

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

DOCUMENTATIE

TEMA  *1*

NUME STUDENT: Solomonean Dan-Adrian

GRUPA: 30222

# Obiectivul temei

Obiectivul principal al temei este implementarea unui calculator de polinoame. In principiu, utilizatorul va putea introduce doua polinoame si va putea alege ce operatie se va efectua cu unul din ele sau intre ele. In primul rand, o sa avem nevoie de un mod de a procesa polinomul primit de la utilizator, apoi vor trebui implementate toate operatiile si apoi o interfata grafica.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Legat de analiza problemei, un polinom este o expresie alcatuita din mai multe monoame, care la randul lor au in componenta un coeficient si un exponent. Descriere pentru fiecare use-case:

* Add use-case: Ca si utilizator, vreau sa fiu capabil sa adun doua polinoame.
* Subtract use-case: Ca si utilizator, vreau sa fiu capabil sa scad doua polinoame.
* Multiply use-case: Ca si utilizator, vreau sa fiu capabil sa inmultesc doua polinoame.
* Differentiate use-case: Ca si utilizator, vreau sa fiu capabil sa derivez polinoam.
* Integrate use-case: Ca si utilizator, vreau sa fiu capabil sa integrez un polinoam.
* Divide use-case: Nu a fost implementata operatia de impartire.

Pentru fiecare use-case, utilizatorul va putea alege un polinom/doua polinoame si sa aleaga operatiile dintre ele.

O imagine care conține diagramă

Descriere generată automat

# Proiectare

O imagine care conține diagramă

Descriere generată automat

Pentru inceput, avem pachetul DataModels, care contine modelele pentru un monom si un polinom. Eu am mai folosit si o clasa Parser, din pachetul “BusinessLogic”, care este folosita in interiorul constructorului din clasa “Polynomial”, aceasta are rol de a parsa string-ul primit. De asemenea, am adaugat si o exceptie noua care este aruncata daca un utilizator introduce un format gresit al polinomului. Pentru a stoca monoamele, in clasa “Polynomial” s-a folosit un TreeMap, care are ca si cheie un Integer, iar ca si valoare un “Monomial”, am ales sa folosesc TreeMap deoarece aceasta are cheile sortate in ordine crescatoare by default si contine si o metoda care le poate sorta si descrescator. Tot in pachetul “BusinessLogic” avem si clasa de “Operations”, in aceasta clasa sunt implementate metodele care se pot efectua pe polinoame. Exista si o clasa pentru unit teste, numita “Tests” si o clasa pentru interfata grafica, numita “PolynomialCalculatorUI”. La randul lor si acestea sunt in pachete diferite, dar nu am vrut sa incarc diagrama UML prea mult. Iar, in clasa main, se apeleaza pur si simplu o instanta a clasei “PolynomialCalculatorUI”.

# Implementare

Clasa Monomial, un monom este alcatuit dintr-un coeficient si un exponent. Clasa contine un constructor care primeste ca parametrii coeficientul si exponentul monomului, metode basic de get/set, o metoda “toString()” care este folosita pentru a afisa monomul si o metoda suprascrisa “equals”, care a fost implementata pentru a compara daca un monom este egal cu alt monom. Am suprascris metoda deoarece, in clasa pentru teste, am considerat ca doua polinoame sunt egale daca au TreeMap-urile egale, un TreeMap avand ca si valoare un monom, a fost nevoie suprascrierea metodei “equals()” din clasa “Monomial”.

Clasa Polynomial, contine un tree map cu cheia “Integer” si valoarea “Monomial”, un constructor gol, un constructor in care se primeste ca parametru un string, metode get/set pentru map.

Clasa “Parser”, aceasta clasa contine doar o metoda statica, numita “parsePolynomial(String p)”, care primeste ca parametru un string si returneaza un TreeMap<Integer, Polynomial>. Pentru a parsa polinomul, s-au folosit doua regex-uri, primul regex o sa faca match si cu eventuale neconformitati ale formatului string-ului introdus de user, de exemplu “3x”, nu este un string valid pentru a fi procesat, in cazul acestui calculator, ar trebui scris “3x^1”, dar primul regex il va lua oricum pe 3x, iar al doilea regex are rol de a valida formatul, in caz ca formatul este gresit se arunca o exceptie.

Clasa “Operations”, aceasta clasa contine operatiile care se pot efectua pe polinoame. In acest caz, se pot efectua toate operatiile in afara de operatia de impartire. Metoda “addPolynomials(..)”, aceasta metoda primeste ca parametrii doua polinoame si returneaza tot un polinom. Pentru a efectua adunarea, se cauta in map-ul celui de-al doilea polinom, daca gasim in map-ul primului polinom o cheie din al doilea, adica un exponent in acest caz, pur si simplu adunam coeficientii, daca nu exista in primul polinom, atunci aduagam tot monomul. La fel si in cazul scaderii, doar ca se scad coeficientii.

Pentru metoda de inmultire, “multiplyPolynomials(Pol p1, Pol p2)”, se parcurge map-ul primului polinom, salvam exponentul si valoarea ( monomul), parcurgem al doilea map, salvam din nou monomul si exponentul, cream un nou exponent care este suma celorlalti doi, si inmultim coeficientii intre ei. Avem un empty map initial, In care adaugam intrari create, acesta il folosim si sa verificam daca cumva am adaugat deja un polinom cu un anumit exponent, daca da, atunci doar adunam coeficinetiii monoamelor.

Pentru metoda de derivare, “differentiatePolynomial(Pol p), se parcurge map-ul polinomului, se salveaza exponentul si monomul, dupa care se adauga intr-un map nou, valoarea exponentului salvat – 1 si un Monom nou, care are ca exponent, cel salvat – 1, iar ca si coeficinet, produsul dintre exponent si coeficientul actual.

Pentru metoda de integrare, la fel ca si pentru derivare, doar ca se aduna 1 la exponent, si coeficientul devine coeficientul precedent impartit la exponent + 1.

Clasa “Tests”, contine trei polinoame care au fost folosite pentru a efectua testele, acestea sunt initializate si curatate dupa fiecare test, folosind anotatiile “BeforeEach” si “AfterEach”, doua dintre polinoame au o anumita valoare, iar unul este un polinom gol.

Pentru interfata grafica, am folosit form generatorul disponibil in Intelij, acesta generaza un form si o clasa legata de acesta. In cazul acesta, clasa se numeste “PolynomialCalculatorUI”, si in constructorul ei sunt ascultate eventurile de click pe butoane. Practic, clasa contine field-uri pentru fiecare componenta din form, aceasta poate fi modificata si din interiorul clasei sau direct din form.

La fiecare buton, este cate un listener, cand este apasat, acesta ia text-ul introdus in fieldu-uri si apeleaza metodele corespunzatoare. La sfarsit, rezultatul este afisat in cel de-al treilea field.

# Rezultate

S-au testate diferite scenarii, acestea reies din numele metodelor de test, consider ca nu este necesara o descriere.

O imagine care conține text, monitor, captură de ecran

Descriere generată automat

# Concluzii

In conculzie, cred ca acest proiect a fost o lectie buna, in special partea de regex. Ca si dezvoltare ulterioara, ar trebui implementata si operatia de impartire si ar trebui, cel mai probabil, modificarea formatului permis pentru polinoame, de exemplu “2x+1” sa poata fi interpretat si tratat ca atare, nu cu un mesaj de eroare.

# Bibliografie

https://www.youtube.com/watch?v=\_Ph9WsCm4Uk