DOCUMENTAȚIE

TEHNICI DE PROGRAMARE

TEMA 1 - Calculator polinomial

NUME STUDENT: ROTARIU LAURA-ALEXANDRA
GRUPA: 30224-2

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei este de a proiecta și implementa un calculator polinomial cu o interfață grafică dedicată în Java care să ofere utilizatorului o modalitate ușoară de introduce polinoame, de a efectua operații cu polinoame și de a vizualiza rezultatul.

Obiectivele secundare ale temei (pașii care trebuie urmați pentru îndeplinirea obiectivului principal):

- Identificarea funcționalităților de bază care vor fi oferite utilizatorilor prin intermediul interfeței grafice simple și intuitive: adunarea, scăderea, înmulțirea, împărțirea, derivarea și integrarea polinoamelor. În această etapă este important să se ia in considerare utilizarea de variabile și constante, coeficienții și exponenții și să se stabilească forma sub care va fi introdus polinomul de către utilizator. Acest pas va fi descris in secțiunea Analiza problemei, modelare, scenarii, cazuri de utilizare.
- Scrierea codului Java necesar pentru a introduce polinoame și pentru a efectua operațiile dorite. Este importantă organizarea codului în clase și pachete în funcție de utilitatea acestuia. Acești pași vor fi descriși în secțiunile Proiectare și Implementare.
- Proiectarea interfeței grafice pentru utilizatori: utilizarea unor component standard, precum panouri, butoane, casete de text, etichete pentru a permite utilizatorilor să introducă polinoame, să selecteze operații și să vadă rezultatul efectuării lor. Această interfață trebuie sa fie simplă și intuitivă astfel încât utilizatorul să o folosească ușor. Acest pas va fi descris in secțiunea Implementare.
- Testarea operațiilor pentru a vedea dacă aplicația funcționează corect. Acest lucru se realizează prin crearea de scenarii de testare și verificarea rezultatelor obținute cu cele așteptate. Trebuie testate toate cazurile posibile pentru a evita erori în momentul utilizării aplicației. Acest pas va fi descris în secțiunea Rezultate.

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

- **Cerințe funcționale:**

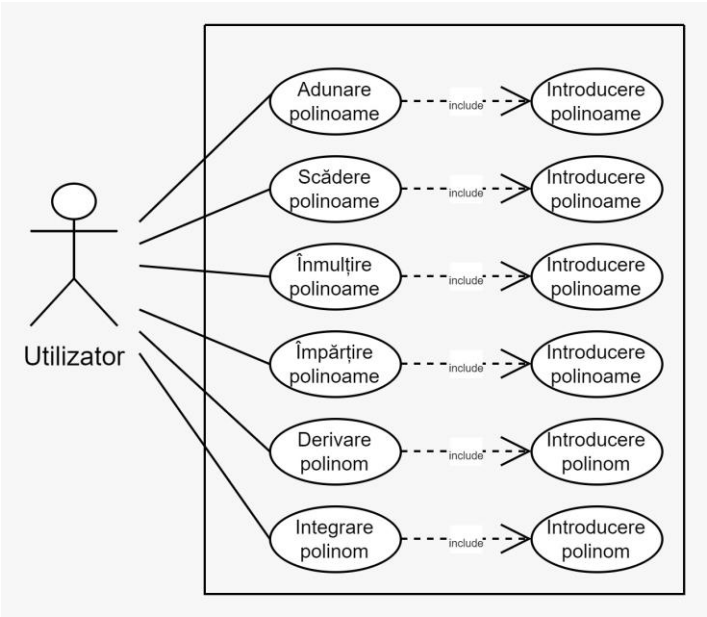
- Calculatorul polinomial trebuie să permită utilizatorilor să introducă polinoame.
- Calculatorul polinomial trebuie să permită utilizatorilor să selecteze operația dorită.
- Calculatorul polinomial trebuie să poată efectua operațiile de adunare, scădere, înmulțire și împărțire a două polinoame și operații de derivare și integrare pentru un polinom.
- Calculatorul polinomial trebuie să permită utilizatorilor să vadă rezultatul obținut în urma efectuării operației selectate.

- **Cerințe non-funcționale:**

- Calculatorul polinomial trebuie să fie intuitiv și ușor de folosit de către utilizator.
- Calculatorul polinomial trebuie să efectueze operațiile și să afișeze rezultatele într-un timp scurt.

- Calculatorul polinomial trebuie să funcționeze indiferent de sistemul de operare folosit.

- **Cazuri de utilizare:**



- **Descrieri use-case-uri:**

Use case: adunare polinoame
Actor principal: utilizator
Scenariul principal cu succes:

1. Utilizatorul introduce cele două polinoame în interfața grafică.
2. Utilizatorul selectează operația de adunare.
3. Calculatorul efectuează adunarea celor două polinoame și afișează rezultatul.

Scenariul alternativ: Polinoame introduse incorect

- Utilizatorul introduce polinoamele incorect, de exemplu dacă introduce o variabilă în afară de "x", exponenți negativi sau spații, se va afișa un mesaj, întrucât polinoamele trebuie să fie de forma "a*x^n+b*x^n-1+..+w*x+z". Polinomul trebuie introdus în forma simplificată, fără a avea termeni cu același exponent de mai multe ori. Exponentul și coeficientul egali cu 1 pot fi omiși. De asemenea, se poate omite "*" dintre coeficient și variabila x.
- Se afișează un mesaj și scenariul se va întoarce la primul pas.

Use case: scădere polinoame
Actor principal: utilizator
Scenariul principal cu succes:

1. Utilizatorul introduce cele două polinoame în interfața grafică.
2. Utilizatorul selectează operația de scădere.
3. Calculatorul efectuează scăderea celor două polinoame și afișează rezultatul.

Scenariul alternativ: Polinoame introduse incorect

Use case: înmulțire polinoame
Actor principal: utilizator
Scenariul principal cu succes:

1. Utilizatorul introduce cele două polinoame în interfața grafică.
2. Utilizatorul selectează operația de înmulțire.
3. Calculatorul efectuează înmulțirea celor două polinoame și afișează rezultatul.

Scenariul alternativ: Polinoame introduse incorect

Use case: împărțire polinoame
Actor principal: utilizator
Scenariul principal cu succes:

1. Utilizatorul introduce cele două polinoame în interfața grafică.
2. Utilizatorul selectează operația de împărțire.
3. Calculatorul efectuează împărțirea celor două polinoame și afișează rezultatul.

Scenariul alternativ: Polinoame introduse incorect sau al doilea polinom egal cu 0
-Dacă utilizatorul nu introduce al doilea polinom sau introduce un polinom egal cu 0, se va afișa un mesaj și scenariul se întoarce la primul pas.

Use case: derivare polinom
Actor principal: utilizator
Scenariul principal cu succes:

1. Utilizatorul introduce un polinom în interfața grafică.
2. Utilizatorul selectează operația de derivare.
3. Calculatorul efectuează derivarea polinomului și afișează rezultatul.

Scenariul alternativ: Polinom introdus incorect

Use case: integrare polinom
Actor principal: utilizator
Scenariul principal cu succes:

1. Utilizatorul introduce polinomul în interfața grafică.
2. Utilizatorul selectează operația de integrare.
3. Calculatorul efectuează integrarea polinomului și afișează rezultatul.

Scenariul alternativ: Polinom introdus incorect.

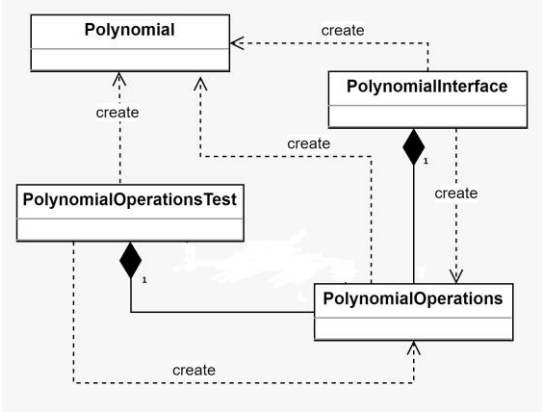
3. Proiectare

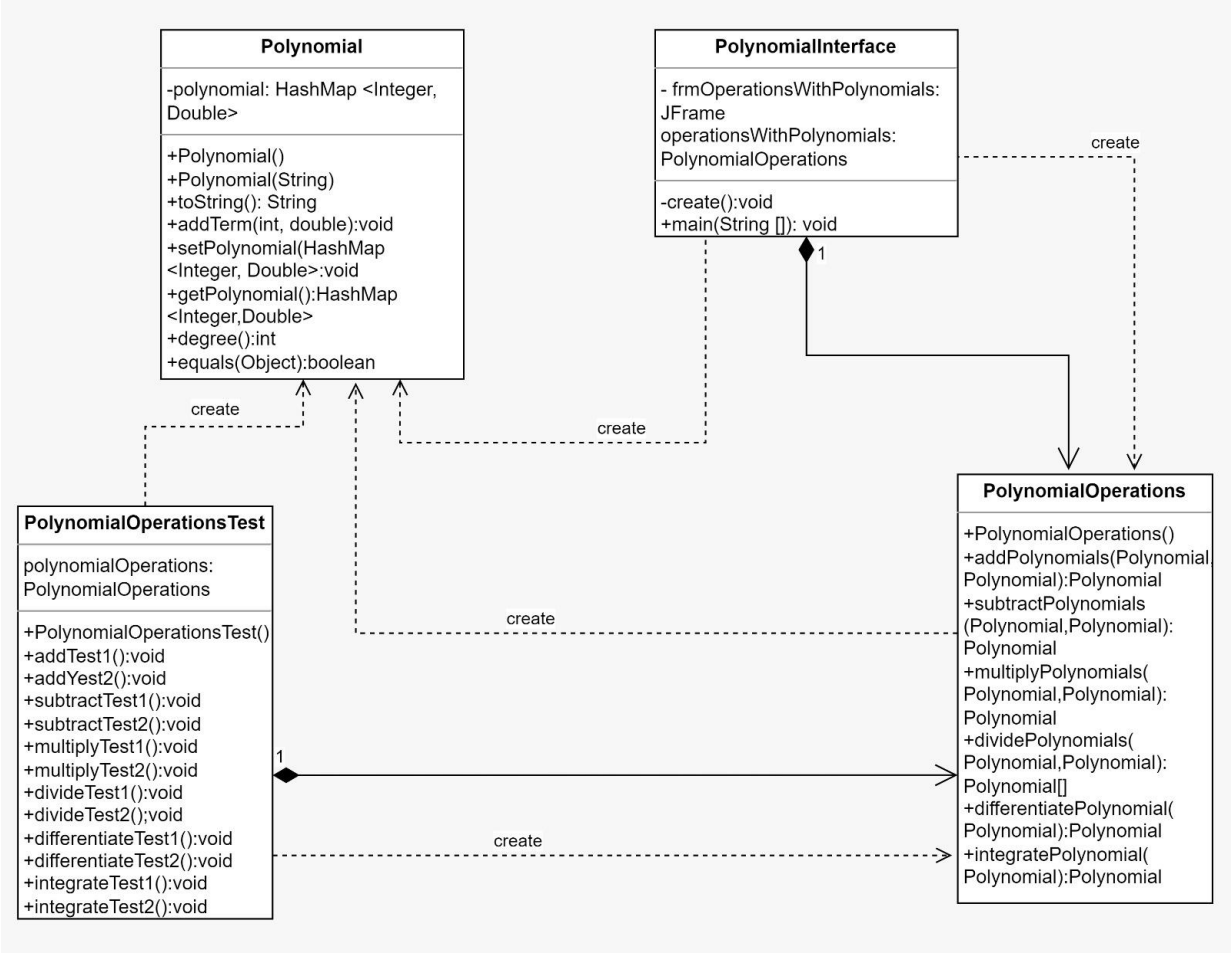
- Proiectarea OOP a aplicației:

Pentru a rezolva specificațiile problemei, am ales sa folosesc patru clase: Polynomial, PolynomialOperations, PolynomialInterface si PolynomialOperationsTest, fiecare inclusă într-un pachet diferit. Clasa Polynomial este inclusă în pachetul org.example.DataModels și aici am implementat un model pentru un polinom de variabilă “x”, utilizând structura de date HashMap, care permite accesul eficient la coeficient pentru un anumit exponent. În această clasă are loc transformarea String-ului introdus de utilizator în interfața grafică într-un polinom și verificarea corectitudinii printr-o expresie regulată, precum și implementarea afișării polinomului într-un

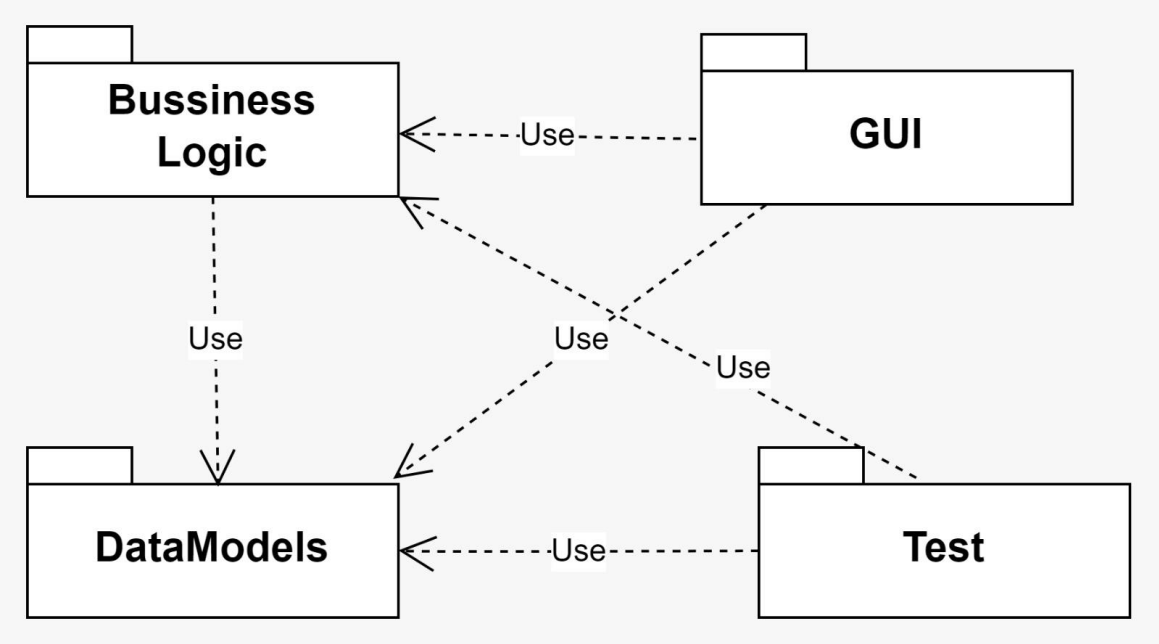
mod ușor de citit și înțeles pentru utilizator. Clasa PolynomialOperations este inclusă în pachetul org.example.BusinessLogic și aici sunt implementate operațiile pe care le poate efectua calculatorul. Clasa PolynomialInterface este inclusă în pachetul org.example.GUI și aici se implementează interfața grafică pe care o va folosi utilizatorul, de aici rulându-se și aplicația. Clasa PolynomialOperationsTest este inclusă în pachetul org.example.Test și aici se face testarea cu JUnit.

- **Diagrama UML de clase**:





- **Diagrama de pachete:**



- **Structurile de date folosite:**

  Structurile de date pe care le-am utilizat în implementarea aplicației sunt:
  - `HashMap <Integer, Double>`: pentru a stoca termenii polinomului, cheia este exponentul, iar valoarea este coeficientul;
  - `ArrayList<Integer>`: pentru a sorta exponenții termenilor din polinom în ordine descrescătoare;
  - `String []`: pentru a despărți polinomul în termeni individuali în funcție de + si -;
  - `StringBuilder`: pentru a construi polinomul ce va fi afișat în interfața grafică;
  - `Polynomial:` clasa creată pentru a stoca polinoamele.

- **Algoritmii folosiți:**

- Algoritmul de adunare a două polinoame:
  $P(X) + Q(X) = (a_n+b_n) * X^n + (a_{n-1}+b_{n-1}) * X^{n-1} + \cdots + (a_1+b_1) * X + (a_0 + b_0)$

- Algoritmul de scădere a două polinoame:
  $P(X)+ Q(X) = (an\text{-}bn) * Xn + (an-1\text{-}bn-1) * X\,n-1 + \cdots + (a1\text{-}b1) * X + (a0 - b0)$

- Algoritmul de înmulțire a două polinoame:
  Pentru a înmulți două polinoame, trebuie să se înmulțească fiecare monom dintr-un polinom cu fiecare monom din celălalt polinom, să se adune rezultatele și să se simplifice dacă este necesar.

- Algoritmul pentru împărțirea a două polinoame:

```
function n / d is
    require d ≠ 0
    q ← 0
    r ← n             // At each step n = d × q + r

    while r ≠ 0 and degree(r) ≥ degree(d) do
        t ← lead(r) / lead(d)       // Divide the leading terms
        q ← q + t
        r ← r − t × d

    return (q, r)
```

- Algoritmul de derivare a unui polinom:
  $d/dx(a_n * X^n + a_{n-1} * X^{n-1} + \cdots + a_1 * X + a_0) = n * a_n * X^{n-1} + (n - 1) * a_{n-1} * X^{n-2} + \cdots + a_1$

- Algoritmul de integrare a unui polinom:
  $\int a_n * X^n + a_{n-1} * X^{n-1} + \cdots + a_1 * X + a_0 = \int a_n * X^n \ dx + \int a_{n-1} * X^{n-1} \ dx + \ldots + \int a_1 * X \ dx + \int a_0 \ dx$, unde: $\int a_n * X^n \ dx = a * (X^{n+1}/n+1) + C$

4. Implementare

- *Clasa Polynomial* - definește un polinom de variabilă "x". Aceasta clasă are ca și atribut un HashMap <Integer, Double> denumit polynomial, în care cheia reprezintă exponentul de tip întreg și valoarea reprezintă coeficientul de tip flotant. Atributul este privat și poate fi accesat prin metode specifice. Clasa are doi constructori, unul fără parametri, care inițializează polinomul cu un HashMap gol și un constructor care primește ca parametru un String și îl transformă într-un polinom. Acest constructor folosește expresii regulate pentru a verifica dacă șirul de caractere corespunde unui polinom valid, daca nu, va afișa o fereastră de dialog cu un mesaj și pune 0 în polinom.

Expresia regulată folosită este următoarea:
"^([-+])?((\\d+(\\d+(\\.\\d+)?)?*?)?(x(\\^(\\d+))?)?$" și verifică dacă termenul începe cu un semn opțional (+ sau -), apoi verifică dacă are coeficientul opțional, care e un număr, urmat de un operator opțional de înmulțire (*), apoi verifică dacă există variabila opțională x, urmată de operatul de ridicare la putere(^) și un exponent opțional al lui x. Dacă șirul este valid, acesta este împărțit în termeni după + sau – și apoi este procesat astfel încât fiecare termen este verificat dacă corespunde formatului corect folosind Pattern și Matcher, iar apoi se extrag coeficienții și exponenții și se adaugă în polinom folosindu-se metoda addTerm(). Dacă apare vreo eroare la procesarea String-ului, se va afișa o fereastră de dialog.

Metodele clasei Polynomial sunt:
- void addTerm(int exponent, double coefficient): adaugă coeficientul și exponentul în polinom doar dacă coeficientul nu e 0, iar dacă e 0, elimină termenul respectiv;
- void HashMap<Integer, Double> getPolynomial: accesează atributul de tip privat;
- void setPolynomial(HashMap <Integer, Double> polynomial): setează valoarea atributului privat;

- int degree(): returnează gradul polinomului, adică cel mai mare exponent din HashMap, sau -1 daca polinomul este gol;
- String toString(): suprascrie metoda toString din clasa Object astfel încât să se afișeze polinomul rezultat într-un mod ușor de înțeles pentru utilizator. Polinomul este format dintr-o sumă de termeni afișați în ordinea descrescătoare a exponentului. Pentru fiecare termen se afișează semnul, coeficientul(dacă e diferit de 1), x(dacă termenul conține x la o putere mai mare decât 0) și exponentul(dacă e mai mare decât 1), iar dacă polinomul este gol, se afișează 0.
- boolean equals(Object other): suprascrie metoda equals din clasa Object pentru a compara două obiecte de tip Polynomial, prin parcurgerea tuturor perechilor de exponenți și coeficienți din cele două polinoame și compararea valorilor coeficienților. Dacă toți sunt egali, metoda returnează true, altfel false.

- *Clasa PolynomialOperations* - nu conține atribute. Conține 6 metode pentru a efectua operații cu polinoame (adunare, scădere, înmulțire, împărțire, derivare și integrare). Primele patru metode primesc ca parametri două obiecte de tipul Polynomial și returnează rezultatul sub forma unui obiect de tip Polynomial (la împărțire sub forma unui vector de obiecte de tip Polynomial), iar ultimele două primesc doar un parametru Polynomial și returnează rezultatul. Metodele folosesc forEach în loc de for.

Metodele sunt:

- Polynomial addPolynomials(Polynomial firstPolynomial, secondPolynomial) care efectuează adunarea a două polinoame primite ca și parametri. Metoda parcurge HashMap-urile fiecărui polinom și adaugă termenii în noul polinom rezultat. Dacă termenii din cele două polinoame au același exponent, se adună coeficienții corespunzători, altfel se adaugă doar termenii dintr-unul în noul polinom. Se returnează polinomul rezultat.
- Polynomial subtractPolynomials(Polynomial firstPolynomial, secondPolynomial) care efectuează scăderea a două polinoame primite ca și parametri și returnează rezultatul. Metoda parcurge HashMap-urile fiecărui polinom și dacă termenii din cele două polinoame au același exponent, se scad coeficienții corespunzători, altfel se adaugă doar dintr-unul în noul polinom.
- Polynomial multiplyPolynomials(Polynomial firstPolynomial, secondPolynomial) care efectuează înmulțirea a două polinoame și returnează rezultatul. Se parcurg HashMap-urile celor două polinoame și se înmulțește fiecare termen din primul polinom cu fiecare termen din al doilea polinom(se adună exponenții și se înmulțesc coeficienții). Dacă în noul polinom există un termen cu aceeași putere, se adună coeficienții corespunzători, altfel se adaugă produsul la noul polinom.
- Polynomial [] dividePolynomials(Polynomial firstPolynomial, secondPolynomial) care efectuează împărțirea a două polinoame și returnează un vector de două polinoame care conține pe prima poziție câtul împărțirii și pe a doua poziție restul împărțirii. Primul polinom este deîmpărțitul, iar al doilea este împărțitorul. Pentru această operație am folosit algoritmul specificat la punctul anterior. Dacă se încearcă împărțirea cu 0, se va afișa un mesaj utilizând clasa JOptionPane.
- Polynomial differentiatePolynomial(Polynomial polynomial) efectuează derivarea unui polinom și returnează rezultatul. Metoda parcurge HashMap-ul polinomului și înmulțește coeficientul cu exponentul corespunzător dacă coeficientul e diferit de 1, dacă

este 1, coeficientul primește valoarea exponentului și dacă exponentul este mai mare decât 1, se decrementează cu 1, altfel primește 0.

- Polynomial integratePolynomial(Polynomial polynomial) efectuează integrarea unui polinom și returnează rezultatul. Metoda parcurge HashMap-ul polinomului și pentru fiecare termen, dacă polinomul conține valori pe acea poziție, incrementează cu 1 valoarea exponentului și coeficientul va fi rezultatul împărțirii lui la noul exponent.

- *Clasa PolynomialOperationsTest* - în această clasă sunt implementate teste JUnit pentru clasa PolynomialOperations, astfel fiind testată corectitudinea operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare. Pentru fiecare operație se fac două teste, unul care trece și unul care eșuează și fiecare test verifică dacă rezultatul operațiilor e egal cu rezultatul așteptat folosind metoda Assert.assertEqual din JUnit care verifică dacă cele două polinoame date ca parametri sunt egale. Clasa are ca atribut un obiect de tipul PolynomialOperations pentru a accesa operațiile din clasa respectivă în cadrul testelor și la fiecare test se instanțiază obiecte de tipul Polynomial și se apelează metoda care trebuie testată cu aceste polinoame.

- *Clasa PolynomialInterface* - implementează interfața grafică pentru un calculator polinomial. Această clasă are ca și câmpuri un obiect de tipul PolynomialOperations pentru a apela metodele din această clasă la apăsarea unor butoane adăugate pentru acest scop și o fereastră JFrame. Clasa are metodele:

- create() unde sunt setate elementele interfeței grafice (panou, etichete, casete, text, butoane, separatori) și sunt adăugați ascultători butoanelor pentru a fi efectuate operații la apăsarea lor;
- metoda statică main care creează o instanță a clasei PolynomialInterface, setează fereastra JFrame să nu poată fi redimensionată și ca fiind vizibilă.

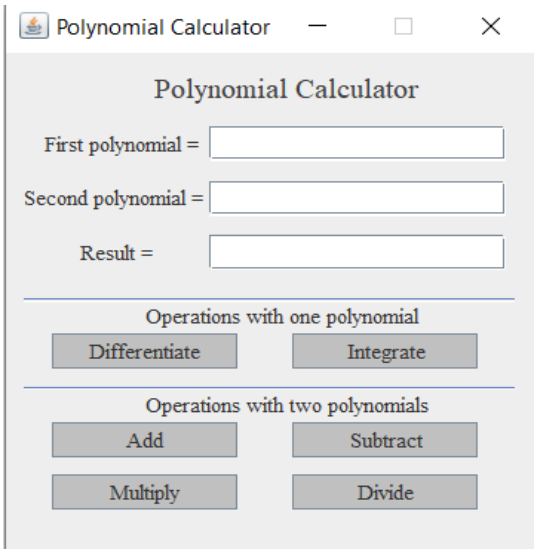
  În constructorul clasei se apelează metoda create().

  Pornirea aplicației are loc în această clasă.

Implementarea interfeței grafice:

Interfața grafică a calculatorului polinomial este simplă și ușor de utilizat. Fereastra are ca titlu Polynomial Calculator. Există două casete text pentru a introduce datele, mai exact două polinoame sub forma de String și o casetă text pentru afișarea rezultatului. Sunt 6 butoane, câte unul pentru fiecare operație, la apăsarea lor efectuându-se operația dorită. Am folosit etichete pentru a preciza utilitatea fiecărei casete text și am denumit butoanele specific. Am folosit separatori pentru a delimita operațiile cu un singur polinom de cele cu două polinoame.

Utilizatorul trebuie să introducă două polinoame pentru a efectua operațiile de adunare, scădere, înmulțire, împărțire și un polinom în prima casetă text pentru derivare și integrare. La apăsarea butonului corespunzător operației, rezultatul operației va fi afișat în caseta specifică. Polinomul introdus trebuie să fie de forma ax^n+bx^n+…wx+z sau a*x^n+b*x^n+…w*x+z. Coeficienții și exponenții egali cu 1 pot fi omiși. Dacă polinomul nu este în formă validă, se va afișa un mesaj cu JOptionPane și utilizatorul va putea introduce un alt polinom.



5. Rezultate

Scenariile pentru testare au fost prezentate în secțiunea a doua a documentației. În clasa PolynomialOperationsTest am efectuat 12 teste, dintre care 6 trebuie să treacă și 6 să eșueze, astfel fiind câte două teste pentru fiecare operație matematică. Testele au fost pentru următoarele exemple:

1. *Adunarea a doua polinoame:*
   Primul polinom: 4*x^5-3*x^4+x^2-8*x+1
   Al doilea polinom: 3*x^4-x^3+x^2+2*x-1
   Rezultatul așteptat: 4*x^5-x^3+2*x^2-6*x
   Rezultatul operației: 4*x^5-x^3+2*x^2-6*x
   Test: trecut

2. *Adunarea a doua polinoame:*
   Primul polinom: 4*x^5-3*x^4+x^2-8*x+1
   Al doilea polinom: 3*x^4-x^3+x^2+2*x-1
   Rezultatul așteptat: 4*x^5
   Rezultatul operației: 4*x^5-x^3+2*x^2-6*x
   Test: eșuat

3. *Scăderea a doua polinoame:*
   Primul polinom: 4*x^5-3*x^4+x^2-8*x+1
   Al doilea polinom: 3*x^4-x^3+x^2+2*x-1
   Rezultatul așteptat: 4*x^5-6*x^4+x^3-10*x+2
   Rezultatul operației: 4*x^5-6*x^4+x^3-10*x+2
   Test: trecut
4. *Scăderea a doua polinoame:*
   Primul polinom: 4*x^5-3*x^4+x^2-8*x+1
   Al doilea polinom: 3*x^4-x^3+x^2+2*x-1

Rezultatul așteptat: 4*x^5-6*x^4
Rezultatul operației: 4*x^5-6*x^4+x^3-10*x+2
Test: eșuat

5. *Înmulțirea a doua polinoame:*
   Primul polinom: 3*x^2-x+1
   Al doilea polinom; x-2
   Rezultatul așteptat: 3*x^3-7*x^2+3*x-2
   Rezultatul operației: 3*x^3-7*x^2+3*x-2
   Test: trecut

6. *Înmulțirea a doua polinoame:*
   Primul polinom: 3*x^2-x+1
   Al doilea polinom; x-2
   Rezultatul așteptat: 3*x^3-x^2+x-2
   Rezultatul operației: 3*x^3-7*x^2+3*x-2
   Test: eșuat

7. *Împărțirea a doua polinoame:*
   Primul polinom: x^3-2*x^2+6*x-5
   Al doilea polinom; x^2-1
   Rezultatul așteptat: Cat: x-2, rest: 7*x-7
   Rezultatul operației: Cat: x-2, rest: 7*x-7
   Test: trecut

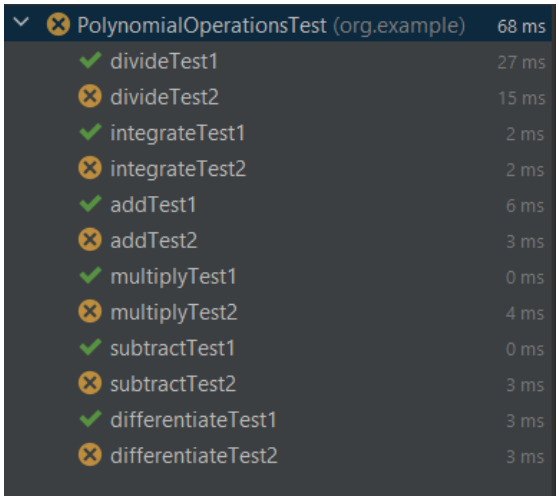
8. *Împărțirea a doua polinoame:*
   Primul polinom: x^3-2*x^2+6*x-5
   Al doilea polinom; x^2-1
   Rezultatul așteptat: Cat: x-1, rest: x-7
   Rezultatul operației: Cat: x-2, rest: 7*x-7
   Test: eșuat

9. *Derivarea unui polinom*
   Polinomul: x^3-2*x^2+6*x-5
   Rezultatul așteptat: 3*x^2-4*x+6
   Rezultatul operației: 3*x^2-4*x+6
   Test: trecut

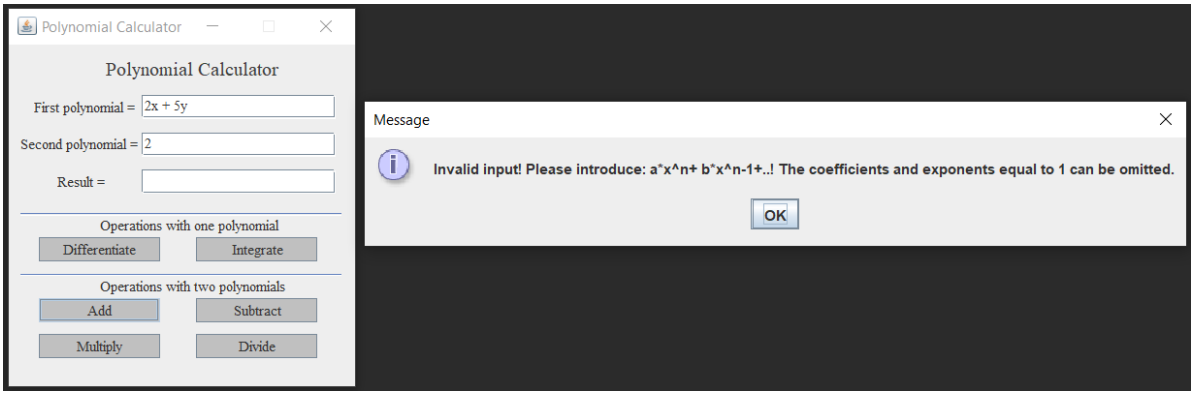
10. *Derivarea unui polinom*
    Polinomul: x^3-2*x^2+6*x-5
    Rezultatul așteptat: 3*x^2
    Rezultatul operației: 3*x^2-4*x+6
    Test: eșuat
11. *Integrarea unui polinom*
    Polinomul: x^3+4*x^2+5
    Rezultatul așteptat: 0.25*x^4+1.33*x^3+5*x

Rezultatul operației: 0.25*x^4+1.33*x^3+5*x
Test: trecut

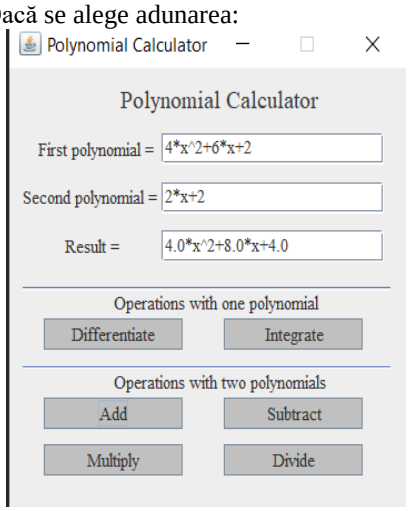
12. *Integrarea unui polinom*
Polinomul: x^3+4*x^2+5
Rezultatul așteptat: 0.25*x^4+1.33*x^3
Rezultatul operației: 0.25*x^4+1.33*x^3+5*x
Test: eșuat



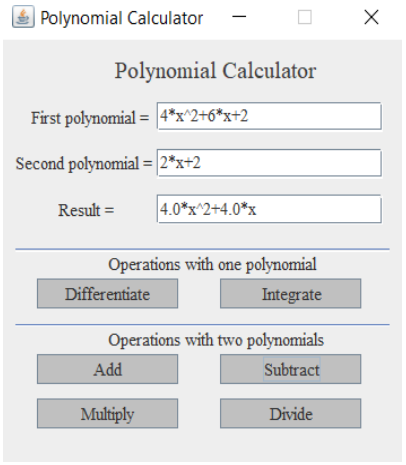
Am testat corectitudinea operațiilor și în interfața grafică pe diverse exemple cu polinoame valide, precum și cu polinoame introduse incorect. În cazul în care se introduce un polinom invalid, se va afișa:



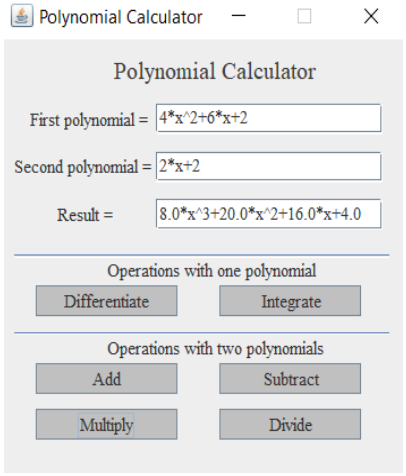
Dacă se alege adunarea:



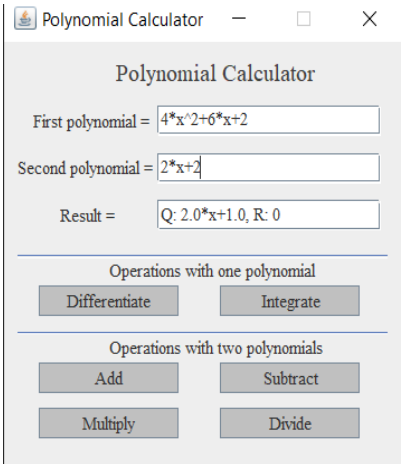
Dacă se alege scăderea:



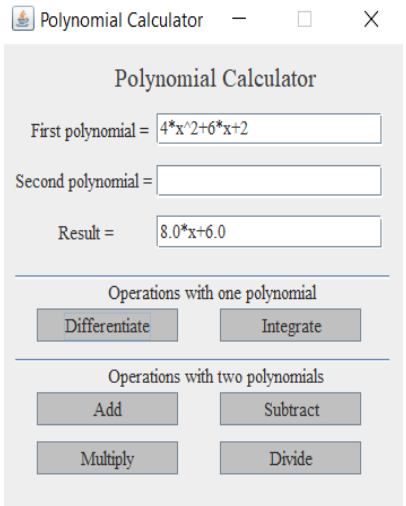
Dacă se alege înmulțirea:



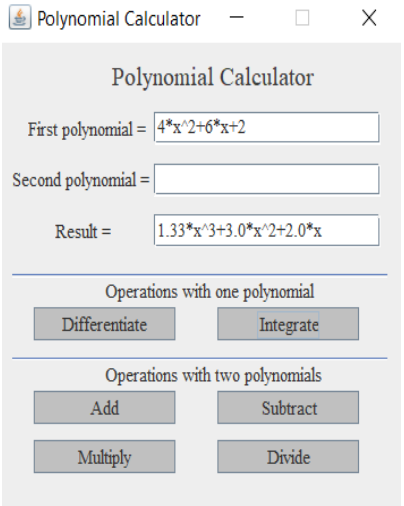
Dacă se alege împărțirea:



Dacă se alege derivarea:



Dacă se alege integrarea:



6. Concluzii

În concluzie, implementarea unui calculator polinomial complex este un lucru destul de greu, atât din punct de vedere al operațiilor, cât și din punct de vedere al interfeței grafice. Unele operații, precum adunarea, scăderea, împărțirea, derivarea și integrarea sunt ușor de implementat, dar împărțirea polinoamelor necesită mai mult lucru.
Totodată, transformarea expresiei introduse de utilizator întru-un polinom necesită multă atenție pentru a trata toate cazurile.

Această temă a ajutat la recapitularea conceptelor legate de Programarea orientată pe obiect, dar și la acumularea unor noi cunoștințe, întrucât am învățat din această temă cum se folosesc expresiile regulate și clasele Pattern și Matcher din pachetul java.util.regex pentru a testa validitatea datelor introduse, precum și cum se utilizează forEach în loc de for. De asemenea, am învățat că este foarte important să fie urmați pașii următori pentru a ajunge la finalizarea sarcinii: analizarea problemei și identificarea funcționalităților calculatorului polinomial, proiectarea calculatorului polinomial, apoi implementarea acestuia și testarea pentru a vedea dacă funcționează fără erori.

Posibile dezvoltări ulterioare ar fi adăugarea mai multor operații precum evaluarea integralei definite, evaluarea polinomului întru-un anumit punct și multe altele. De asemenea, interfața grafică poate fi îmbunătățită, de exemplu fereastra poate fi mai mare astfel încât dacă rezultatul unei operații este foarte mare, să se vadă total.

7. Bibliografie

1. *Fundamental Programming Techniques – Suport Presentations and Lectures* - https://dsrl.eu/courses/pt/
2. *Java Regular Expression* - https://www.w3schools.com/java/java_regex.asp
3. *Java forEach* - https://zetcode.com/java/foreach/
4. *Java HashMap* - https://www.w3schools.com/java/java_hashmap.asp
5. *JUnit* - https://www.tutorialspoint.com/junit/junit_test_framework.htm
6. https://stackoverflow.com/