UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

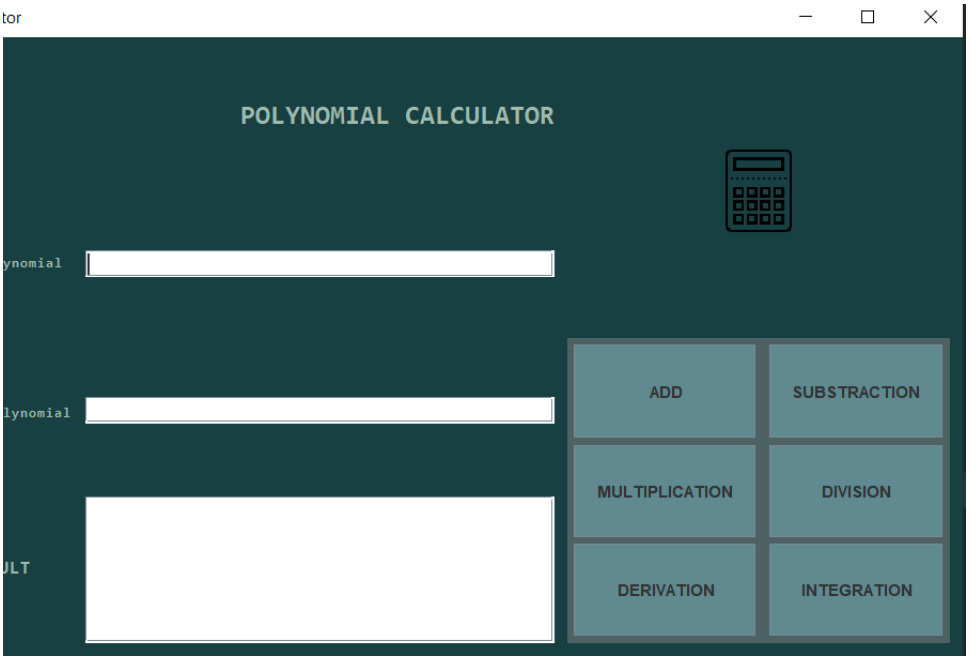
DOCUMENTAȚIE
CALCULATOR DE POLINOAME

Iepure Denisa-Alexandra

grupa: 30223

CUPRINS:

1.OBIECTIVELE PROIECTULUI



Această aplicație de calculator de polinoame este proiectată pentru a oferi utilizatorilor posibilitatea de a efectua operații matematice complexe cu polinoame într-un mod ușor și intuitiv. Interfața grafică a aplicației permite utilizatorilor să introducă polinoamele dorite prin intermediul câmpurilor , urmând ca operațiile selectate să fie efectuate prin intermediul butoanelor de selecție.

Obiectivele secundare(Pașii urmați)

- **Analiza problemei**
  Înainte de a începe să proiectăm și să implementăm calculatorul nostru de polinoame, trebuie să înțelegem problemele și cerințele pe care le vom întâlni. În această etapă , vom avea nevoie de o minte critică , deoarece necesită o abordare logică și rațională pentru a identifica și a înțelege în mod corect cerințele problemei. În timpul analizei problemei, este important să se evalueze informațiile disponibile, să se identifice lacunele sau ambiguitățile și să se dezvolte o înțelegere precisă a cerințelor.

- **Proiectare**

  După identificarea cerințelor, se proiectează interfața grafică a aplicației, astfel încât utilizatorul să poată introduce și gestiona polinoamele. Această etapă trebuie să includă designul butoanelor, meniurilor, câmpurilor de text. Proiectarea algoritmilor fundamentali (adunare, scădere, înmulțire, împărțire,derivare și integrare).
  -

- **Implementarea**

  După ce am proiectat interfața grafică și clasele noastre, putem începe să implementăm codul pentru a face funcțional calculatorul nostru de polinoame.
  Pentru a da viață calculatorului, implementăm interfața ActionListener și vom suprascrie metoda actionPerformed (). Această metodă va fi apelată automat de fiecare dată când se produce un eveniment pe componenta asupra căreia este adăugat ActionListener, în cazul nostru butoanele pentru operații.

- **Testarea**

  După ce am implementat calculatorul nostru de polinoame, trebuie să testăm aplicația pentru a ne asigura că funcționează așa cum este prevăzut. În timpul testelor, ar trebui să ne asigurăm că toate operațiile matematice funcționează

corect și că interfața grafică este ușor de utilizat. Dacă găsim erori sau bug-uri, trebuie să le remediem.

2.ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

Cerințe funcționale

- cerințele funcționale rezultă din analiza problemeii (implementarea unui calculator de polinoame capabil sa realizeze diferite operații)
  - ☐ inserarea polinoamelor de la tastatura , sub formatul pe care funcția de regex îl detectează coef x^exponent ex: x^3+7x+1
  - ☐ selectarea operației matematice
  - ☐ efectuarea operației: adunare, scădere, înmulțire, împărțire, divizare, integrare
  - ☐ afișarea rezultatului
  - ☐ erori pentru gestionarea greșită a aplicației

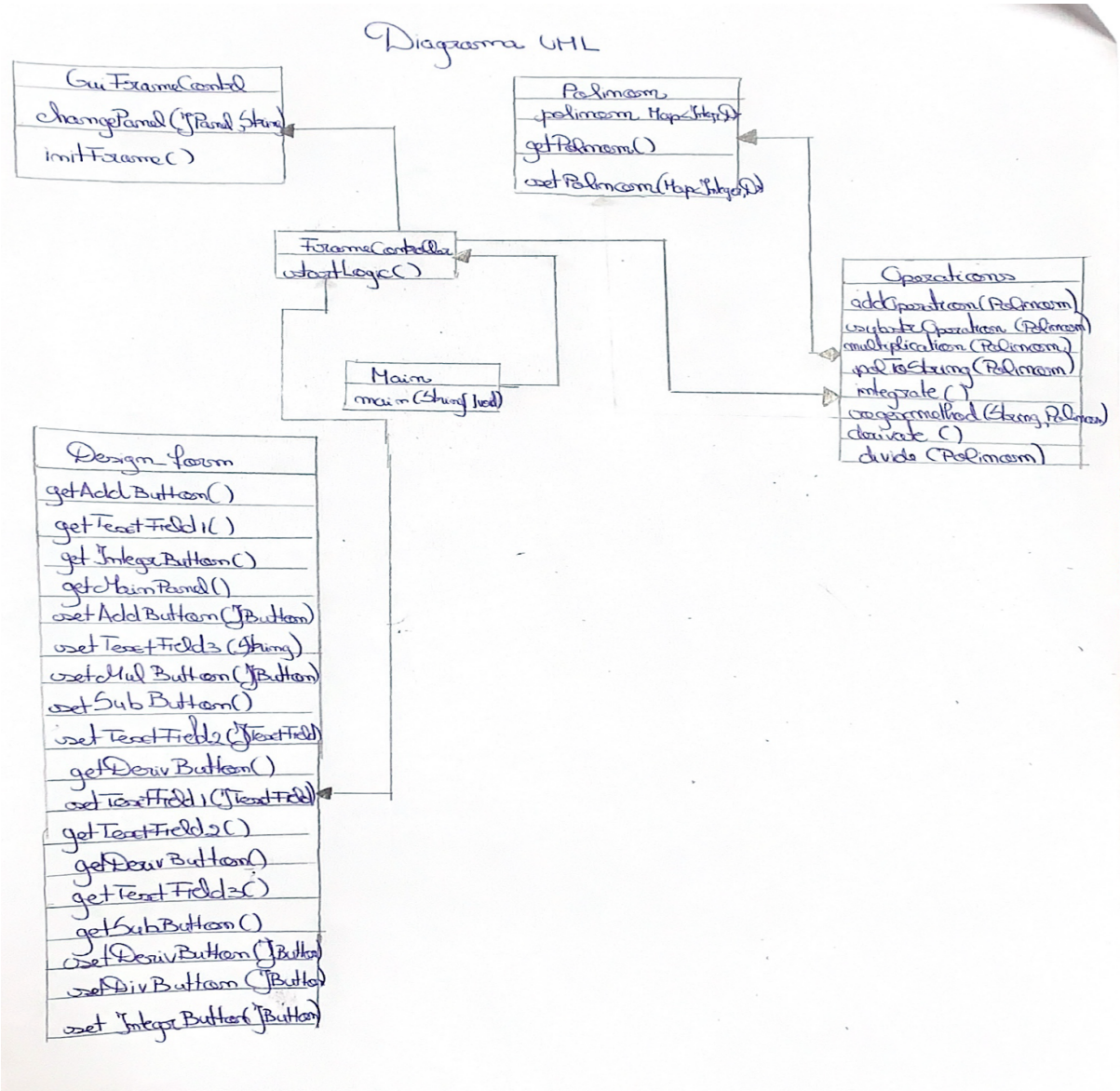
Cazuri de utilizare

- prin intermediul interfeței grafice orice tip de utilizator poate să interacționeze și să se bucure de obiectivul calculatorului, astfel totul în interfața grafică este foarte sugestiv și ușor de utilizat

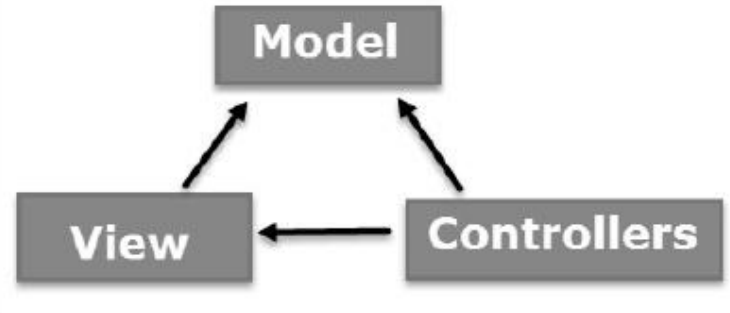
3.PROIECARE

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.

Diagrama UML a calculatorului este formată din 6 clase:Polinom, FrameController, GUIFrameContrl, Operations ,Design_form și Main. Legăturile care s-au creat pe parcursul programului, care fac posibilă funcționarea calculatorului se pot vedea în imaginea de mai jos:



Model-View-Controller este un model arhitectural care separă o aplicație(în cazul nostru calculatorul de polinoame) în trei componente principale: **model, view, controller** . Fiecare componentă este construită să gestioneze diferite părți ale aplicației .



MODEL

- în cazul implementării mele , am ales ca în package-ul Model să includ clasa Polinom , deoarece reprezintă data care este transferată între View și Controller. Propriu-zis reprezintă data care se modifică și care este manipulată.

VIEW

- Acesta este reprezentat de interfața grafică ce interacționează cu utilizatorul . Rolul său este de a evidenția informația obținută până ce ea ajunge la Controller.
- Pentru a construi interfața grafică am utilizat design Swing.Swing este o bibliotecă de interfață grafică pentru utilizator (GUI) pentru platforma Java, care este puternică și versatilă.

CONTROLLERS

- Are rolul de a controla întreaga aplicație. Se pot controla fișiere, scripturi,programe (în general cam orice tip de informație permisă de

interfață). În acest fel putem diversifica conținutul nostru de o formă dinamică și statică, în același timp.

- În acest proiect am implementat clasele FrameController(în general pentru partea de ActionListeners), GUIFrameContrl (pentru frame-ul principal) și clasa Operations unde se manipulează Polinomul prin intermediul metodelor.

4.IMPLEMENTARE

Tot proiectul se bazează pe clasa de bază Polinom, un polinom l-am creat ca fiind de tipul Map, care este un tip de dată care stochează o colecție de date (cheie- valoare), fiecărei chei îi este atribuită o valoare. Dar, l-am implementat ca fiind un TreeMap - ceea ce înseamnă că este sortat după cheie.
Utilizand cheia ca fiind exponentul polinomului și valoarea fiind coeficientul acestuia am gestionat cu usurință aceste două noțiuni.

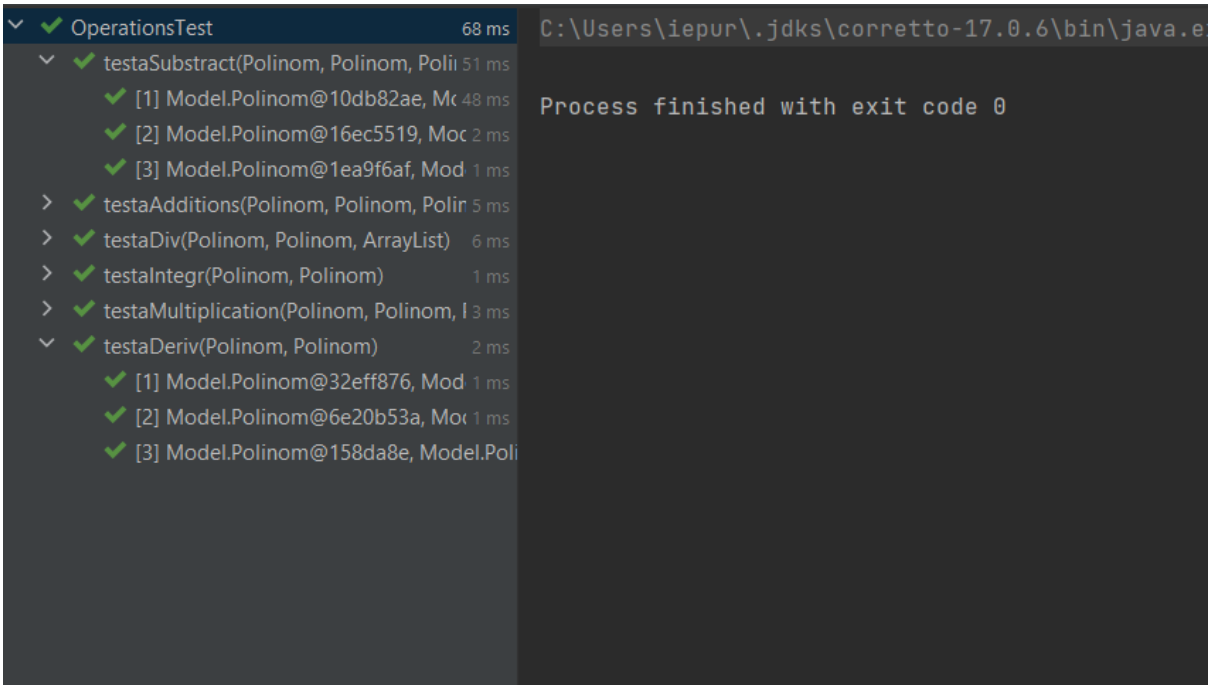
```
polinom= new TreeMap<Integer,Double>(Collections.reverseOrder());
```

Cheile sunt sortate de la cea mai mare , la cea mai mică conform regulilor de scriere din matematică.

Altă clasă importantă, care a fost critică pentru proiect o reprezintă Operations. Aici metoda `regexmethod` și `polToString` fac legătura cu utilizatorul .
Astfel, prin metoda regexmethod caută și validează textul din TextField adică polinomul. În acest mod putem manipula pentru prima data datele introduse de utilizator , astfel încat programul să îl detecteze ca pe un Polinom.
Clasa polToString construiește prin intermediul parcurgerii polinomului un String astfel incat acesta să poată fi afișat.

5.REZULTATELE TESTĂRII

Am implemetat clasa OperationsTest , o clasă destinată testării metodelor. Introducand date corecte am obținut o validare a tuturor metodelor implementate.



5.CONCLUZII

În concluzie, această temă m-a ajutat să gestionez mai bine structura unui proiect, modul în care există o logică de implementare și mai ales cum se lucrează în industrie. Familiarizarea cu Map-urile , cu Design Swing și cu Unit Testing-ul .Toate acestea au ajutat în dezvolatrea abilităților specifice necesare pentru a-l finaliza cu succes.

6.BIBLIOGRAFIE

https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm