DOCUMENTATIE

TEMA 1

NUME STUDENT: Tinca Andrei
GRUPA: 30224

CUPRINS

1. Obiectivul temei 3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3
3. Proiectare 3
4. Implementare 3
5. Rezultate 3
6. Concluzii 3
7. Bibliografie 3

1. Obiectivul temei

- **Obiectivul principal**

  Implementarea unui calculator de polinoame in limbajul de programarea java, cu interfata GUI.

- **Obiective secundare:**
  Analiza problemei, modelare, scenarii, cazuri de utilizare (Capitolul 2)
  Proiectare OOP a aplicației (Capitolul 3)
  Implementare a funcționalităților principale și a interfeței utilizator (Capitolul 4)
  Testare și validare a funcționalităților (Capitolul 5)
  Concluzii și posibile dezvoltări ulterioare (Capitolul 6)

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

- **Scenarii si cazuri de utilizare**

  Pentru a evidenția modul în care utilizatorii vor interacționa cu calculatorul de polinoame, vom defini scenarii și cazuri de utilizare.

  Adunare a două polinoame

  Utilizatorul introduce cele doua polinoame, aplicatia le desparte in monoame, iar sistemul realizeaza operatia de adunare pe monoame, afisand rezultatul in interfata grafica.

  Scăderea a două polinoame

  Utilizatorul introduce cele doua polinoame, aplicatia le desparte in monoame, iar sistemul realizeaza operatia de scadere pe monoame, afisand rezultatul in interfata grafica.

  Înmulțirea a două polinoame

  Utilizatorul introduce cele doua polinoame, aplicatia le desparte in monoame, iar sistemul realizeaza operatia de inmultire pe monoame, afisand rezultatul in interfata grafica.

Derivarea unui polinom

Utilizatorul introduce polinomul, aplicatia il desparte in monoame, iar sistemul realizeaza operatia de derivare pe monoame, afisand rezultatul in interfata grafica.

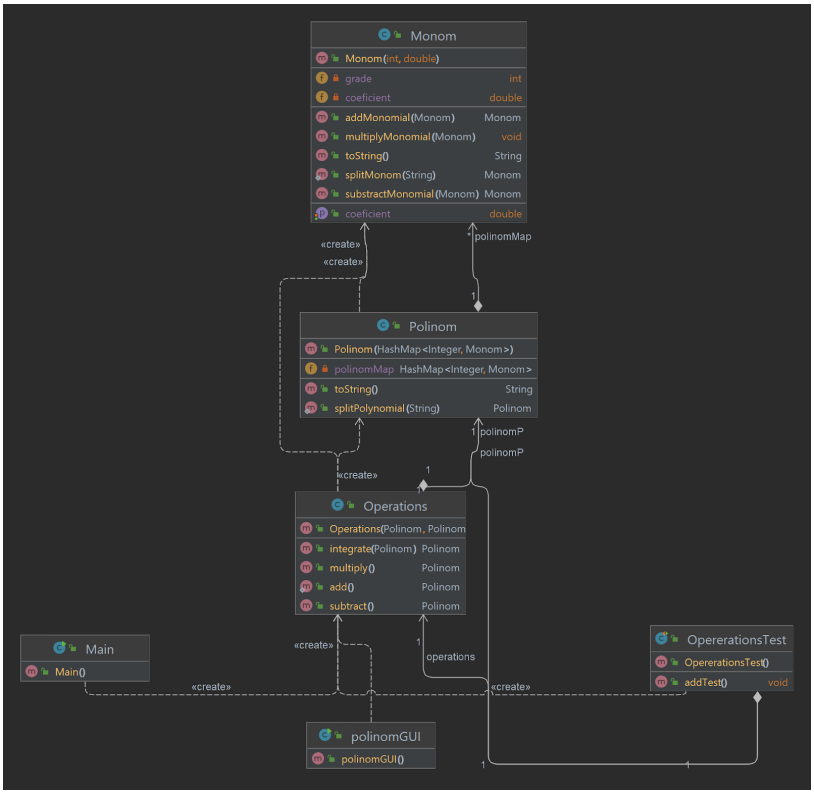
Integrarea unui polinom

Utilizatorul introduce polinomul, aplicatia il desparte in monoame, iar sistemul realizeaza operatia de derivare pe monoame, afisand rezultatul in interfata grafica.

Aceste cazuri de utilizare acoperă operațiile de bază pe care un utilizator le-ar putea efectua cu calculatorul de polinoame și oferă o bază solidă pentru proiectarea și implementarea funcționalităților. De asemenea, acestea vor fi utilizate în testarea și validarea corectitudinii aplicației.

3. Proiectare

- **Diagrama UML de clase a aplicatiei**

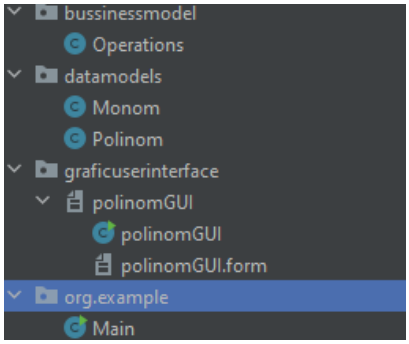


- **Structurile de date folosite:**

Pentru stocarea polinoamelor și a coeficienților lor, ai ales să folosești HashMaps. Acest lucru este o alegere potrivită, deoarece HashMaps oferă acces rapid la elemente și sunt potrivite pentru stocarea datelor în perechi cheie-valoare. De exemplu, cheia ar putea fi exponentul, iar valoarea coeficientul corespunzător.

4. Implementare

- **Descriere generala**



Pentru realizarea aplicatiei au fost create 3 pachete: **bussinessmodel**, pachet in care se afla clasa Operations, unde sunt descriese metodele care implementeaza operatiile pe polinoame, **graficuserinterface,** unde se afla descrierea interfetei grafice, iar **datamodels**, unde se realizeaza parsarea polinomului care este introdus ca sir de caractere, fiind transformat in forma de HashMap cu gradul jucand rolul de cheie, iar valoare fiind de forma Monom, o clasa cu atribute grad si coeficient.

1. **Clasa Operations**

Această clasă este responsabilă pentru realizarea operațiilor matematice pe polinoame, cum ar fi adunarea, scăderea, înmulțirea, derivarea și integrarea.

Metoda add():

Este responsabilă pentru adunarea a două polinoame.
Creează o nouă hartă (HashMap) pentru a stoca rezultatul.
Iterează prin fiecare termen al primului polinom și verifică dacă există un termen corespunzător în al doilea polinom.
Dacă există, adună coeficienții și adaugă noul termen la rezultat.
În final, adaugă termenii neîntâlniți din al doilea polinom în rezultat.

Metoda subtract():

Similară cu metoda add(), dar responsabilă pentru scăderea a două polinoame.

Metoda multiply():

Este responsabilă pentru înmulțirea a două polinoame.
Creează o nouă hartă pentru a stoca rezultatul.
Iterează prin fiecare termen al primului polinom și prin fiecare termen al celui de-al doilea polinom.
Calculează gradul și coeficientul noului termen rezultat din înmulțirea celor două termeni și adaugă rezultatul în map.

Metoda derivate():

Este responsabilă pentru derivarea unui polinom dat.
Creează o nouă hartă pentru a stoca rezultatul.
Iterează prin fiecare termen al polinomului dat și calculează coeficientul și gradul noului termen rezultat din derivarea termenului original.

Metoda integrate():

Este responsabilă pentru integrarea unui polinom dat.
Creează o nouă hartă pentru a stoca rezultatul.
Iterează prin fiecare termen al polinomului dat și calculează coeficientul și gradul noului termen rezultat din integrarea termenului original.

2. **Clasa Polinom**

Această clasă reprezintă un polinom și oferă metode pentru manipularea sa.

Constructorul Polinom(HashMap<Integer, Monom> map): Inițializează un polinom cu un map de monoame.
Metoda statică *splitPolynomial(String polynomialAsString):* Descompune un șir de caractere în monoame și returnează un polinom.
Metoda *getPolinomMap*(): Returnează map-ul de monoame al polinomului.
Metoda *toString*(): Returnează reprezentarea sub formă de șir de caractere a polinomului.

3. **Clasa Monom**

Această clasă reprezintă un monom al unui polinom.

Constructorul Monom(int grade, double coeficient): Inițializează un monom cu gradul și coeficientul dat.
Metoda statică *splitMonom(String monomAsString*): Descompune un șir de caractere reprezentând un monom și returnează un monom.
Metoda *getCoeficient():* Returnează coeficientul monomului.
Metoda *getGrade():* Returnează gradul monomului.
Metoda *setCoeficient(double coeficient):* Setează coeficientul monomului.
Metoda *toString()*: Returnează reprezentarea sub formă de șir de caractere a monomului.
Metoda *addMonomial(Monom x):* Adună monomul curent cu un alt monom dat.
Metoda *substractMonomial(Monom x):* Scade monomul curent cu un alt monom dat.

Aceste două clase formează baza pentru operațiile de manipulare a polinoamelor. Clasa Polinom utilizează clasa Monom pentru a reprezenta fiecare termen al polinomului, iar metodele din clasa Operations efectuează operațiile matematice asupra acestor obiecte.

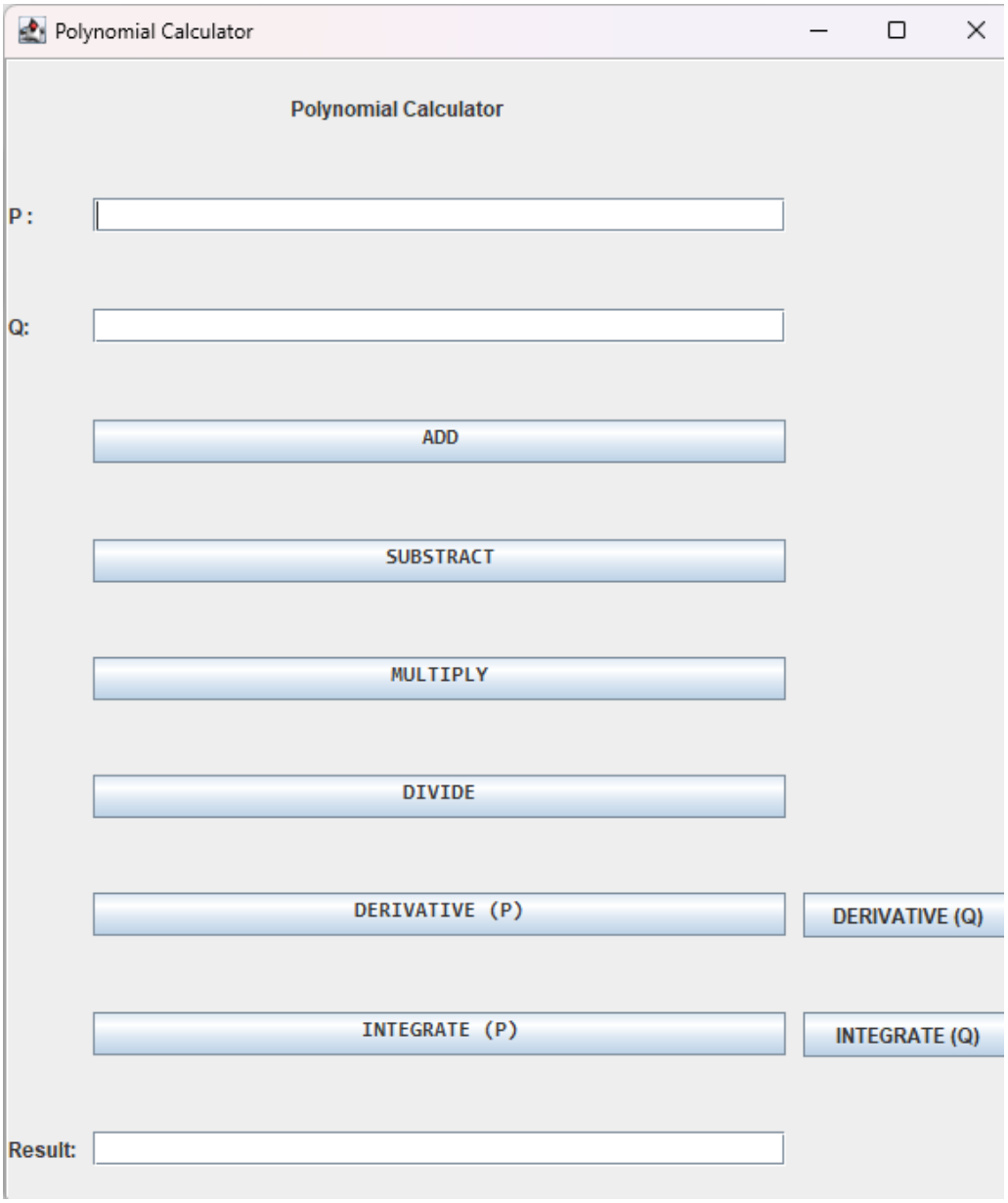
4.Clasa PolinomGui

Clasa polinomGUI este o interfață grafică utilizator (GUI) pentru calculatorul de polinoame. Aceasta este implementată folosind Java Swing.

Interfața constă în câteva componente UI, cum ar fi butoane pentru diferite operații (adunare, scădere, înmulțire, etc.), câmpuri text pentru introducerea polinoamelor de intrare și afișarea rezultatului, precum și câteva alte butoane (potențial pentru operații suplimentare).

Acțiunile butoanelor sunt definite prin adăugarea de ascultători la acestea. De exemplu, atunci când butonul "Adunare" este apăsat, se creează două obiecte Polinom din șirurile de caractere introduse în câmpurile de intrare, apoi se utilizează clasa Operations pentru a efectua adunarea polinoamelor și rezultatul este afișat în câmpul rezultat.

Codul este organizat pentru a utiliza funcționalitățile deja implementate din clasele Operations și Polinom, ceea ce face ca implementarea interfeței utilizator să fie simplă și clară.



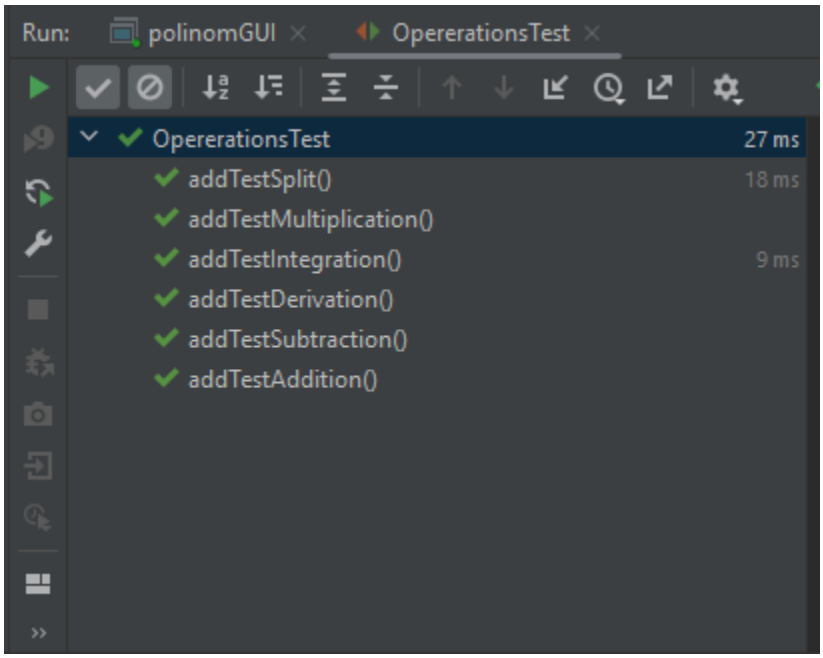
5. Rezultate

Aceste teste validează corectitudinea implementării operațiilor matematice pe polinoame și funcționalitățile asociate din clasa Operations si pentru transformarea din string in HashMap.

Fiecare test este definit într-o metodă marcată cu adnotarea **@Test**. Pentru fiecare operație matematică (adunare, scădere, înmulțire, derivare, integrare) sunt executate teste care validează rezultatul corect și non-corect al operației.

Pentru fiecare test, sunt folosite metode de aserțiune (assertEquals și assertNotEquals) pentru a compara rezultatul obținut cu valoarea așteptată.

Rezultatele in urma testarii cu Junit



6. Concluzii

În concluzie, implementarea unui calculator de polinoame în limbajul de programare Java a fost o experiență instructivă și utilă. Iată câteva concluzii și observații:

Înțelegerea problemelor de matematică: Implementarea acestui calculator a necesitat o înțelegere profundă a conceptelor matematice legate de polinoame, precum adunarea, scăderea, înmulțirea, derivarea și integrarea acestora.

Implementarea algoritmilor: Dezvoltarea operațiilor matematice necesare pentru calculul polinoamelor a implicat aplicarea și implementarea algoritmilor specifici pentru fiecare operație în parte.

Utilizarea structurilor de date și a interfețelor Java: A fost important să folosim corect și eficient structuri de date precum hashmaps pentru stocarea și manipularea termenilor polinomului. De asemenea, am folosit interfețe Java pentru a organiza și abstractiza codul într-un mod modular și ușor de înțeles.

Testarea unitară și asigurarea calității: Implementarea testelor unitare cu JUnit a fost esențială pentru a valida corectitudinea operațiilor matematice și pentru a asigura calitatea codului. Testele unitare ajută la identificarea și remedierea erorilor într-un mod eficient și automatizat.

Dezvoltări ulterioare: Există diverse posibilități de dezvoltare ulterioară a acestui proiect. Acestea includ adăugarea de noi funcționalități precum împărțirea polinoamelor, implementarea unei interfețe grafice mai elaborate sau extinderea calculatorului pentru a manipula și alte tipuri de funcții matematice.

7. Bibliografie

Se vor adauga referintele care au fost consultate de student pe parcursul implementarii temei .
Exemplu:

1. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.*
2. *What are Java classes? - www.tutorialspoint.com*
3. *Curs Programarea Orientata pe Obiect, Autor: Conferentiar universitar Brehar Raluca Didona, UTCN*
4. *https://www.w3schools.com/java/*
5. *https://www.geeksforgeeks.org/java/*
6. *https://regexr.com*
7. *https://regex101.com*
8. *https://www.javatpoint.com/java-string-trim*
9. *https://www.baeldung.com/java-decimalformat*