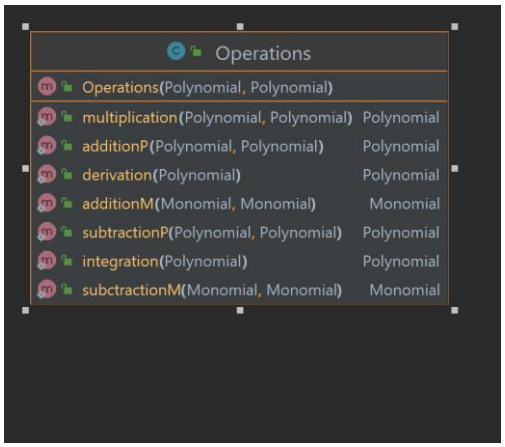
In clasa “Polynomial” s-a declarat un TreeMap in care se vor stoca monoamele. De asemenea, in aceasta clasa avem implementata metoda toString , un constructor si o metoda “addM” care ne permite sa adaugam monoame in TreeMap.

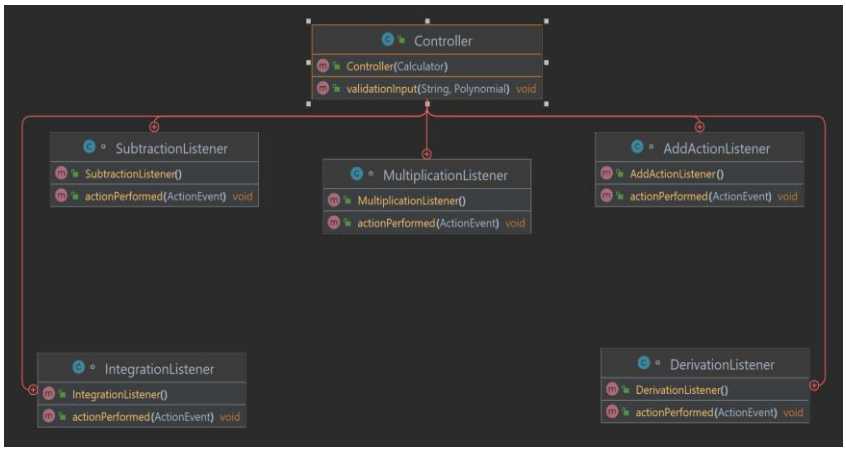


In clasa “Operations” avem implementate metode pentru operatiile de: adunarea monoamelor (additionM), scaderea monoamelor (subctracrtionM), adunarea polinoamelor (additionP), scaderea polinoamelor (subtractionP), inmultirea polinoamelor (multiplication), derivarea polinoamelor (derivation) si integrarea polinoamelor (integration). Derivarea si integrarea sunt operatii care sunt efectuate asupra primului polinom.

In clasa “Controller” s-a implementat o functie puntru validare a input-ului primit de la utilizator (validationInput) si Action Listener pentru butoanele folosite in interfata grafica.

In clasa “Main” s-a instantiat un calculator si un controller

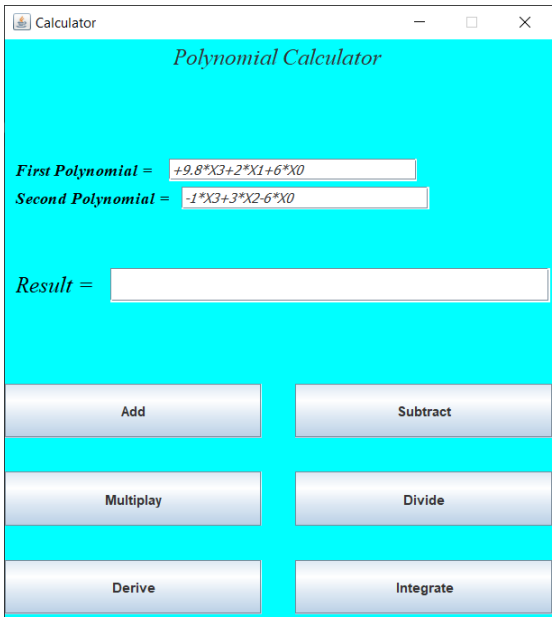




In clasa “Calculator” s-a implementat interfata cu utilizatorul. Interfata consta intr-un frame si un main panel pe care am pozitionat mai multe panel-uri secundare : panel-ul pentru titlu, panel-ul in care se cer si se introduc polinoamele, panel-ul in care se afiseaza rezultatul si panel-ul in care avem butoanele. In panel-uri s-au plasat label-uri, buttons si text box-uri. Avem butoane pentru toate operatiile implementate.

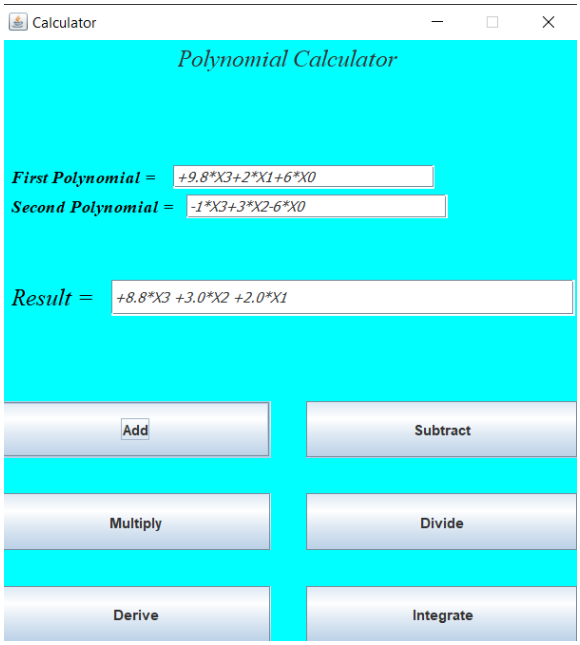
```
public Calculator()
{

    this.frame = new JFrame();
    this.panelMain = new JPanel();
    this.panel1 = new JPanel();
    this.panel2 = new JPanel();
    this.panel3 = new JPanel();
    this.panel4 = new JPanel();
    this.labelResult = new JLabel();
    this.labelPolyOne = new JLabel();
    this. labelPolyTwo = new JLabel();
    this.labelTitle = new JLabel();
    this.textOne = new JTextField();
    this. textTwo = new JTextField();
    this.textResult = new JTextField();
    this.addButton = new JButton( text: "Add");
    this.subtractButton = new JButton( text: "Subtract");
    this.multiplicateButton = new JButton( text: "Multiplay");
    this.divisionButton = new JButton( text: "Divide");
    this.derivationButton = new JButton( text: "Derive");
    this.integrationButton = new JButton( text: "Integrate");
```

La rulalrea programului, apare interfata iar in text box-urile pentru polinoame apare un model, acest model este menit sa ofere utilizatorului un exemplu de cum arata un input valid.

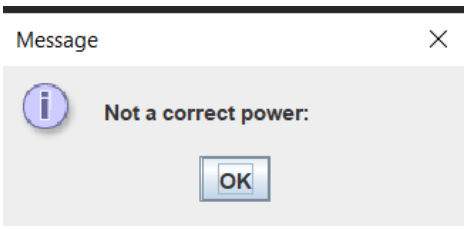


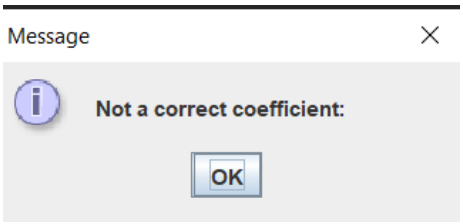
5. Rezultate

In cazul primului scenariu de testare s-au introdus doua polinoame valide si s-a efectuat una dintre operatiile implementate:



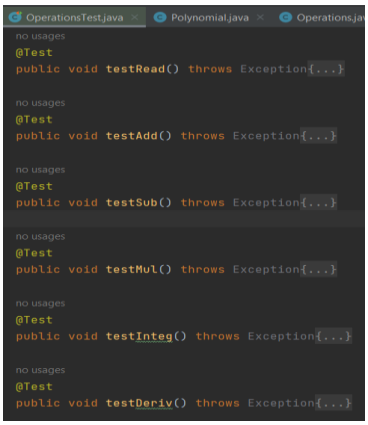
In cel de al doilea scenariu de testare se va introduce un input invalid pentru a vedea raspunsul programului:



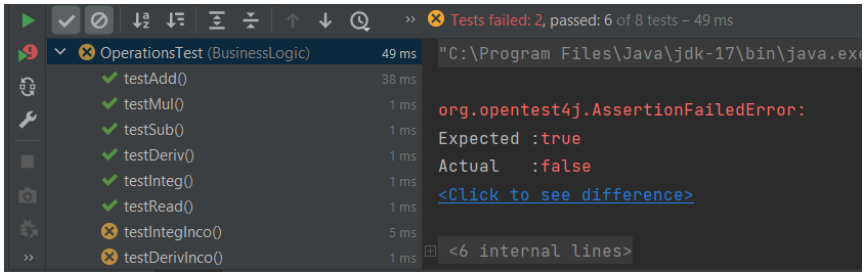


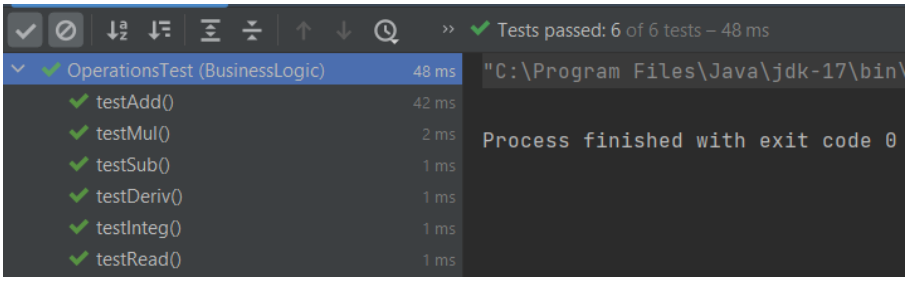
In cazul in care input-ul este invalid la derivare sau integrare calculatorul nu va afisa nici un rezultat.

S-a implementat si testarea cu JUnit, care verifica corectitudinea operatiilor efectuate de calculator:



In cazul introducerilor unor valori incorecte atunci testele vor esua, ca in exemplul de mai jos:





6. Concluzii

In concluzie, aceasta tema ne-a oferit posibilitata de a lucra la un proiect mai amplu care implica mai multe clase, si din care am invatat cum sa "legam" interfata de clasele initiale, cum s-a validam input-ul folosind expresiile regulate(regex) si cum sa lucram cu JUnit.

In viitor la acest proiect s-ar putea dezvolta mai multe operatii pe polinoame si s-ar putea implementa o modalitate de validare a input-ului mai flexibila. De asemenea, s-ar putea implementa o functie de reprezentare grafica a rezultatului.

7. Bibliografie

1. *What are Java classes? - www.tutorialspoint.com*
2. *Diagramele UML - https://www.microsoft.com/en-ww/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling*
3. *Java Swing - https://www.javatpoint.com/java-swing*
4. *Cerinte functionale / non functionale - https://ro.myservername.com/features-functional-requirements*
5. *JUnit - https://junit.org/junit5/docs/current/user-guide/*