

DOCUMENTATIE

TEMA 1

(Sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi)

Luca Andrada Aurelia

Grupa 30228

CUPRINS

**1.** Obiectivul temei

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

**3.** Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

**4.** Implementare si testare

**5.** Rezultate

**6.** Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

**7.** Bibliografie

**OBIECTIVUL TEMEI**

**Cerinta:**

Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

**Obiective:**

Realizarea unei aplicatii Java care se realizeze o serie de operatii asupra unor polinoame:

* Citirea si afisarea unui polinom
* Operatii aplicabile unui singur polinom (derivare, integrare) si operatii aplicate mai multor plolinoame(adunare, scadere, inmultire)
* Implementarea unui interfete grafice cu utilizatorul
* Testarea aplicatie Java, folosind JUnit.

**2. Analiza problemei si modelarea acesteia**

Prin analiza problemei, ne referim la un prim set abstract de operatii si proprietati prin care încercam sa depistam eventualele însusiri și comportamente ale proceselor necunoscute. Programarea orientata ne oferă aici un avantaj clar, tocmai fiindcă ea permite să taclam problema de la un nivel superiror, fara a mai fi constrasi, într-o asa masura, de caracteristicile tehnice.

Această strategie de conceptualizare, mai poartă numele si de bottom-up design. Este foarte avantajoasa din prisma găsirii componentelor constituente, deoarece pot fi gasite, relativ usor, structuri cu o legatura directă în lumea reala( obiecte, acțiuni etc.).

De obicei se porneste da le documentatia proiectului:

* Cautam substantive legate de proiect (in cazul nostru Monom si Polinom)
* Cautam verbe legate de proiect (in cazul nostru: aduna , scade, inmulteste, deriveaza, integreaza)

De ce cautam substantive si verbe? Deoarece substantivele sunt candidatii perfecti pentru a deveni clase, iar verbele pentru a deveni metode.

Odata realizat pasul de mai sus, ar trebui să avem o idee foarte generala asupra problemei. Pasul urmator constă în descrierea functionala a acesteia

Programul va putea fi accesat de mai multi utilizatori, de aceea interfata cu utilizatorul devine punctul de pornire al proiectului. Ea trebuie sa permita, într-o maniera cat mai convenabila, comunicarea utilizatorului cu aplicatia. Astfel prin intermediul interfetei vom introduce polinoame sub forma de string-uri de tipul „2x^3+3x^0” , modul cel mai utilizat de reprezantare a polinoamelor, iar apoi for fi prelucrate astfel incat sa se poate folosi coeficientii si gradurile in cadrul operatiilor. Polinomul va fi implementat ca si o lista de monoame (ArrayList).

Faptul ca cerinta presupune introducerea unor polinoame cu coeficienti intregi va genera o eroare la efectuarea anumitor operatii, cum sunt integrarea sau eventual impartirea. De aceea am ales solutia declararii coeficientilor de tip double.

De asemenea, cand gradul unui monom va fi zero, se va afisa doar coeficientul acestuia, fiind o utilizare mai generala a polinoamelor.

Utilizarea polinoamelor este prezenta in cea mai mare parte in cadrul graficii pe calculator, spre exemplu ajutand la determinarea pozitiei anumitor obiecte fata de altele,si pentru formarea curbelor din roller-coaster.

**Cazuri de utilizare**

* User-ul introduce un polinom “dezordonat”, iar rezultatul este un polinom minificat (verificat) si ordonat
* Suma/Diferenta si Produsul între primul polinom si al doilea
* Calcularea rezultatelor numerice pentru primul polinom și al doilea
* Caractere speciale utilizate in stringurile ce contin polinoame

**Scenarii**

Utilizarea normala a programului presupune introducerea celor doua polinoame in formatul clasic , în mod aleator, iar apoi se poate sa se efectueze operatiile elementare existente sub TextField-uri. La fiecare din operatiile mentionate mai sus, polinomul rezultat se va afisa ordonat descrescator intr-un TestField.

Totusi, cazul de mai sus este putin probabil si adesea lucruri neasteptate pot sa apars. Dintre acestea se remarca :

* Introducerea de text, fara semnificatie numerica, în campurile test
* Introducerea de necunoscute suplimentare(pe langă x/X)
* Folosirea altor cuvinte cheie decat cele specificate
* Scriere corecta a polinoamelor, dar moduri sintactice care difera putin de la user la user( de ex. 2x + 2 vs. 2\*x^3+5\*x)

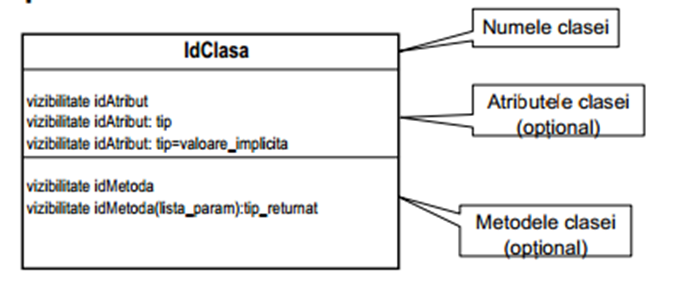
3. PROIECTARE

Proiectarea acestui sistem de calcul se bazeaza pe trei clase Monom, Polinom si Operatii legate printr-o relatie de dependenta. Clasa Monom este caracterizata de doua variabile ce reprezinta gradul si coeficientul, in clasa Polinom este declarat un ArrayList<Monom> in care sunt adaugate monoamele corespunzatoare polinomului instantiat, iar calsa Operatii este clasa in care sunt implemamntate toate operatiile pe polinoame.

**Diagrame UML:**

UML este notaţia internaţională standard pentru analiza şi proiectarea orientată pe obiecte. Diagramele UML de clase sunt folosite in modelarea orientata obiect pentru a descrie structura statica a sistemului, modului in care este el structurat. Ofera o notatie grafica pentru reprezentarea: claselor - entitati ce au caracteristici comune relatiilor - relatiile dintre doua sau mai multe clase , Reprezentarea UML a claselor.

Structura unei diagrame UML:

****

Toate relatiile din cadrul diagramei UML sunt reprezentate grafic printr-o succesiune de segmente orizontale sau verticale care leaga o clasa de alta.

Asocierea este o conexiune intre clase, ceea ce inseamna o legatura semnaticaintre obiectele claselor implicate in relatie.

Specificarea atributelor:

**vizibilitate idAtribut : tip = valoare\_implicitia** unde:

-vizibilitate reprezintă protecția atributului și poate să aibă una din următoarele valori: + = public, - = privat si # = protected

- idAtribut este identificatorul atributului

- tip – este tipul acestuia

- valoare\_implicita reprezintă valoarea inițială a atributului

Specificarea metodelor:

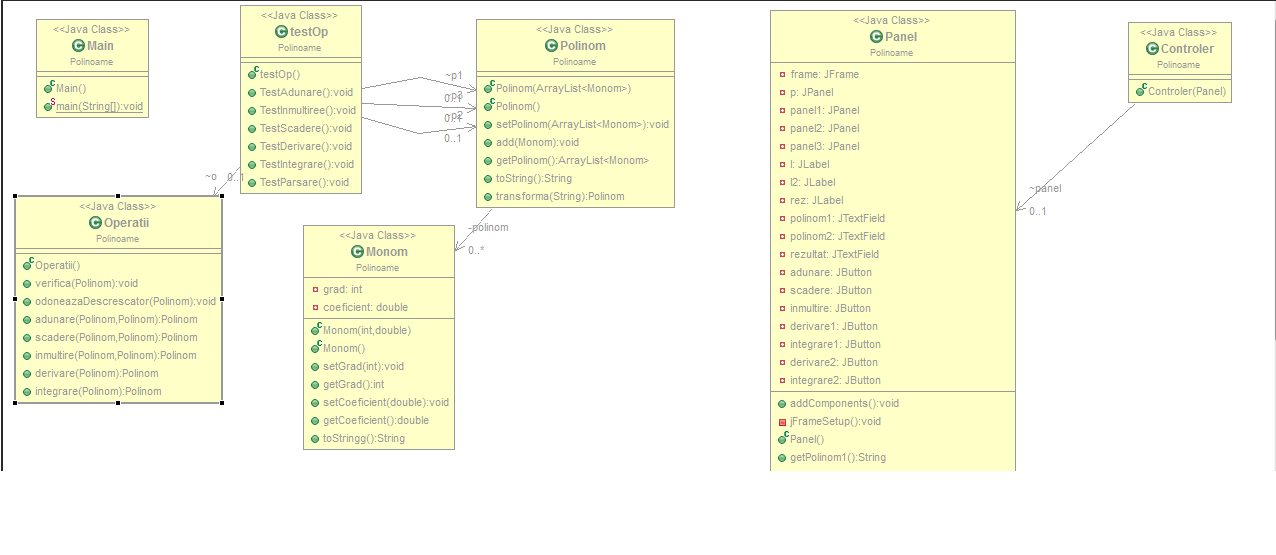
**vizibilitate idMetoda(idP1:tip1, ..., idPn:tipn) : tip\_returnat** unde:

-vizibilitate reprezintă protecția metodei. Poate să aibă una din următoarele valori: + = public, - = privat si # = protected

-idMetoda este identificatorul metodei

- idP1,..., idPn sunt parametri metodei

-tip1, ..., tip n sunt tipurile parametrilor ¨ tip\_returant reprezintă tipul valorii returnate de metoda.



**Structuri de date:**

Structura de date utilizata este ArrayList. O colecție (numită și container) este un obiect care grupează mai multe elemente într-o singura unitate. Colecțiile sunt folosite la stocarea, regăsirea, manipularea și comunicarea datelor agregate. Colecţiile sunt mult mai uşor de folosit pentru programare de uz general, în special atunci când datele sunt foarte volatile – multe adăugări, ştergeri şi modificări directe în timp, precum in cazul polinoamelor (adaugare monom).

**Proiectare clase:**

Pentru realizarea proiectului, am ales implementarea a 6 clase: Monom, Polinom, Operatii, Panel, Controller, TestOp..

Clasa „Monom” este o clasa ce contine o variabila instanta pentru coeficientul unui monom, una pentru gradul acestuia si un format pentru afisarea cu mai multe zecimale dupa virgula in cazul coeficientului, fiind vizibile doar in clasa respectiva (private). Clasa are un constructor definit si unul fara parametri. De asemenea, in clasa se definesc metode publice de setare si returnare a coeficientului/gradului si metoda „toString” folosita pentru afisarea unui monom (de exemplu: „ 4 x ^ 2 „ ).

In clasa „Polinom” am definit o coletie de monoame (ArrayList<Monom>) ca si variabila instanta de tip private a clasei, metodele de setare si returnare a monoamelor din ArrayList si metoda „toString” pentru afisarea unui polinom.Tot aici am definit si metoata de transformare a stringului introdus in interfata prin intermediul RegEX.

In clasa „Operatii” am definit metodele pentru efectuarea operatiilor: adunare, scadere, inmultire, derivare, integrare, de verificare (daca exista doua monoame cu acelasi grad intr-un polinom sa le adune si sa stearga), si de ordonare descrescatoare.

In clasele „Panel” si „Controller” am definit interfata grafica cu utilizatorul (GUI), folosind controalele din Jpanel-ul GUI SWING. Controller contine si metoda main a programului.

Clasa „testOp” este o testare unitara a programului, folosind JUnit, in care s-a verificat validitatea operatiilor din clasa „Operatii”, si parsarea.

**Algoritmi:**

**OrdDescresc():** metoda pentru ordonoare descrescatoare a polinoamelor, in functie de grad. Pentru acest algoritm de sortare am folosit metoda bulelor („bubble sort”). Am declarat o variabila „k” pentru validare. Parcurgem colectia si comparam doua cate doua monoame, astfel: daca gradul monomului de la cheia „i+1” este mai mare decat gradul monomului de la cheia „i”, facem o interschimbare prin declararea unui monom auxiliar. Daca a avut loc o interschimbare „k” primeste 1, algoritmul repetandu-se cat timp acest „k” e valid (1).

**Verifica():** metoda pentru verificare polinoamelor. Am declarat o variabila „n”, in care se salveaza dimensiunea polinomului dat ca parametru. Parcurgem cu 2 chei (i si j) ArrayList-ul, astfel ca in momentul gasirii a doua monoame cu acelasi grad, coeficientului monomului de la cheia „i” i se aduna coeficientul monomului de la cheia „j”, se sterge monomul de la cheia „j”, iar dimensiunea colectiei si pozitia indicatorului „j” scad cu 1.

**Adunare():** metoda de adunare este o metoda ce returneaza un polinom si primeste ca parametrii doua polinoame. In interiorul metodei declaram un polinom suma pentru salvarea rezultatului adunarii. Parcurgem cele doua polinoame cu un for each, astfel ca pentru fiecare monom din polinom, se creeaza un monom auxiliar ce primeste coeficientul si gradului monomului curent, ulterior fiind agaugat la polinomul suma. Dupa ce a fost adaugat fiecare monom din cele 2 polinoame, parcurgem cu 2 chei (i si j)polinomul suma, astfel ca in momentul gasirii a doua monoame cu acelasi grad, coeficientului monomului de la cheia „i” i se aduna coeficientul monomului de la cheia „j”, se sterge monomul de la cheia „j”, iar dimensiunea colectiei si pozitia indicatorului „j” scad cu 1.

**Scade():** metoda pentru scaderea a doua polinoame, primeste ca parametrii doua polinoame. Am parcurs cele 2 polinoame cu un for each, adaugand fiecare monom intr-un polinom rezultat „scadere”. Deoarece e vorba de o scadere, am decis ca moanele din al doilea polinom(care se scade) sa fie adaugate cu semnul minus. Dupa ce a fost adaugat fiecare monom din cele 2 polinoame, parcurgem cu 2 chei (i si j)polinomul suma, astfel ca in momentul gasirii a doua monoame cu acelasi grad, coeficientului monomului de la cheia „i” i se aduna coeficientul monomului de la cheia „j”, se sterge monomul de la cheia „j”, iar dimensiunea colectiei si pozitia indicatorului „j” scad cu 1.

**inmultire():** metoda de inmultire a doua polinoame, primeste ca parametrii doua polinoame. Parcurgem cele doua polinoame, inmultind fiecare monom din primul polinom cu fiecare monom din cel de-al doilea polinom si adaugand rezultatul intr-un polinom „inmultire”.

**derivare():** metoda pentru derivarea unnui polinom, ce returneaza un polinom si are ca parametru un polinom. Pentru fiecare monom dintr-un polinom, declaram un monom auxiliar, al carui coeficient primeste coeficientului inmultit cu gradului monomului curent, iar gradul devine gradul curent minus 1. Fiecare monom „j” din polinomul derivat (parcurs cu un for each) va fi adaugat intr-un polinom rezultat „derivare”, ordonat

**integrare():**metoda pentru integrarea unnui polinom, ce returneaza un polinom si are ca parametru un polinom. Pentru fiecare monom dintr-un polinom, declaram un monom auxiliar, al carui coeficient primeste coeficientului impartit cu gradului monomului curent plus 1, iar gradul devine gradul curent plus 1. Fiecare monom „j” din polinomul derivat (parcurs cu un for each) va fi adaugat intr-un polinom rezultat „integrare”,

**transforma():** metoda statica pentru prelucrarea unui polinom introdus ca si String de la tastatura. Declaram un pattern (model) pentru care se face cautarea. Cat timp string-ul se potriveste cu acest pattern, declaram un coeficient in care salvam primul grup din pattern si un gard in care salvam al doilea grup din pattern. Se creeaza un Monom auxiliar format din coeficientul si gradul declarate anterior, ce va fi adaugat intr-o colectie.

IMPLEMENTARE SI TESTARE

La rularea programului, utilizatorul ii va fi vizibila o interfata grafica prin care acesta poate comanda operatiile si poate introduce polinoamele si valorile dorite.

Reguli de utilizare a aplicatiei :

**1.** Fiecare polinom se va scrie intr-o casuta destinata lui.

**2.** Fiecare coeficient va fi mentionat (inclusiv 0 si1).

**3.** Derivarea si integrarea se vor realiza pentru fiecare polinom in parte.

**4.** Polinoamele trebuie introduse sub forma: a1x^a1+/-b1x^b1+/-c1x^c2+/-…

**Implementarea catorva algoritmi**:

**public** **Polinom** **transforma**(**String** string)

{

**Polinom** **polinom** = **new** Polinom();

//am creat un pattern pentru a identifica monoame in Stringul introdus ca parametru

//(-?\\b\\d+) identifica coeficientu si semnul acestuia = group(1)

//[xX]\\^ identifica daca intre coeficient si grad este x^ sau X^

//(-?\\d+\\b) identifica gradu = group(2)

**String** **pattern** ="(-?\\b\\d+)[xX]\\^(-?\\d+\\b)";

**Pattern** **p** = **Pattern**.*compile*(pattern);

**Matcher** **m** =p.matcher(string);

**while**(m.find())

//daca am gasit un monom sub forma de strig construim un nou monom

//convertim gradul respectiv coeficientul din string in int respectiv double

{**int** **gradd**=**Integer**.*parseInt*(m.group(2));

**double** **coeff**= **Double**.*parseDouble*(m.group(1));

**Monom** **monom** = **new** Monom(gradd,coeff);

// si is adaugam intr-un Polinom

polinom.getPolinom().add(monom);

}

**return** polinom;

}

**public** **void** **verifica**(**Polinom** p1) {

**int** **n** = p1.getPolinom().size();

**int** **i** = 0;

**while** (i < n - 1) {

**int** **j** = i + 1;

**while** (j < n) {

**int** **gradP1** = p1.getPolinom().get(i).getGrad();

**int** **gradP2** = p1.getPolinom().get(j).getGrad();

// daca doua monoame au gradele egale

**if** (gradP1 == gradP2) {

**double** **coeficientP1** = p1.getPolinom().get(i).getCoeficient();

**double** **coeficientP2** = p1.getPolinom().get(j).getCoeficient();

// adunam coeficientii in primul monom

//si stergem al doilea monom

p1.getPolinom().get(i).setCoeficient(coeficientP1 + coeficientP2);

p1.getPolinom().remove(j);

n--;

}

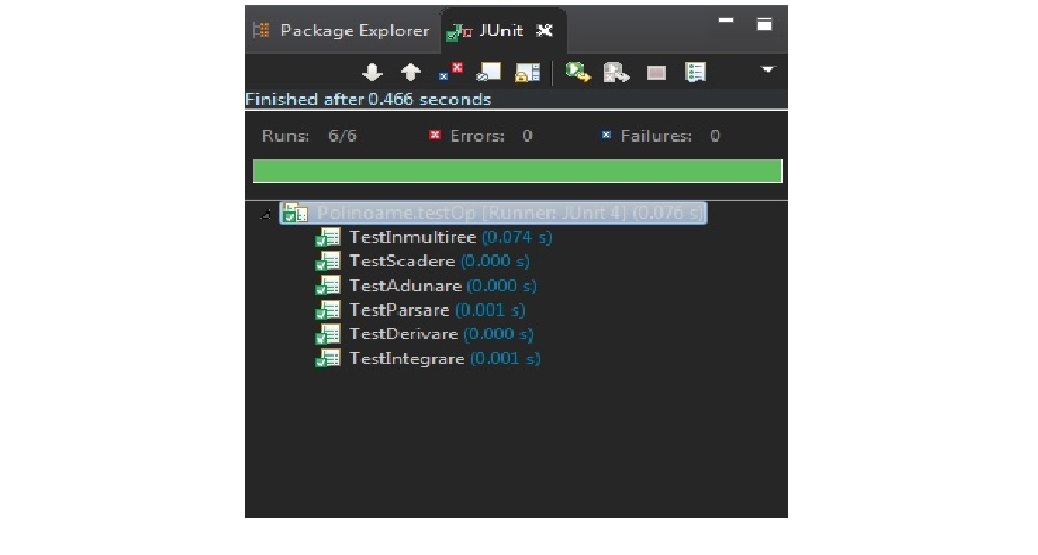
j ++;

}

i++;

REZULTATE

S-a obtinut o implementare a cerintei problemei care trateaza anumite operatii asupra polinoamelor: adunare, scadere, inmultire, derivare, integrare. Proiectul de asemenea ofera utilizatorului o interfata grafica usor de utilizat.

Rezultate testare unitara: 

Ca eventualele imbunatatiri ce pot fi aduse programului se mentioneaza:

--adaugarea unor operatii suplimentare ce pot fi exectuate asupra polinoamelor precum : impartirea polinoamelor, aflarea valorii intr-un punct etc.

--afisarea graficului unui polinom

--folosirea JavaFX pentru interfata grafica cu utilizatorul, in locul Swing-ului.

CONCLUZII

Prin prima tema, am aprofundat anumite concepte de programare orientata pe obiecte,precum colectiile (folosirea ArrayList), partea de Swing, vizibilitatea metodelor in clase si pachete, utilizarea Regex si de asemenea folosirea JUnit pentru testarea unitara.

}

BIBLIOGRAFIE

<http://users.utcluj.ro/~igiosan/teaching_poo.html>

<http://stackoverflow.com/questions/28859919/java-regex-separate-degree-coeff-of-polynomial>

<http://inf.ucv.ro/~mihaiug/courses/mps/labs/junit/>