DOCUMENTAȚIE

TEMA 1

NUME STUDENT: Iosif Adela

GRUPA: 30223

CUPRINS

1. Obiectivul temei

Se dorește proiectarea și implementarea unui calculator pentru polinoame. Astfel, putem considera că un polinom este format din mai multe monoame (Map<Integer, Monom> monoame). Dar, utilizatorul va introduce polinomul ca un string, iar acesta va trebui să fie interpretat special, utilizând regex (expresii regulate). Mai apoi, vom implementa operații specifice pe polinoame: adunare, scădere, înmulțire, derivare și integrare. De asemenea, este necesară o interfață grafică pentru a-i facilita utilizatorului interacțiunea cu programul.

2. Analiza problemei, modelare

Polinoamele sunt construite din termeni (monoame), fiecare fiind alcătuit din coeficient (o constantă) înmulțită cu una sau mai multe variabile (care pot avea un exponent întreg constant). Totodată, după efectuarea unor operații ca: adunare, scădere, înmulțire, derivare sau integrale pe polinoame, rezultatul obținut va fi tot de tip polinom.

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$ *- forma generală*

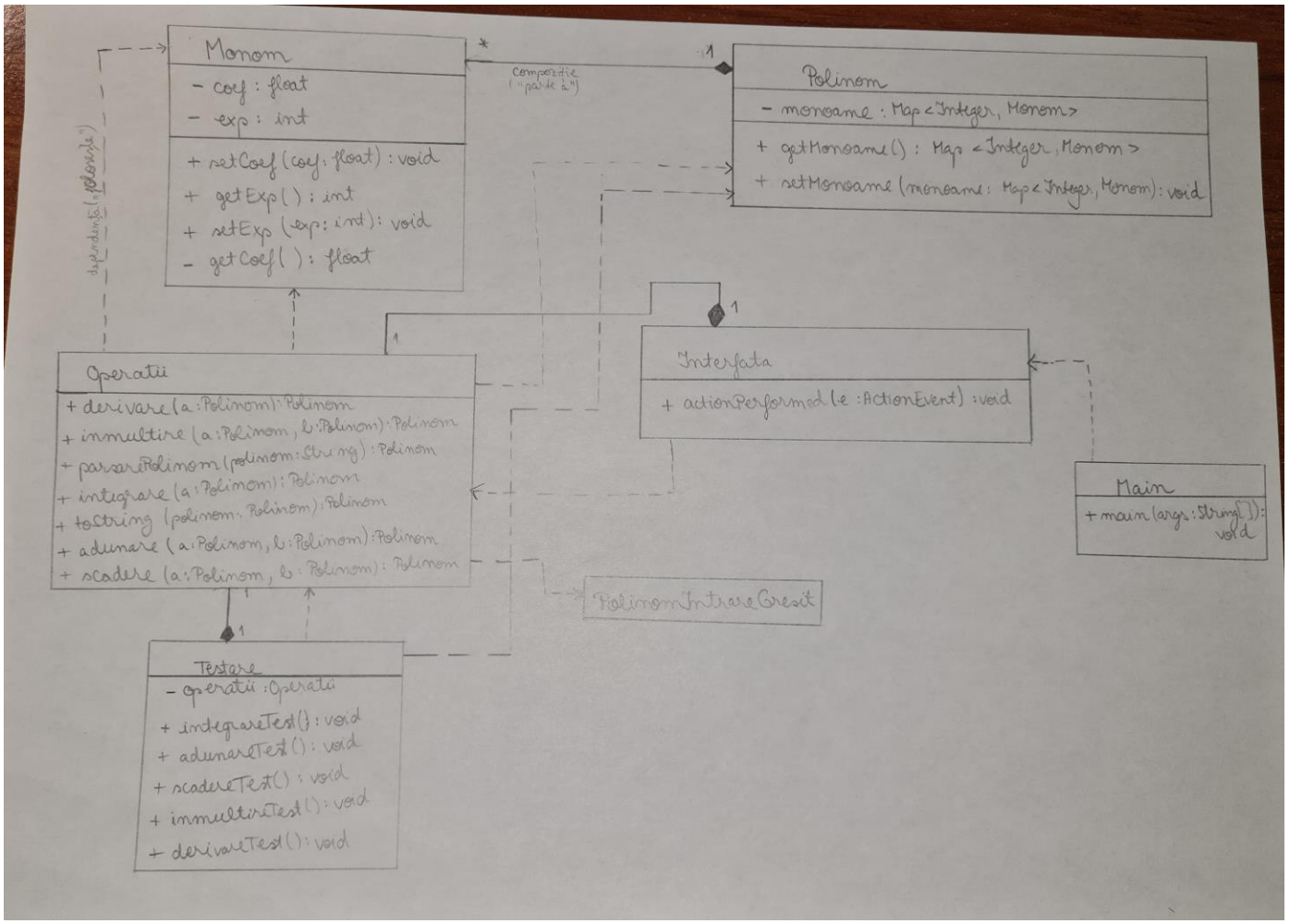
3. Proiectare

Am ales să utilizez 4 pachete pentru o mai bună organizare a codului:

- dataModel: conține clasele Monom, Polinom – pentru proiectarea structurilor de date cu care vom lucra; și clasa PolinomIntrareGresit folosită pentru a trata posibile excepții
- interfata: conține clasa Interfata, care implementează întreg design-ul grafic al aplicației
- operații: conține clasa Operatii, unde sunt implementate toate operațiile pe polinoame, inclusiv parsarea lor și metoda toString
- main: conține doar clasa Main, care deschide aplicația

Mai mult, am implementat și o clasă (Testare) pentru testarea unitară (cu JUnit) a operațiilor pe polinoame.

Diagrama UML



4. Implementare

- Pachetul dataModel

Clasa Monom conține două variabile instanță (private float coef; private int exp;) pentru coeficientul și exponentul monomului; un constructor cu parametri și metode de get și set.

Clasa Polinom descrie modelul de date pe care îl voi folosi mai departe, utilizând un map (încapsulat) între gradul monomului, și monomul propriu-zis private Map<Integer, Monom> monoame).

- Pachetul operatii

Aici sunt implementate operațiile de adunare, scădere, înmulțire, derivare și integrare, folosindu-se structura Map.

De asemenea, este suprascrisă metoda toString, care este utilizată pentru a afișa utilizatorului polinomul rezultat în forma descrisă la proiectare. Sunt tratate și cazuri speciale precum: dacă avem 0 după virgula coeficientului, nu se mai scrie; dacă coeficientul lui x este -1, afișăm doar „-”, nu „-1x” (la fel pentru 1); dacă puterea este 0, se va returna constanta „1”.

Tot în acest pachet găsim și metoda parsarePolinom, care se ocupă de prelucrarea caracterelor introduse de utilizator.

String pattern = "((\\+|\\-|^)(\\d*))x{0,1}(\\^\\-{0,1}(\\d+))*";

- Explicații:

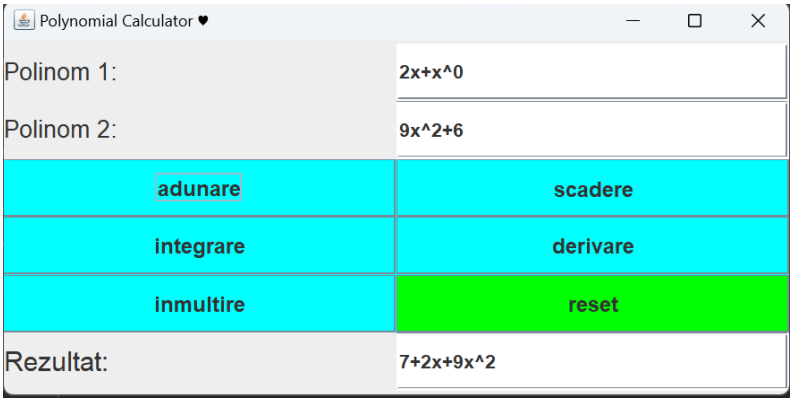
((\\+|\\-|^)(\\d*)) – acest grup se ocupă de coeficientul variabilei; face match dacă semnul este +, - sau dacă nu este semn, iar apoi verifică existența cifrelor (poate să nu fie, să existe una sau mai multe)

x{0,1} – caracterul x poate să apară o dată sau niciodată

(\\^\\-{0,1}(\\d+))* - match-uiește exponentul; el trebuie să înceapă cu ^, iar dacă e negativ poate conține un „-”, urmat de una sau mai multe cifre (\\d+); exponentul este optional („*”)

- Pachetul interfata

Cuprinde toate componentele cu care va interacționa utilizatorul (butoane, label-uri, text field-uri).



5. Rezultate

Am folosit testarea unitară (cu JUnit) pentru verificarea funcționării corecte a metodelor pentru polinoame. Am declarat două string-uri (sau unul pentru derivare și integrare) și am furnizat rezultatul corect care ar trebui să apară, iar dacă este generat corespunzător, testul se va finaliza cu succes.

```
private final Operatii operatii = new Operatii();                    ⚠ 4

no usages  Adela06 *
@Test
public void adunareTest() {
    String s1 = "x^2+2x^0";
    String s2 = "7x^3+6x";
    Polinom rezultat;
    try {
        rezultat = operatii.adunare(operatii.parsarePolinom(s1), operatii.parsarePolinom(s2));
        Assert.assertEquals( expected: "2+6x+x^2+7x^3", operatii.toString(rezultat));

    } catch (PolinomIntrareGresit ex) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Nu coincide rez",
                title: "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

6. Concluzii

Tema a fost utilă pentru învățarea unor concepte noi precum regex sau testare unitară, dar și pentru aprofundarea structurilor de date din Java (ex: Map) cu care nu am lucrat atât de mult până acum. De asemenea, am învățat să organizăm codul mai eficient în pachete.

Câteva îmbunătățiri care ar putea fi aduse proiectului sunt: afișarea ieșirii în ordinea descrescătoare a gradelor polinoamelor; introducerea unei tastaturi care să poată fi accesată cu mouse-ul pe ecran; stilizarea interfeței grafice; adăugarea mai multor operații (împărțire, transformata Laplace, etc.).

7. Bibliografie

1. Google Java Style Guide.
2. https://www.w3schools.com/java/java_regex.asp
3. https://www.javatpoint.com/java-regex
4. https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html
5. https://docs.oracle.com/