DOCUMENTATIE

TEMA *1*

NUME STUDENT: Magalau Robert Rayan
GRUPA: 30222

CUPRINS

1. Obiectivul temei

Obiectivul principal:

- Dezvoltarea unei aplicații de calculator pentru operații cu polinoame.

Obiective secundare:

- Implementarea operațiilor de adunare, scădere, înmulțire și împărțire a polinoamelor.
- Realizarea operațiilor de integrare și diferențiere a polinoamelor.
- Interfața grafică a aplicației trebuie să fie intuitivă și ușor de utilizat.
- Utilizarea limbajului Java și a platformei JavaFX pentru dezvoltarea aplicației.

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

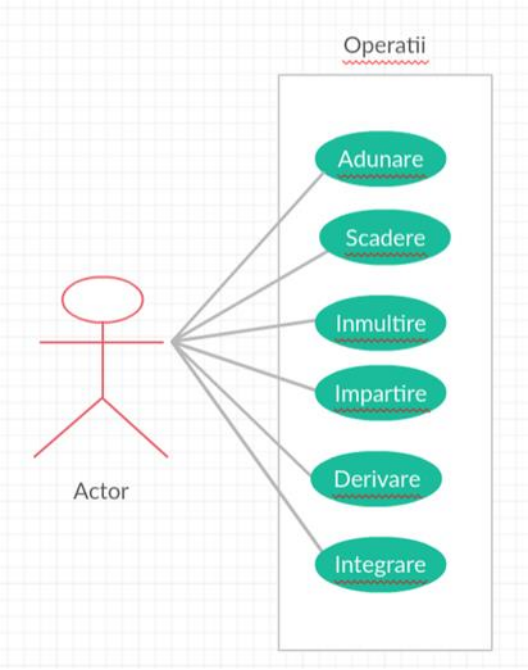
Cerințe funcționale:

- *Utilizatorul poate introduce coeficienții polinoamelor pentru a efectua diferite operații.*
- *Aplicația trebuie să ofere operațiile de adunare, scădere, înmulțire și împărțire a polinoamelor.*
- *Utilizatorul poate efectua operații de integrare și diferențiere asupra unui polinom dat.*
- *Rezultatele operațiilor trebuie să fie afișate în mod clar și ușor de înțeles.*

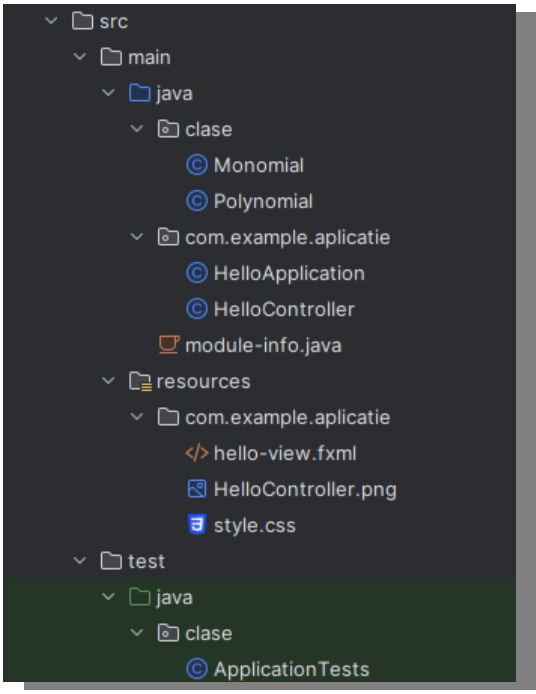
Cazuri de utilizare:

1. *Utilizatorul introduce coeficienții pentru două polinoame și apasă butonul corespunzător operației dorite.*
2. *Utilizatorul poate introduce un singur polinom pentru operațiile de integrare și diferențiere.*

Diagrama use case a proiectului:



3. Proiectare



Pentru o mai buna structura a proiectului am ales sa il impart in pachete: Clase, Resurse, Test si Aplicatie. Primul pachet contine clasele: “Momonial”, “Polynomial”.

Al doilea pachet contine clasele “CalculatorApplication” si “CalculatorController” care impreuna formeaza interfata grafica a utilizatorului.

Cel de al treilea pachet contine clasele “hello-view.fxml”, imaginea cu diagrama UML, si un css pentru style ul fisierului fxml.

Cel de al patrulea pachet contine clasa ApplicationTest, unde fiecare metoda a clasei Polynomial este testata.

1. Pachetul "Clase":

- **Clasa "Monomial":**
    - Această clasă definește un monom, care este un termen al unui polinom și constă dintr-un coeficient și un exponent.
    - Metodele clasei includ getteri pentru coeficient și exponent.
- **Clasa "Polynomial":**
    - Această clasă reprezintă un polinom și oferă operații matematice de bază precum adunare, scădere, înmulțire, împărțire, integrare și diferențiere.
    - Utilizează o TreeMap pentru a stoca monomialele în ordine descrescătoare a exponentului.
    - Metodele includ constructori pentru crearea polinomului din diverse formate (șir de caractere, TreeMap), operații matematice și metode pentru integrare și diferențiere.

2. Pachetul "Resurse":

- **Clasa "CalculatorApplication":**
    - Această clasă este punctul de intrare în aplicație și extinde clasa Application din JavaFX.
    - Inițializează și afișează interfața grafică a utilizatorului, încărcând fișierul FXML și aplicând stilurile CSS.

- **Clasa "CalculatorController":**
    - Această clasă acționează ca un controler pentru interfața grafică a utilizatorului (GUI).
    - Conține metode care gestionează acțiunile utilizatorului, cum ar fi adăugarea, scăderea, înmulțirea, împărțirea, integrarea și diferențierea polinoamelor.

3. Pachetul "Test":

- **Clasa "ApplicationTest":**
    - Această clasă conține metode de testare unitară pentru fiecare metodă a clasei "Polynomial".
    - Folosește utilitarul JUnit pentru a verifica corectitudinea implementării metodelor.

4. Pachetul "Aplicatie":

- **Fișierul "hello-view.fxml":**
    - Acest fișier FXML definește structura interfeței grafice a utilizatorului (GUI) folosind elemente precum textfield-uri, butoane și etichete.
- **Imaginea cu diagrama UML:**
    - Această imagine prezintă diagramele UML pentru clasele "Monomial" și "Polynomial", evidențiind relațiile și structura acestora.
    -
- **Fișierul CSS pentru stilul fișierului FXML:**
    - Acest fișier CSS definește stilul vizual al elementelor din fișierul FXML, cum ar fi culorile, fonturile și alinierea.

Interfata Graphic User Interface (GUI)

Interfața grafică (în engleză: Graphical User Interface sau GUI este o interfață cu utilizatorul bazată pe un sistem de afișaj ce utilizează elemente grafice. Interfața grafică este numit sistemul de afișaj grafic-vizual pe un ecran, situat funcțional între utilizator și

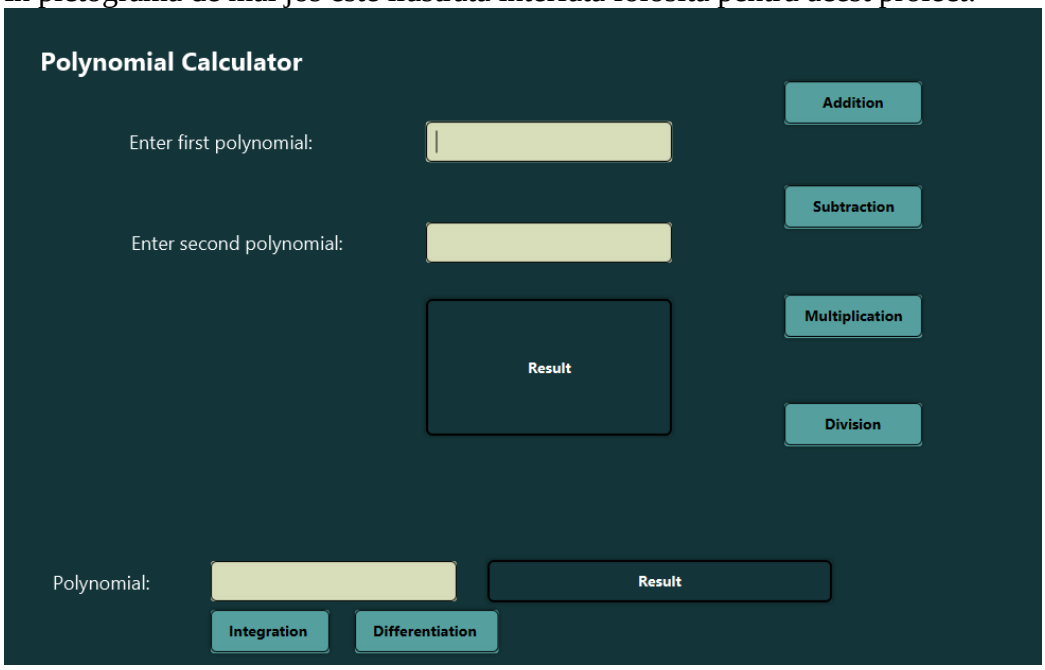
dispozitive electronice cum ar fi computere, dispozitive personale de tip hand-held, aparate electrocasnice și unele echipamente de birou. Pentru a prezenta toate informațile și acțiunile disponibile, un GUI oferă pictograme și indicatori vizuali, în contrast cu interfețele bazate pe text, care oferă doar nume de comenzi (care trebuie tastate) sau navigația text.

Acestea sunt reprezentate de sisteme de programe care, sub o formă sau alta, iniţiază şi întreţin un dialog cu utilizatorul calculatorului, în scopul utilizării şi / sau configurării acestuia.

Avantajele folosirii GUI sunt:

- Interactiune cu computerul mai usoara si mai eficienta pentru utilizator
- Simplificarea instructiunilor complexe, realizata cu ajutorul pictogramelor si a meniurilor
- Initiere intuitiva a comenzilor catre computer
- Programela si fisierele sunt mai usor de manuit si organizat

In pictograma de mai jos este ilustrata interfata folosita pentru acest proiect:



1. **Clasa Monomial:**
   - **Câmpuri:**
     - **coefficient**: reprezintă coeficientul monomului.
     - **exponent**: reprezintă exponentul monomului.
   - **Metode:**
     - **getCoefficient()**: returnează coeficientul monomului.
     - **getExponent()**: returnează exponentul monomului.
2. **Clasa Polynomial:**
   - **Câmpuri:**
     - **monomials**: un **TreeMap** care stochează monomii, cu exponentul ca cheie și coeficientul ca valoare.

```
private final TreeMap<Integer, Double> monomials;
```

Constructor ul: reprezinta una din cele mai complexe notiuni ale proiectului, deoarece pe baza acestui constructor se face parsarea. Astfel utilizatorul introduce un string, urmand sa

se extraga stringul si sa fie transformat intr un polinom. Acest lucru se realizeaza pe baza unui regex

```
private static final Pattern REGEX_EXPR =
Pattern.compile("([+]?(?:(?:\\d*x\\^\\d+)|(?:\\d+x)|(?:\\d+)|(?:x)))");
```

Acest regex obtine fiecare monom al polinomul, construindu l in ordine descrescatoare

```
TreeMap<Integer, Double> tempMonomials = new
TreeMap<>(Comparator.reverseOrder());
```

Metode:

- **addMonomial(int exponent, double coefficient)**: adaugă un monom nou în polinom.
- **add(Polynomial polyToAdd)**: adună doi polinoame.
- **subtract(Polynomial polyToSubstract)**: scade un polinom din altul.
- **multiply(Polynomial polyToMultiply)**: înmulțește două polinoame. Pentru multiply am utilizat foreach de 2 ori astfel parcurgand primul polinom obtinand fiecare monom urmand sa fie inmultit cu fiecare monom al celui de al doilea

```
        for (Map.Entry<Integer, Double> entry1 : monomials.entrySet())
                for (Map.Entry<Integer, Double> entry2 :
polyToMultiply.monomials.entrySet()) {
int newExponent = exponent1 + exponent2;
double newCoefficient = coefficient1 * coefficient2;
```

- **divide(Polynomial divisor)**: împarte un polinom la altul, returnând atât cât și restul. Pentru a implementa divide am avut nevoie de o functie ajutatoare care ma ajuta sa determin gradul cel mai mare dintr un polinom

```
public int highestDegree() {
    int highestDegree = 0;
    for (int degree : monomials.keySet()) {
        highestDegree = Math.max(highestDegree, degree);
    }
    return highestDegree;
}
```

urmand sa parcurg impart polinomul cu cel mai mare grad la celalalt, pentru a obtine un cat si un rest.

```
return new Pair<>(quotient, remainder);
```

- **integrate()**: efectuează operația de integrare asupra polinomului.
- **differentiate()**: efectuează operația de diferențiere asupra polinomului.
- **toString()**: returnează o reprezentare sub formă de șir de caractere a polinomului.

Implementarea Interfeței Utilizator:

Interfața utilizatorului este implementată în clasele **CalculatorApplication** și **CalculatorController**, care sunt parte a pachetului **com.example.aplicatie**.

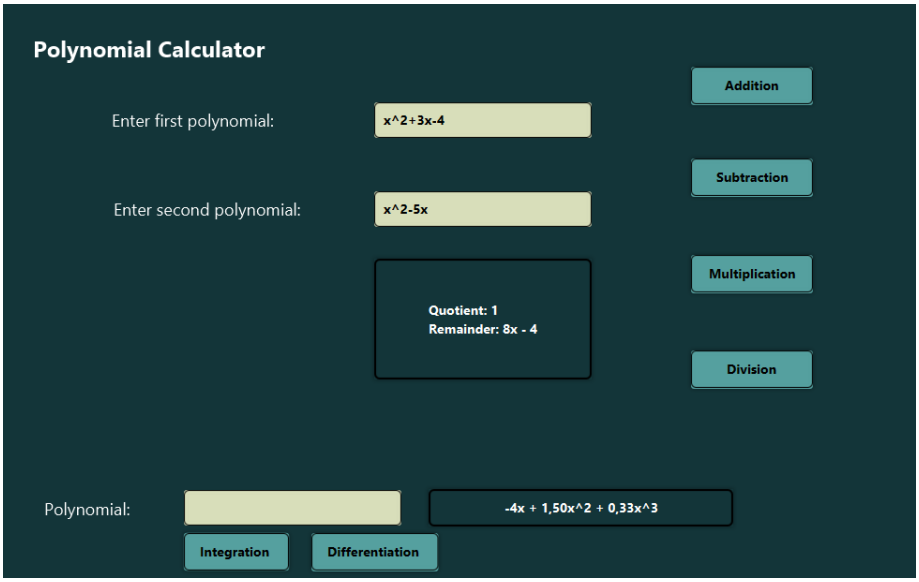
1. **Clasa CalculatorApplication:**
   - Se ocupă de inițializarea și afișarea ferestrei aplicației.
   - Încarcă fișierul FXML și îi atașează un controller.
   - Setează titlul ferestrei și afișează scena.

2. **Clasa CalculatorController:**
   - Gestionează evenimentele de acțiune generate de utilizator.
   - Definește metode pentru fiecare acțiune posibilă în interfața grafică: adunare, scădere, înmulțire, împărțire, integrare și diferențiere.

- Accesează valorile introduse de utilizator din câmpurile de text și afișează rezultatele în etichetele corespunzătoare.
- Această clasă acționează ca un intermediar între interfața grafică și funcționalitatea polinomială din pachetul **clase**.

Fișierul FXML ("hello-view.fxml") conține următoarele elemente:

- **VBox:**
  - Este un container care organizează elementele într-un layout vertical (pe verticală).
  - Are o înălțime și o lățime preferate specificate.
  - Atributul "alignment" este setat la "CENTER", ceea ce înseamnă că elementele vor fi aliniate în centru pe axa verticală.



- **Pane:**
  - Este un container folosit pentru a organiza și a dispune elementele pe scena JavaFX.
  - Are dimensiuni preferate specificate.
- **Text (etichete de text):**
  - Două etichete de text sunt utilizate pentru a afișa instrucțiuni de introducere a polinoamelor.
  - Specifică fontul, dimensiunea și culoarea textului.
- **TextField (câmpuri de text):**
  - Două câmpuri de text sunt folosite pentru introducerea polinoamelor de către utilizator.
  - Fiecare câmp de text are un identificator (fx:id) care este utilizat în codul Java pentru a accesa valorile introduse de utilizator.
- **Label (etichete):**
  - Două etichete sunt folosite pentru afișarea rezultatelor operațiilor matematice (adunare, scădere, etc.).
  - Acestea sunt inițializate cu textul "Result" și vor fi actualizate cu rezultatele calculate în timpul rulării programului.
- **GridPane (grilă):**
  - Este un layout flexibil care organizează elementele într-o grilă de rânduri și coloane.

    - Aici sunt plasate butoanele pentru operațiile matematice (adunare, scădere, înmulțire, împărțire).
- **Button (butoane):**
    - Sunt patru butoane pentru efectuarea operațiilor matematice: adunare, scădere, înmulțire, împărțire.
    - Fiecare buton are un text asociat și o metodă de acțiune specificată, care este apelată atunci când butonul este apăsat.
- **Button pentru integrare și diferențiere:**
    - Două butoane sunt utilizate pentru integrare și diferențiere.
    - Aceste butoane au, de asemenea, metode de acțiune specificate pentru a trata evenimentele de clic.
- **Text (alte etichete de text):**
    - Un text suplimentar este folosit pentru a indica că trebuie introdus un polinom pentru operațiile de integrare și diferențiere.
- **TextField suplimentar:**
    - Este folosit pentru introducerea polinomului pentru operațiile de integrare și diferențiere.
- **Label suplimentar:**
    - Afișează rezultatul operațiilor de integrare și diferențiere.

4. Rezultate

În secțiunea rezultate, anumite excepții și cazuri speciale au fost tratate pentru a asigura o afișare corectă și coerentă a polinoamelor. Mai jos sunt câteva exemple de astfel de cazuri speciale și cum au fost gestionate:

Excluderea coeficienților și exponentului 1 implicit:

Situația: Dacă coeficientul unui monom este 1 sau -1 și exponentul este mai mare decât zero, atunci afișarea sub forma 1x^k sau -1x^k poate fi redusă la simpla formă x^k.
Soluția: În metoda toString() din clasa Polynomial, am inclus o verificare specială pentru coeficienții 1 și -1 și am eliminat afișarea explicită a acestora, astfel încât să se afișeze doar x^k în loc de 1x^k sau -1x^k.

```
if (Math.abs(coefficient) != 1 || exponent == 0) {
   if (Math.floor(coefficient) == coefficient) {
      output.append((int) Math.abs(coefficient));
   } else {
      output.append(decFor.format(Math.abs(coefficient)));
   }
}
```

Tratarea cazului când coeficientul este 0:

Situația: Dacă coeficientul unui monom este 0, acesta nu ar trebui să fie afișat în reprezentarea polinomului.
Soluția: În metoda toString(), am adăugat o verificare pentru a exclude monomii cu coeficientul 0 din afișare.

if (coefficient == 0) { continue; }

Tratarea cazului când polinomul este gol (conține doar monomul cu coeficientul 0):

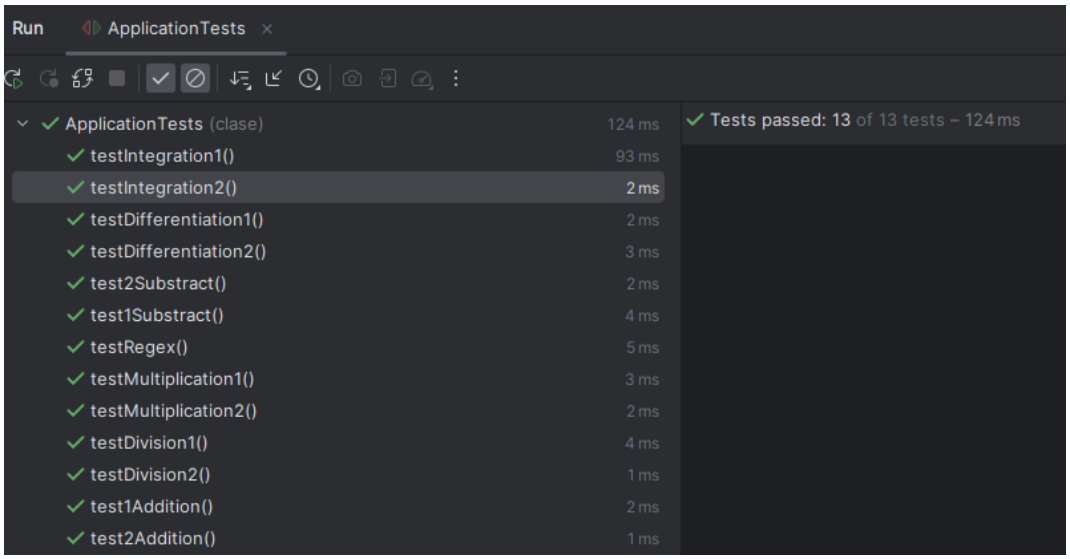
Situația: Dacă polinomul este gol, ar trebui să afișăm explicit că acesta este 0.
Soluția: La finalul metodei toString(), am inclus o verificare pentru a afișa explicit 0 în cazul în care polinomul este gol.

```
if (output.isEmpty()) {
   output.append("0");
}
```

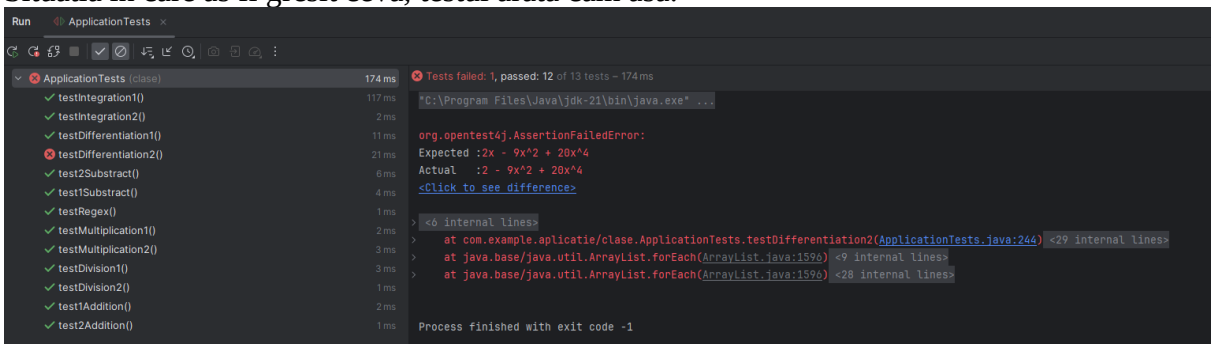
Clasa **ApplicationTest** este responsabilă pentru testarea unitară a funcționalităților din clasa **Polynomial**, utilizând framework-ul de testare JUnit. Aceasta conține mai multe metode, fiecare testând câte o anumită funcționalitate a clasei **Polynomial**. În mod obișnuit, fiecare metodă de test are o structură similară: se pregătesc datele de test, se apelează metoda care urmează să fie testată, iar apoi se verifică rezultatul obținut cu rezultatul așteptat.
Exemple de metode de test includ:

1. **testAddition()**: Acest test verifică corectitudinea operației de adunare a două polinoame. Se creează două polinoame, se adună folosind metoda **add()**, iar rezultatul obținut este comparat cu rezultatul așteptat.
2. **testSubtraction()**: Acest test verifică corectitudinea operației de scădere a două polinoame. Se creează două polinoame, se scad folosind metoda **subtract()**, iar rezultatul obținut este comparat cu rezultatul așteptat.
3. **testMultiplication()**: Acest test verifică corectitudinea operației de înmulțire a două polinoame. Se creează două polinoame, se înmulțesc folosind metoda **multiply()**, iar rezultatul obținut este comparat cu rezultatul așteptat.
4. **testDivision()**: Acest test verifică corectitudinea operației de împărțire a două polinoame. Se creează două polinoame, se împart folosind metoda **divide()**, iar rezultatul obținut este comparat cu rezultatul așteptat, atât pentru cât, cât și pentru rest.

În cadrul fiecărui test, se folosesc aserțiuni pentru a verifica coerența între rezultatul obținut și rezultatul așteptat. Dacă acestea sunt conforme, testul este considerat ca trecut cu succes, altfel, testul eșuează și este necesară investigarea și remedierea problemei.



Situatia in care as fi gresit ceva, testul arata cam asa:



5. Concluzii

Proiectul a atins obiectivele propuse, oferind funcționalități complete pentru operații cu polinoame, implementarea interfeței grafice a fost realizată conform cerințelor.

Dezvoltarea acestui proiect a consolidat cunoștințele în programarea orientată pe obiecte (OOP) și utilizarea interfeței grafice cu utilizatorul (GUI) în JavaFX.
S-au tratat cu atenție excepțiile și s-au efectuat validările necesare pentru asigurarea corectitudinii operațiilor și securității aplicației.

Există potențial pentru extinderea funcționalității și îmbunătățirea aplicației, oferind oportunitatea pentru dezvoltare viitoare și perfecționare, in cadrul GUI poate fi mai interactiva, iar pe baza de cod, se poate extinde la operatii cu numere complexe precum si trasarea graficelor a polinoamelor, inlocuirea variabilei x, aflarea radacinilor
În ansamblu, proiectul a fost o experiență valoroasă, oferind nu doar oportunitatea de a aplica cunoștințele teoretice, ci și de a dobândi abilități practice relevante în domeniul dezvoltării de aplicații software.

6. Bibliografie

1. *Oracle Documentation: JavaFX https://docs.oracle.com/javase/8/javafx/api/toc.htm*
2. *Stack Overflow. https://stackoverflow.com/*
3. *Geeksforgeeks https://www.geeksforgeeks.org/*