



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

ASSIGNMENT 1

POLYNOMIAL CALCULATOR

NUME STUDENT: Boar Daniel-Ioan
GRUPA: 30223

Cuprins

1. Obiectivul temei.....	2
2. Analiza problemei, modelare, scenarii, cazuri de utilizare.....	2
3. Proiectare.....	2
4. Implementare.....	3
5. Rezultate.....	6
6. Concluzii.....	8
7. Bibliografie.....	8

1. Obiectivul temei

Obiectivul propus in rezolvarea acestei teme este de a realiza un program care preia, prin intermediul unei interfete user friendly, doua polinoame si efectueaza anumite operatii asupra acestora. Operatiile realizate pot fi de doua feluri: operatii care folosesc ambele polinoame sau operatii care se refera doar la un singur polinom. Spre exemplu, adunarea, scaderea, inmultirea si impartirea sunt operatii care se efectueaza intre cele doua polinoame, iar derivarea si integrarea sunt operatii care se refera doar la un polinom.

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

Pentru realizarea aplicatiei am folosit 4 clase si inca o clasa pentru testare. Aceste clase sunt: clasa **Polinom** in care ne creem polinomul si avem mai multe metode pentru a ne ajuta in rezolvarea problemei; clasa **Operatie** in care avem implementate operatiile de adunare, scadere, inmultire, impartire, derivare si integrare; clasa **Interfata** unde avem creata interfata user friendly si unde am legat fiecare buton de operatia specificata; si clasa **Main** unde avem doar o apelare a metodei „Interfata()”. Clasa pentru testare se numeste **OperationTest**, iar aici am testat cate un exemplu pentru fiecare operatie, in afara de impartire, folosind JUnit.

Pentru pasarea polinoamelor am folosit o metoda care implementeaza un regex astfel incat noi cand introducem string-ul („2x²+2”), regexul ne va extrage pe rand, coeficientul si gradul fiecarui monom si il va introduce in HashMap.

Exemplu de introducere a polinoamelor:

- $2x^3+4x^2-x+7$
- $-x^2+3x-2$

In momentul in care dorim sa afisam aceste doua polinoame, ele vor arata astfel:

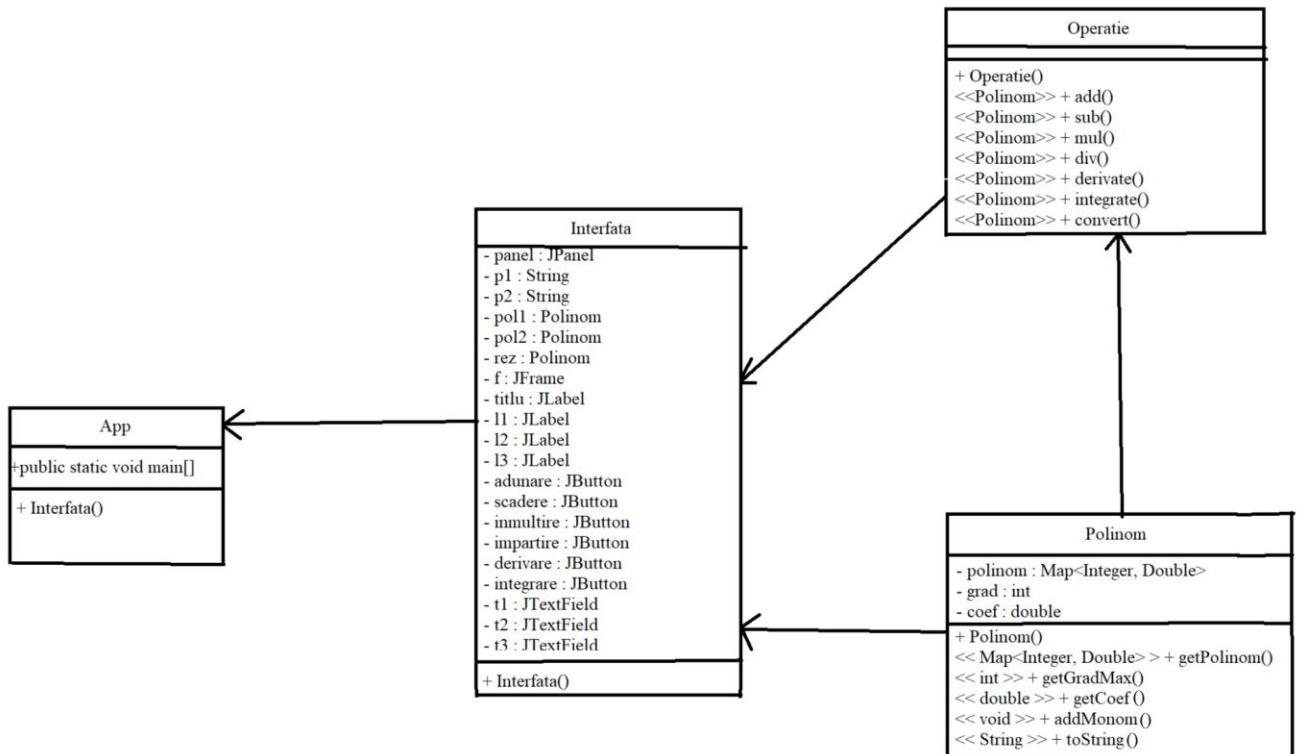
- $2.0*x^3+4.0*x^2-x+7.0$
- $-x^2+3.0*x-2.0$

3. Proiectare

In momentul in care am suprascris functia toString() folosita pentru afisarea polinomelor, am tinut cont de toate aspectele necesare pentru ca afisarea polinomului sa fie facuta cat mai profesionist. Astfel, unde gradul monomului este 1, acesta va afisa doar coeficientul sau urmat de x, iar acolo unde gradul este 0 si coeficientul este diferit de 0, va afisa doar coeficientul. Am tratat si cazul in care nu imi pune „+” la inceputul polinomului daca coeficientul este pozitiv (ex $2x^2$ nu imi pune $+2.0x^2$).

Toate testele efectuate pana acum au demonstrat faptul ca operatiile pe polinoame functioneaza asa cum ar trebui, mai putin impartirea acestora, operatie pe care nu am implementat-o inca.

Diagrama UML a claselor



4. Implementare

Dupa cum se poate vede si in diagrama UML a claselor, aplicatia noastra dispune de 4 clase. Aceste clase impreuna cu metodele si functionalitatile lor vor fi descrise mai jos.

• Clasa Polinom

Aceasta clasa are 2 attribute: coeficientul si gradul polinomului. Acestea sunt private deoarece utilizam conceptul de incapsulare a datelor. Pe langa aceste attribute mai avem declarat polinomul ca fiind un `Map<Integer, Double>`, coeficientul fiind `double`, iar gradul intreg. Asa cum ni se precizeaza in cerinta temei, clasa `Polinom` dispune de un `Map` de `integer` si `double`.

Constructorul clasei este folosit pentru a crea un nou obiect de tip `Polinom` de tip `HashMap`.

```
public Polinom()
```

Dupa acest constructor se afla metoda `getPolinom` care ne returneaza polinomul creat.

```
public Map<Integer, Double> getPolinom()
    return this.polinom;
```

Mai avem o metoda care ne returneaza gradul maxim al unui polinom. Aceasta metoda am folosit-o pentru a sti in afisarea polinoamelor care este gradul maxim pentru a afisa descrescator polinomul.

```
public int getGradMax()
```

O alta metoda folosita este cea care ne returneaza coeficientul unui grad dat ca input. Am folosit-o pentru a-mi fi mai usor sa gasesc coeficientul gradului dorit din polinom.

```
public double getCoef(int grad)
```

Am creat si o metoda de „addMonom” care imi adauga un monom in polinomul nostru, avand ca input-uri gradul si coeficientul monomului. Puteam folosi si metoda de „put” din Map, doar ca m-am folosit mult si de metoda mea.

```
public void addMonom(int grad, double coef)
```

In afara de metodele prezentate mai sus, mai avem o metoda destul de stufoasa numita toString. In aceasta metoda am incercat sa fac un pretty-print a polinomului. Pentru a face acest lucru, m-am gandit la toate posibilitatile pentru ca un polinom sa arate cat mai frumos, astfel, daca gradul polinomului este 0, nu-l vom mai afisa pe x, vom afisa doar coeficientul monomului, daca gradul monomului este 1 vom afisa doar coeficientul*x, iar daca coeficientul gradului cel mai mare este pozitiv, nu vom afisa „+”-ul din fata coeficientul.

```
public String toString()
```

• Clasa Operatie

Clasa Operatie este o clasa destul de mare si foarte utila. In aceasta clasa se implementeaza toate operatiile pe polinoame plus metoda de convertire a polinomului (folosind regex). Aceasta clasa nu are niciun atribut si nici un alt constructor decat cel implicit.

Metoda de adunare a 2 polinoame, care primeste 2 polinoame ca parametru si returneaza polinomul rezultat care este declarat in interiorul clasei. Pentru a realiza adunarea polinoamelor se foloseste urmatoare strategie: se iau, rand pe rand, elementele, pornind de la gradul maxim al celor 2 si se aduna coeficientii gradelor egale. Coeficientul fiecarui grad este adaugat in polinom pe pozitia gradului la care suntem.

```
public static Polinom add(Polinom p1, Polinom p2)
```

Dupa cum sugereaza si numele, in aceasta metoda se face scaderea celor doua polinoame, respectiv din polinomul p1, se scade polinomul p2. De fapt, aceasta metoda este asemanatoare metodei de adunare, totul este identic, ce difera este ca semnul „+” a devenit semnul „-”.

```
public static Polinom sub(Polinom p1, Polinom p2)
```

Metoda de inmultire a doua polinoame nu a fost greu de implementat. Se parcurg cele 2 polinoame cu 2 for each-uri, iar gradele se aduna si coeficientii se inmultesc. Ce a trebuit sa verific in plus a fost in cazul in care gradul la care suntem a avut deja coeficient si trebuia sa adun coeficientul vechi cu coeficientul nou calculat. Rezultatul este adaugat intr-un polinom nou care a fost declarat in metoda.

```
public static Polinom mul(Polinom p1, Polinom p2)
```

Metoda derivate este metoda care implementeaza metoda de derivare a primului polinom primit ca parametru. Polinomul primit este parcurs monom cu monom si pe acesta se aplica metoda de derivare. In cazul in care gradul monomului este mai mare ca 0, gradul curent se inmulteste cu coeficientul monomului si mai apoi se seteaza gradul monomului ca fiind gradul precedent – 1.

```
public static Polinom derivate(Polinom p1)
```

Metoda de integrare a unui polinom functioneaza in felul urmatoar: se parcurge polinomul monom cu monom, gradul se incrementeaza , iar coeficientul se imparte la gradul monomului dupa incrementare. Din cauza ca se imparte un double la int, am afisat coeficientul cu 2 zecimale pentru a fi mai usor de citit polinomul.

```
public static Polinom integrate(Polinom p1)
```

Metoda de impartire a doua polinoame primeste 2 polinoame ca si parametrii si furnizeaza rezultatul intr-un polinom declarat la inceputul metodei. Am incercat cateva implementari ale metodei, insa nu mi-au iesit, deci metoda nu este implementata.

```
public static Polinom div(Polinom p1, Polinom p2)
```

Ultima metoda implementata din aceasta clasa este o metoda pe care eu am numit-o „convert”. Aceasta metoda primeste un string sub forma de polinom si imi va returna polinomul, dupa ce a extras din acel string fiecare grad si coeficientul fiecarui grad. Pentru a implementa aceasta metoda m-am folosit de un regex care imi gaseste in string un grup care reprezinta fie coeficientul, fie gradul polinomului. Dupa cum am spus, daca se gasesc gradul si coeficientul unui monom, acestea se vor adauga in polinom.

```
public static Polinom convert(String s)
```

- **Clasa Interfata**

Clasa Intefata dispune de 6 butoane (Add, Substraction, Multiply, Division, Derivative, Integral) si de 3 textField-uri. Primele doua textField-uri sunt pentru introducerea polinoamelor, iar cel de-al 3-lea numit „Result” afiseaza rezultatul la operatii.

In aceasta clasa sunt definite aceste butoane, dar si label-uri, dimensiunea ferestrei cat si tratarea evenimentelor, cum ar fi cand apasam un buton, sa ne efectueze o operatie specifica butonului. In fiecare operatie, aveam declarate 2 stringuri in care se citeau cele 2 polinoame,

fiecare string este convertit in cate un polinom folosind metoda „convert” din clasa „Operatie”, iar rezultatul era afisat in al 3-lea textField, dupa cum am explicat si mai sus.

- **Clasa Main**

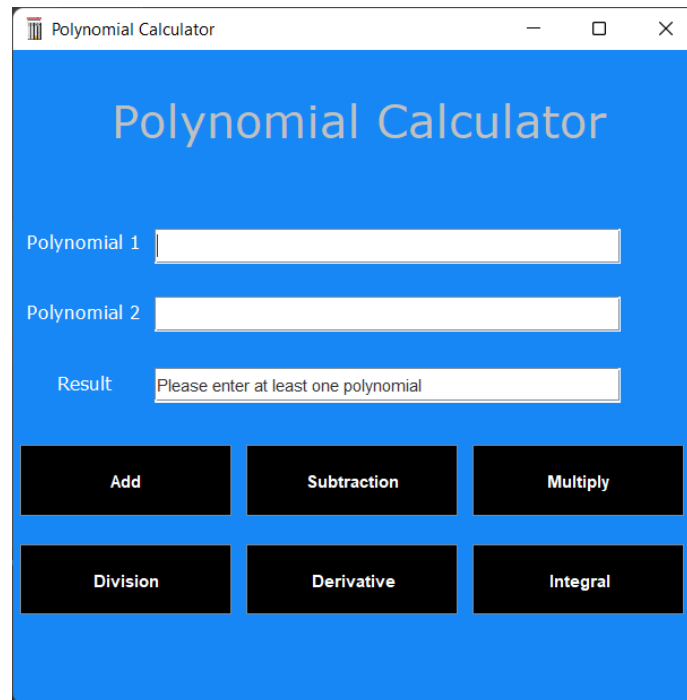
Aceasta clasa este de fapt clasa care implementeaza functia main, adica locul unde setam Interfata ca fiind vizibila.

- **Clasa OperationTest**

Aceasta este o clasa in care am folosit JUnit. Aici am testat fiecare operatie pe cate un exemplu folosind metoda „assertEquals” ce are ca prim input rezultatul la operatia facuta de mine, iar al doilea input este echivalentul rezultatului pe care il astept sa il returneze operatia. Operatia de impartire nu a fost testata deoarece nu este implementata.

5. Rezultate

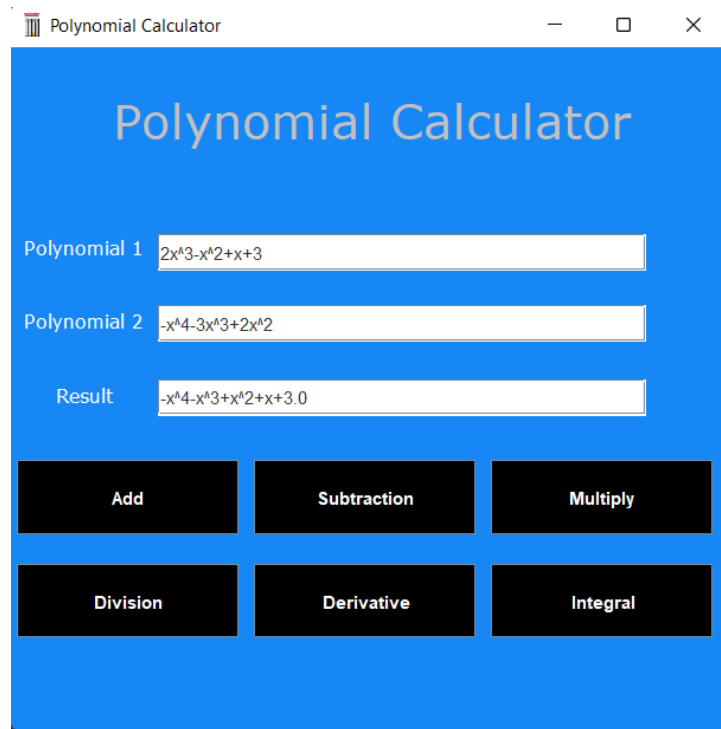
In momentul in care rulam aplicatia, ea arata astfel:



Dupa cum se vede, aplicatia asteapta ca input cel putin un polinom pentru a putea efectua operatiile. Daca nu se va introduce niciun polinom si incercam sa apasam pe un buton, ne va afisa la rezultatul numarul 0.

Daca introducem ca input un string care nu este de tipul polinom, de exemplu „a@b%#@!#”, atunci rezultatul va fi tot 0.

Haideti sa testam programul nostru si sa introducem ca input-uri, primul polinom sa fie „ $2x^3-x^2+x+3$ ” iar al doilea polinom sa fie „ $-x^4-3x^3+2x^2$ ”. Pentru aceste input-uri vom dori sa se efectueze operatia de adunare.



Polynomial Calculator

Polynomial 1

Polynomial 2

Result

Add Subtraction Multiply

Division Derivative Integral

Dupa cum se poate vedea, calculatorul nostru a efectuat operatia cu succes, rezultatul fiind vizibil in textField-ul 3.

Am efectuat si 5 testari cu JUnit pentru operatiile adunare, scadere, inmultire, derivare si integrare. Operatiile au fost cu efectuate succes, dupa cum se poate vedea si in poza de mai jos:

✓ OperationsTest (org.example)	68 ms
✓ subTest()	43 ms
✓ integrateTest()	22 ms
✓ addTest()	1 ms
✓ derivativeTest()	1 ms
✓ mulTest()	1 ms

6. Concluzii

În concluzie, consider că această temă a fost foarte benefică pentru a ne dezvolta modul de lucru în Java în momentul în care vom avea proiecte mai mari de făcut.

La început, problema implementării unei aplicații Java care să opereze pe polinoame a părut destul de simplă, dar când a trebuit să fac implementarea propriu-zisă, am întâmpinat o multitudine de erori și câteva probleme de logică deoarece nu am încă formată foarte bine gândirea pentru programarea orientată pe obiecte. Cel mai mult mi-a creat dificultate metoda de `toString` a polinoamelor și metoda de `convert` deoarece a fost prima dată când m-am lovit de implementarea unei metode folosind `regex` și mi-a luat puțin timp până l-am înțeles cât de cât.

Ca dezvoltări ulterioare, va trebui să implementez metoda de împărțire deoarece nu am reușit să o fac să funcționeze deloc și cel mai probabil va trebui să mai lucrez la eficiența programului meu sau va trebui să fac interfața grafică mult mai prietenoasă și mai ușor de folosit.

7. Bibliografie

- [1] <https://regex101.com/>
- [2] <https://stackoverflow.com/questions/>