DOCUMENTAȚIE

TEMA1

Nume student: Buda Andreea Rodica

Grupa: 30221

CUPRINS

1. OBIECTIVUL TEMEI

(i) Obiectivul principal al temei este dezvoltarea unei aplicații software care să permită efectuarea de operații cu polinoame.

(ii) Obiectivele secundare sunt:

Nr.	Obiectivul secundar	Capitol
1.	Analiza problemei și identificarea cerințelor.	Capitolul 2
2.	Implementarea codului sub foram unei structuri de tip Model-View-Controller, implementând clase și subclase.	Capitolul 2
3.	Implementarea operațiilor de adunare, scadere, înmulțire și împărțire a două polinoame.	Capitolul 4
4.	Implementarea operațiilor de derivare și integrare a unui polinom.	Capitolul 4
5.	Crearea interfeței grafice folosind Java Swing.	Capitolul 3
6.	Validarea input-ului utilizatorului pentru evitarea erorilor de input și afișarea unor mesaje de eroare adecvate.	Capitolul 3
7.	Testarea și validarea aplicației, asigurandu-se funcționalitatea corectă a fiecărei operații, inclusiv în cazul de input-uri de dimensiuni mari, folosind JUnit.	Capitolul 5

Documentarea codului este, de asemenea, un obiectiv important al acestei teme, pentru a face codul să fie ușor de înțeles și de urmărit.

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

a. Analiza problemei:

Se dorește implementarea unei aplicații care să permită utilizatorului să efectueze operații matematice pe polinoame cu coeficienți întregi. Aplicația trebuie să permită inserarea polinoamelor, selectarea operației matematice de efectuat (adunare, scădere, înmulțire, împărțire, derivare, integrare) și afișarea rezultatului.

Date de intrare:

- primul polinom

- al doilea polinom

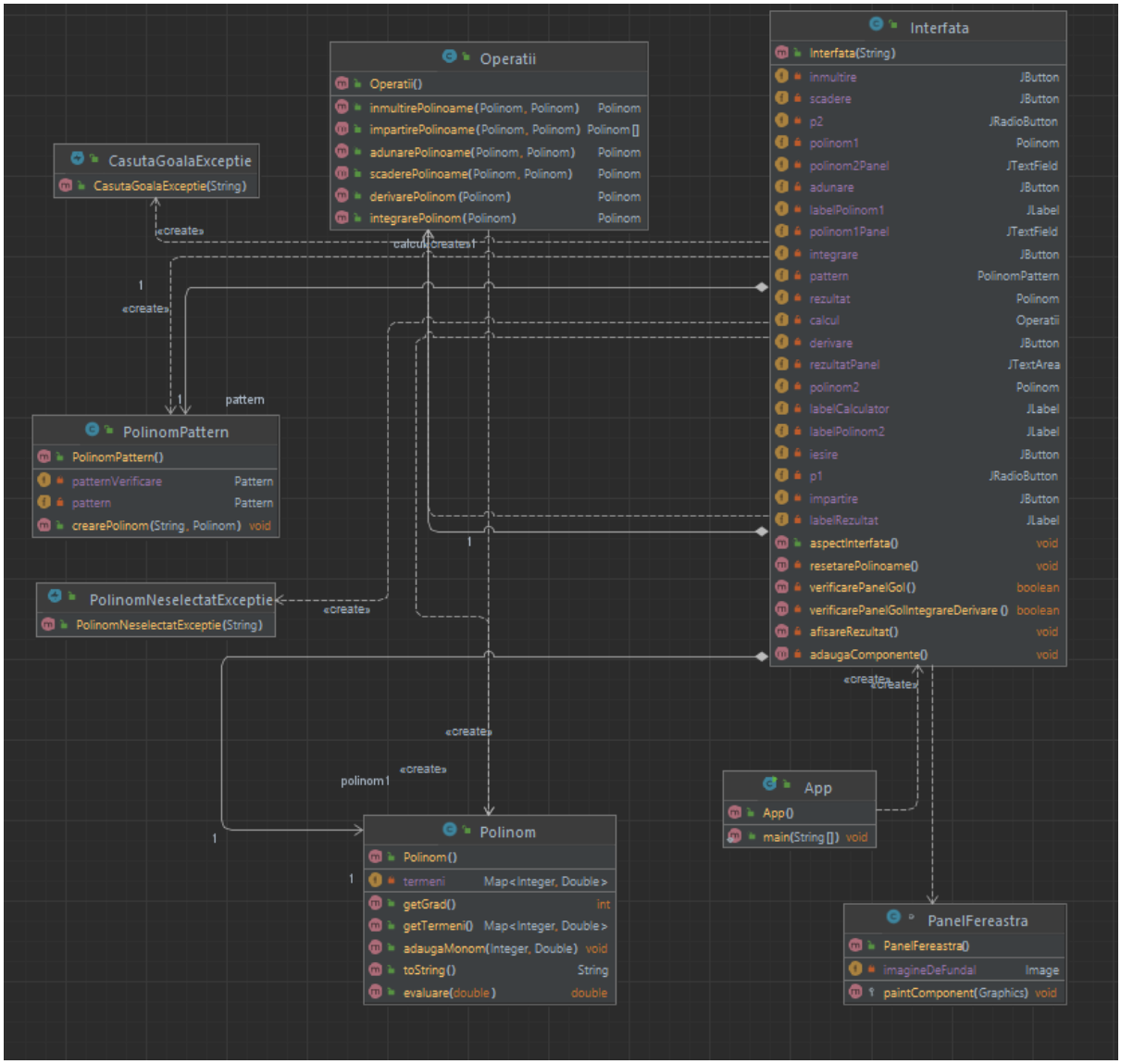
- operația selectată

Date de ieșire:

- polinomul rezultat

b. Modelare:

Diagrama UML:



Clasele sunt organizate în următoarele pachete:

- exception - conține clasa CasutaGoalaExceptie si PolinomNeselectatExceptie
- gui - conține clasa Interfata si PanelFereastra
- model - conține clasa Polinom
- util - conține clasa PolinomPattern
- controller - conține clasa Operatii

Descrierea claselor:

Clasa CasutaGoalaExceptie este o excepție personalizată care este aruncată atunci când o căsuță din interfața utilizator este lăsată goală.

Clasa Interfata este clasă principală care implementează interfața grafică utilizator. Acesta conține elementele grafice pentru afișarea polinoamelor, introducerea de polinoame, selectarea operației și afisarea rezultatului. Aceasta utilizează PanelFereastra pentru a conține toate elementele grafice.

Clasa Operatii conține implementările tuturor operațiilor de bază care pot fi efectuate asupra polinoamelor, cum ar fi adunarea, scaderea, înmulțirea, împărțirea, derivarea și integrarea.

Clasa PanelFereastra este o clasă auxiliară care conține elementele grafice pentru afișarea polinoamelor, introducerea de polinoame și afișarea mesajelor de eroare. Aceasta este utilizată de Interfata pentru a grupa elementele grafice și a le plasa corect.

Clasa Polinom reprezintă un polinom și conține o mapă care stochează fiecare termen al polinomului și coeficientul său. Aceasta oferă metode pentru a adauga, scadea, înmulți, împărți, deriva și integra polinoame.

Clasa PolinomNeselectatExceptie este o excepție personalizată care este aruncată atunci când nu este selectat niciun polinom pentru a fi procesat.

Clasa PolinomPattern este o clasă utilitară care conține un pattern predefinit pentru a extrage coeficienții și gradul unui polinom dintr-un string. Aceasta este utilizată pentru a valida si procesa polinoamele introduse de utilizator.

Interfața utilizator:

Interfața utilizator are la bază două elemente principale: zona de introducere a polinoamelor și zona de afișare a operațiilor. În zona de introducere a polinoamelor, utilizatorul poate introduce polinoamele folosind coeficienții și gradul acestora. Utilizatorul poate selecta un polinom pentru a fi procesat. În zona de afișare a operațiilor, utilizatorul poate selecta operația dorită și poate vizualiza rezultatul operației pentru polinoamele selectate. În cazul în care este detectată o eroare, cum ar fi o casuță goală sau pentru operațiile de derivare și integrare, neselectarea unui polinom anume, se va afișa un mesaj de eroare corespunzător.

Descrierea cazurilor de utilizare:

1. Inserează polinom:

- Utilizatorul introduce coeficienții polinomului prin intermediul interfeței grafice.

- Utilizatorul selectează gradul polinomului.

- Utilizatorul confirmă introducerea polinomului.

- Sistemul validează polinomul.

- Sistemul adaugă polinomul în lista de polinoame.

2. Efectuează adunare:

- Utilizatorul introduce două polinoame.
- Sistemul efectuează adunarea polinoamelor.
- Sistemul afișează rezultatul adunării.

3. Efectuează scădere:

- Utilizatorul introduce două polinoame.
- Sistemul efectuează scăderea polinoamelor.
- Sistemul afișează rezultatul scăderii.

4. Efectuează înmulțire:

- Utilizatorul introduce două polinoame.
- Sistemul efectuează înmulțirea polinoamelor.
- Sistemul afișează rezultatul înmulțirii.

5. Efectuează împărțire:

- Utilizatorul introduce două polinoame.
- Sistemul efectuează împărțirea polinoamelor.
- Sistemul afișează rezultatul împărțirii.

6. Efectuează derivare:

- Utilizatorul introduce și selectează un polinom.
- Sistemul efectuează derivarea polinomului.
- Sistemul afișează rezultatul derivării.

7. Efectuează integrare:

- Utilizatorul introduce și selectează un polinom.
- Sistemul efectuează integrarea polinomului.
- Sistemul afișează rezultatul integrării.

c. Scenarii:

1. Inserează polinom:

- Utilizatorul introduce un polinom printr-o lista de perechi determinate de coeficienții polinomului și gradele acestuia.

- Sistemul salvează polinomul în memoria sa internă.

2. Selectează operația matematică:

- Utilizatorul selectează operația matematică dorită dintr-o lista de butoane.

- Sistemul validează operația matematică selectată.

3. Adunare polinoame:

- Utilizatorul selectează operația de adunare.

- Utilizatorul introduce două polinoame in panele.

- Sistemul validează polinoamele selectate și efectuează operația de adunare.

- Sistemul afișează rezultatul adunării.

4. Scădere polinoame:

- Utilizatorul selectează operația de scădere.

- Utilizatorul introduce două polinoame in panele.

- Sistemul validează polinoamele selectate și efectuează operația de scădere.

- Sistemul afișează rezultatul scăderii.

5. Înmulțire polinoame:

- Utilizatorul selectează operația de înmulțire.

- Utilizatorul introduce două polinoame in panele.

- Sistemul validează polinoamele selectate și efectuează operația de înmulțire.

- Sistemul afișează rezultatul înmulțirii.

6. Împărțire polinoame:

- Utilizatorul selectează operația de împărțire.
- Utilizatorul introduce două polinoame in panele.
- Sistemul validează polinoamele selectate și efectuează operația de împărțire.
- Sistemul afișează rezultatul împărțirii.

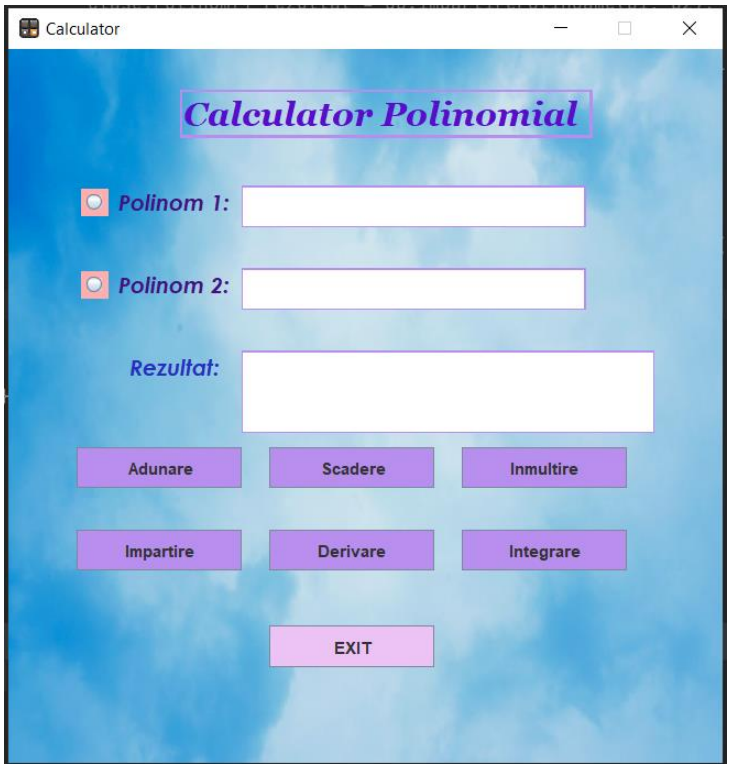
7. Derivare polinom:

- Utilizatorul selectează operația de derivare.
- Utilizatorul introduce cel puțin un polinom in panele.
- Utilizatorul selectează polinom pe care dorește să îl deriveze.
- Sistemul validează polinomul selectat și efectuează operația de derivare.
- Sistemul afișează rezultatul derivatei.

8. Integrare polinom:

- Utilizatorul selectează operația de integrare.
- Utilizatorul introduce cel puțin un polinom in panele.
- Utilizatorul selectează polinom pe care dorește să îl integreze.
- Sistemul validează polinomul selectat și efectuează operația de integrare.
- Sistemul afișează rezultatul integralei.

3. PROIECTARE



Proiectarea este una dintre cele mai importante etape în dezvoltarea oricărui proiect software, inclusiv în dezvoltarea aplicației Java descrisă în tema dată. Aceasta implică stabilirea structurii generale a aplicației, a modului în care componentele sale interacționează între ele și a modului în care datele sunt stocate și procesate. Pentru a îndeplini cerințele temei, se urmează o abordare de proiectare orientată obiect, structuri de tip Map pentru a modela polinoamele și implementarea unei interfețe grafice pentru utilizatori, folosind Java Swing. Pentru o calitate bună a codului se vor defini clase cu o lungime maximă de 300 de linii și metode cu o lungime maximă de 30 de linii. De asemenea se folosesc convențiile de denumire Java.

Odată finalizată proiectarea, se poate trece la implementarea efectivă a aplicației, urmând planul de proiectare stabilit.

Expresia de mai jos este un pattern de regex folosit pentru a valida un polinom cu un singur termen, care poate fi utilizat pentru a extrage coeficienții și puterile variabilei x. Această expresie regulată are următorul format:

([+-]?\d*(\.\d+)?(x(\^[0-9]+)?)?)+

În esență, acest regex verifică dacă un șir de caractere introdus de utilizator este un polinom valid, verificând dacă respectă regulile de bază ale unui polinom: coeficienții sunt numere întregi sau cu zecimale, variabila este "x" și exponentul (dacă este prezent) este un număr întreg pozitiv.

4. IMPLEMENTARE

În implementarea proiectului, am început prin a defini clasele și metodele necesare pentru a realiza operațiile cu polinoame. Am creat clasa Polinom pentru a reprezenta polinoamele. Am decis să folosesc o mapare pentru a reprezenta polinoamele, cu cheia fiind gradul monomului și valoarea fiind coeficientul monomului respectiv. Această abordare simplifică operațiile cu polinoame, deoarece putem adăuga și scădea monoame cu același grad folosind doar cheia asociată gradului respectiv.

Am implementat operațiile de adunare și scădere, urmate de operațiile de înmulțire, împărțire, derivare și integrare. Pentru operația de înmulțire, am utilizat o metodă similară cu cea pe care o folosim manual, adică înmulțim fiecare monom din primul polinom cu fiecare monom din al doilea polinom și adăugăm monoamele rezultate într-un nou polinom. Pentru operația de derivare, am aplicat regula de derivare a polinoamelor pentru fiecare monom din polinom. Pentru operația de integrare, am aplicat regula de integrare a polinoamelor pentru fiecare monom din polinom.

Am folosit expresii regulate și potrivirea șabloanelor pentru a extrage coeficienții polinomului dintr-un șir de caractere introdus de utilizator. Această abordare a făcut posibilă extragerea automată a coeficienților și a gradelor monomilor din polinom, fără a fi nevoie ca utilizatorul să introducă manual fiecare coeficient și grad.

Pentru a testa codul, am folosit JUnit, o platformă de testare pentru Java. Am creat teste unitare pentru fiecare metodă din clasa Operatii pentru a verifica dacă returnează rezultatele corecte pentru diferite scenarii.

În ceea ce privește interfața grafică, am utilizat Java Swing pentru a crea o interfață simplă și intuitivă pentru utilizator. Interfața permite utilizatorului să introducă polinoamele și să selecteze operațiile pe care dorește să le efectueze. Rezultatele operațiilor sunt afișate într-un câmp de text dedicat.

Explicarea implementării claselor:

1.Clasa Polinom:

Această clasă reprezintă un polinom. Polinomul este stocat sub forma unui dicționar, unde cheile reprezintă gradele monomilor, iar valorile sunt coeficienții corespunzători acestora. Clasa Polinom este echipată cu o serie de metode pentru adăugarea și manipularea termenilor polinomului.

Metode:

- Polinom(): Constructorul clasei, care inițializează polinomul cu un dicționar gol.

- getTermeni(): Map<Integer, Double>: Returnează dicționarul ce conține termenii polinomului.

- adaugaMonom(Integer exp, Double coef): Adaugă un monom în polinom, reprezentat prin exponentul și coeficientul corespunzător.

- toString(): String: Transformă polinomul într-un șir de caractere (String), în vederea afișării. Acesta este obținut prin concatenarea termenilor, ordonați după grade și într-o formă ușor de înțeles pentru utilizatorii umani.

- getGrad(): int: Returnează gradul polinomului, adică cel mai mare exponent din polinomul dat.

2. Clasa Operatii:

Această clasă conține metode pentru realizarea operațiilor matematice de adunare, scădere, înmulțire și împărțire între două polinoame. Clasa utilizează obiecte de tip Polinom pentru a realiza aceste operații.

Metode:

- Operatii(): Constructorul clasei, care nu primește niciun parametru.

- adunarePolinoame(Polinom pol1, Polinom pol2): Polinom: Realizează adunarea a două polinoame primite ca parametri și returnează un nou polinom care este suma lor.

- scaderePolinoame(Polinom pol1, Polinom pol2): Polinom: Realizează scăderea a două polinoame primite ca parametri și returnează un nou polinom care este diferența lor.

- inmultirePolinoame(Polinom pol1, Polinom pol2): Polinom: Realizează înmulțirea a două polinoame primite ca parametri și returnează un nou polinom care este produsul lor.

- impartirePolinoame(Polinom pol1, Polinom pol2): Polinom[]: Realizează împărțirea a două polinoame primite ca parametri și returnează un vector de două polinoame: primul este rezultatul împărțirii, iar al doilea este restul împărțirii.

Acestea sunt metodele definite în clasa Operatii pentru efectuarea operațiilor cu polinoame.

3. Clasa PolinomPattern implementează două tipuri de expresii regulate pentru a verifica dacă un șir de caractere poate fi interpretat ca un polinom valid.

Expresia regulată patternVerificare se folosește pentru a verifica dacă un șir de caractere este un polinom valid, iar expresia regulată pattern se folosește pentru a extrage monoamele din șirul de caractere.

Pentru a verifica dacă un șir de caractere este un polinom valid, se folosește expresia regulată patternVerificare:

"([+-]?\\d*(\\.\\d+)?(x(\\^[0-9]+)?)?)+"

Această expresie regulată verifică dacă șirul de caractere începe cu un semn opțional (+/-), urmat de un număr opțional (cu sau fără parte zecimală), urmat de o variabilă opțională (x), urmată de un exponent opțional (cu semnul ^ și un număr întreg pozitiv). Acest șir de caractere poate fi repetat de mai multe ori.

Pentru a extrage monoamele dintr-un șir de caractere, se folosește expresia regulată pattern:

"([+-]?\\d*(\\.\\d+)?(x(\\^[0-9]+)?)?)"

Această expresie regulată extrage un monom care poate începe cu un semn opțional (+/-), urmat de un coeficient opțional (cu sau fără parte zecimală), urmat de o variabilă opțională (x), urmată de un exponent opțional (cu semnul ^ și un număr întreg pozitiv).

Pentru a crea un obiect Polinom dintr-un șir de caractere, se folosește metoda crearePolinom din clasa PolinomPattern. Această metodă primește ca argumente un șir de caractere și un obiect Polinom, și adaugă monoamele extrase din șirul de caractere în obiectul Polinom.

4. Clasa Interfata este o clasă ce definește interfața grafică pentru o aplicație de calculator polinomial. Ea este definită în cadrul pachetului Clase și extinde clasa JFrame din librăria javax.swing.

Interfața grafică conține câteva componente, și anume:

- JLabel-uri: labelCalculator, labelPolinom1, labelPolinom2, labelRezultat, care reprezintă etichetele pentru fiecare câmp de intrare (polinomul 1, polinomul 2 și rezultatul)

- JTextField-uri: polinom1Panel, polinom2Panel, care reprezintă casetele pentru introducerea polinoamelor

- JTextArea: rezultatPanel, care reprezintă caseta pentru afișarea rezultatului

- JRadioButton-uri: p1, p2, care permit utilizatorului să selecteze cu care dintre cele două polinoame vrea să lucreze

- JButton-uri: adunare, scadere, inmultire, impartire, derivare, integrare, iesire, care permit utilizatorului să efectueze diferite operații cu polinoamele.

În plus, clasa conține și câteva variabile private, cum ar fi:

- polinom1, polinom2, care reprezintă cele două polinoame cu care se lucrează

rezultat, care reprezintă rezultatul unei operații efectuate între cele două polinoame

calcul, o instanță a clasei Operatii care se va ocupa de efectuarea operațiilor aritmetice asupra polinoamelor

- pattern, o instanță a clasei PolinomPattern, folosită pentru validarea polinoamelor introduse de utilizator.

În ceea ce privește metodele din clasa Interfata, avem:

- aspectInterfata(), care se ocupă de setarea aspectului componentelor din interfața grafică (culoare, font, poziție etc.)

- resetarePolinoame(), care se ocupă de ștergerea termenilor din cele două polinoame (pentru a putea introduce altele)

- afisareRezultat(), care se ocupă de afișarea rezultatului în caseta corespunzătoare.

Este important de menționat că această clasă doar defineste și afișează interfața grafică, însă operațiile efective asupra polinoamelor se fac în cadrul clasei Operatii.

5. REZULTATE

În urma implementării, am obținut o aplicație Java funcțională care permite utilizatorilor să introducă două polinoame, să selecteze o operație matematică și să afișeze rezultatul operației alese. Am utilizat o abordare orientată pe obiecte și am folosit clase pentru a modela polinoamele și operațiile matematice.

Testele unitare efectuate cu JUnit au confirmat funcționarea corectă a tuturor claselor și metodelor importante, iar documentația bine structurată și detaliată face codul să fie ușor de citit.

Pentru a verifica dacă implementarea operațiilor este corectă, am scris câte un test de verificare pentru fiecare operație din clasa Operatii. Pentru scrierea testelor am folosit Junit.

1. Adunarea

```
@Test
public void testareSuma(){
    Polinom p1=new Polinom();
    p1.adaugaMonom( exp: 2, coef: 3.0);
    p1.adaugaMonom( exp: 0, coef: 1.0);
    p1.adaugaMonom( exp: 1, coef: -1.0);

    Polinom p2=new Polinom();
    p2.adaugaMonom( exp: 0, coef: -2.0);
    p2.adaugaMonom( exp: 3, coef: 3.0);
    p2.adaugaMonom( exp: 1, coef: 1.0);

    Polinom p3=new Polinom();
    p3.adaugaMonom( exp: 3, coef: 3.0);
    p3.adaugaMonom( exp: 2, coef: 3.0);
    p3.adaugaMonom( exp: 1, coef: 0.0);
    p3.adaugaMonom( exp: 0, coef: -1.0);

    Polinom suma = op.adunarePolinoame(p1,p2);

    assertEquals(p3.toString(), suma.toString());
}
```

Această funcție de test verifică dacă metoda de adunare a două polinoame din clasa Operatii returnează rezultatul așteptat.
Pentru a testa metoda:

- se instanțiază două obiecte de tip Polinom și unul de tip Operatii
- apelăm funcția getPolynomialFromString
- stocăm rezultatul adunarii într-un nou polinom
- comparăm rezultatul așteptat cu rezultatul actual, dacă este corect crește numărul de teste executate cu success

2. Scăderea

```
@Test
public void testareDiferenta(){
    Polinom p1=new Polinom();
    p1.adaugaMonom( exp: 2, coef: 3.0);
    p1.adaugaMonom( exp: 1, coef: -1.0);
    p1.adaugaMonom( exp: 0, coef: 1.0);

    Polinom p2=new Polinom();
    p2.adaugaMonom( exp: 3, coef: 3.0);
    p2.adaugaMonom( exp: 1, coef: 1.0);
    p2.adaugaMonom( exp: 0, coef: -2.0);

    Polinom p3=new Polinom();
    p3.adaugaMonom( exp: 3, coef: -3.0);
    p3.adaugaMonom( exp: 2, coef: 3.0);
    p3.adaugaMonom( exp: 1, coef: -2.0);
    p3.adaugaMonom( exp: 0, coef: 3.0);

    Polinom diferenta = op.scaderePolinoame(p1,p2);

    assertEquals(p3.toString(), diferenta.toString());
}
```

Această funcție testează metoda scaderePolinoame() din clasa Operatii.

Pentru a testa metoda:

- se instanțiază două obiecte de tip Polinom și unul de tip Operatii
- se adaugă monoame la fiecare polinom
- se calculează diferența dintre cele două polinoame folosind metoda scaderePolinoame()
- se compară rezultatul așteptat cu rezultatul actual, dacă sunt egale, testul este considerat valid

3. Înmulțirea

```
@Test
public void testareInmultire(){
    Polinom p1=new Polinom();
    p1.adaugaMonom( exp: 2, coef: 3.0);
    p1.adaugaMonom( exp: 1, coef: -1.0);
    p1.adaugaMonom( exp: 0, coef: 1.0);

    Polinom p2=new Polinom();
    p2.adaugaMonom( exp: 3, coef: 3.0);
    p2.adaugaMonom( exp: 1, coef: 1.0);
    p2.adaugaMonom( exp: 0, coef: -2.0);

    Polinom p3=new Polinom();
    p3.adaugaMonom( exp: 5, coef: 9.0);
    p3.adaugaMonom( exp: 4, coef: -3.0);
    p3.adaugaMonom( exp: 3, coef: 6.0);
    p3.adaugaMonom( exp: 2, coef: -7.0);
    p3.adaugaMonom( exp: 1, coef: 3.0);
    p3.adaugaMonom( exp: 0, coef: -2.0);

    Polinom produs = op.inmultirePolinoame(p1,p2);

    assertEquals(p3.toString(), produs.toString());
}
```

Aici avem un test pentru metoda inmultirePolinoame a clasei Operatii. Testul are urmatorii pasi:

➢Instanțiem două obiecte de tip Polinom, p1 și p2, și un obiect de tip Operation, op.

➢Folosind funcția adaugaMonom a claselor Polinom, adaugăm termenii polinoamelor p1 și p2.

➢Apelăm funcția inmultirePolinoame a obiectului op pentru a calcula produsul dintre p1 și p2.

➢Stocăm rezultatul înmultirii într-un nou obiect de tip Polinom, produs.

➢Comparăm rezultatul așteptat, stocat în obiectul p3, cu rezultatul actual, stocat în obiectul produs, folosind funcția assertEquals. Dacă testul este corect, numărul de teste executate cu succes va crește.

4. Împărțirea

```
@Test
public void testareImpartire(){
    Polinom p1=new Polinom();
    p1.adaugaMonom( exp: 3, coef: 1.0);
    p1.adaugaMonom( exp: 2, coef: -2.0);
    p1.adaugaMonom( exp: 1, coef: 6.0);
    p1.adaugaMonom( exp: 0, coef: 4.0);

    Polinom p2=new Polinom();
    p2.adaugaMonom( exp: 2, coef: 1.0);
    p2.adaugaMonom( exp: 1, coef: 2.0);
    p2.adaugaMonom( exp: 0, coef: 3.0);

    Polinom p3=new Polinom();
    p3.adaugaMonom( exp: 1, coef: 1.0);
    p3.adaugaMonom( exp: 0, coef: -4.0);

    Polinom p4=new Polinom();
    p4.adaugaMonom( exp: 1, coef: 11.0);
    p4.adaugaMonom( exp: 0, coef: 16.0);

    Polinom[] rezultat = op.impartirePolinoame(p1, p2);
    Polinom cat  = rezultat[0];
    Polinom rest = rezultat[1];

    assertEquals(p3.toString(), cat.toString());
    assertEquals(p4.toString(), rest.toString());
}
```

Testul de mai sus verifică funcționalitatea metodei impartirePolinoame din clasa Operatii.

➢Pasul 1: Se instantiază două obiecte de tip Polinom p1 și p2 și se adaugă monoame la acestea.

➢Pasul 2: Se apelează metoda impartirePolinoame din obiectul op de tip Operatii cu parametrii p1 și p2. Rezultatul este stocat într-un array de două elemente, cat și rest.

➢Pasul 3: Se compară rezultatele așteptate, obținute prin instanțierea și adăugarea monoamelor la p3 și p4, cu rezultatele actuale obținute în pasul 2 prin accesarea membrilor cât și rest din array-ul de rezultate.

➢Dacă testul trece cu succes, se crește numărul de teste executate cu succes.

5. Derivarea

```
@Test
public void testareDerivare(){
    Polinom p1=new Polinom();
    p1.adaugaMonom( exp: 2, coef: 3.0);
    p1.adaugaMonom( exp: 0, coef: 1.0);
    p1.adaugaMonom( exp: 1, coef: -1.0);

    Polinom p2=new Polinom();
    p2.adaugaMonom( exp: 1, coef: 6.0);
    p2.adaugaMonom( exp: 0, coef: -1.0);

    Polinom der = op.derivarePolinom(p1);

    assertEquals(der.toString(), p2.toString());
}
```

În acest test, se face verificarea funcției de derivare a unui polinom.

➢ se instanțiază un obiect de tip Polinom și unul de tip Operatii

➢ apelam funcția getPolynomialFromString

➢ stocăm rezultatul derivarii într-un nou polinom

➢ comparăm rezultatul așteptat cu rezultatul actual, dacă este corect crește numarul de teste executate cu success.

6. Integrarea

```
@Test
public void testareIntegrare(){
    Polinom p1=new Polinom();
    p1.adaugaMonom( exp: 2, coef: 3.0);
    p1.adaugaMonom( exp: 0, coef: 1.0);
    p1.adaugaMonom( exp: 1, coef: -1.0);

    Polinom p2=new Polinom();
    p2.adaugaMonom( exp: 3, coef: 1.0);
    p2.adaugaMonom( exp: 2, coef: -0.5);
    p2.adaugaMonom( exp: 1, coef: 1.0);

    Polinom intr = op.integrarePolinom(p1);

    assertEquals(intr.toString(), p2.toString());
}
```

În acest test, se face verificarea funcției de integrare a unui polinom.

➢se instantiaza un obiect de tip Polinom si unul de tip Operation

➢ apelăm funcția getPolynomialFromString

➢ stocăm rezultatul integrarii într-un nou polinom

➢ comparăm rezultatul așteptat cu rezultatul actual, dacă este corect crește numărul de

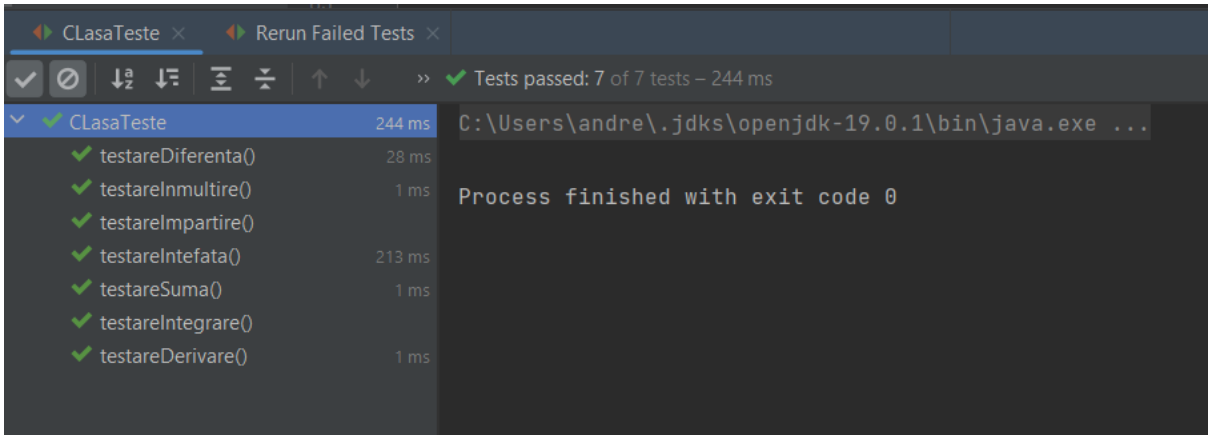
teste executate cu success

7. Interfața

```
@Test
public void testareIntefata(){

    Interfata calculator1 = new Interfata( title: "Calculator");
}
```

După ce rulăm toate testele, putem observa care dintre ele au avut succes și care nu.



6. CONCLUZII

În final, am reușit să implementăm o aplicație Java care permite utilizatorilor să introducă două polinoame, să selecteze o operație matematică și să afișeze rezultatul. Am utilizat programarea orientată pe obiecte și am definit clase adecvate pentru polinoame și monoame, am modelat polinoamele cu ajutorul unei Map-uri și am implementat operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare.

În general, am atins obiectivele proiectului și am reușit să livrăm o aplicație Java funcțională și bine documentată care îndeplinește cerințele specificate. Cu toate acestea, există încă loc pentru îmbunătățiri, cum ar fi adăugarea mai multor teste și creșterea acoperirii testelor pentru a asigura o mai bună calitate a codului și o performanță mai bună a aplicației noastre.

7. BIBLIOGRAFIE

➔ How to extract the coefficients and exponent of a polynomial as string: https://stackoverflow.com/questions/53143991/how-to-extract-the-coefficients-and-exponent-of-a-polynomial-as-string

➔ Regex generator https://regex-generator.olafneumann.org/?sampleText=5X%5E3%2B3X%5E2-3X-2&flags=i

➔ Tutorial de la Oracle:
https://docs.oracle.com/javase/tutorial/uiswing/index.html
https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
https://gluonhq.com/products/scene-builder/
https://docs.oracle.com/javase/8/docs/api/java/util/regex/package-summary.html

➔ JUnit de la JUnit.org
https://junit.org/junit5/docs/current/user-guide/

➔ Java Naming Conventions de la Oracle:

https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

➔ Java Collections Framework de la Oracle:
https://docs.oracle.com/javase/8/docs/technotes/guides/collections/

➔ Java API de la Oracle:
https://docs.oracle.com/en/java/javase/16/docs/api/index.html

➔ Alte resurse:
https://dsrl.eu/courses/pt/