# Sistem de procesare a polinoamelor

## 1.Obiectivul temei

Tema aceasta consta in crearea unui system de calcul a polinoamelor de o singura varriabila. Nu mentrionez “cu coeficienti intregi” pentru ca proiectul functioneaza la fel de bine si cu coeficienti reali. Acesta fiind obiectivul principal.

In ceea ce priveste obiectivele secundare,  ne axam strict pe operatiile necesare pentru a indeplini obiectivul mentionat mai sus. Vorbim despre operatiile de adunare, scadere, inmultire, impartire, integrare si derivare a polinoamelor. Astfel, in urmatoarele pagini o sa va descriu, pe de o parte, cu lux de amanunte, cum functioneaza aplicatia creata si cum o puteti folosi, iar pe de alta parte,  o sa incerc sa va fac sa intelegeti cum am gandit eu proiectul, ce metodele, algoritmi si modalitatile de a stoca un polinom am ales pentru ca totul sa fie la un nivel acceptabil, din punct de verede al optimizarii.

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

Cum am analizat eu problema? Raspunsul e simplu: am plecat de la ceea ce trebuie sa fie capabila aplicatia mea sa faca. Astfel am inceput in felul urmator : utilizatorul va introduce un polinom sau doua intr-o casuta Text si programul v-a trebui sa-i furnizeze rezultatul. Cum se va furniza rezultatul ? Prin apasarea unui buton corezpunzator operatiei dorite. Asa am dedus ca interfata va trebui sa arata cam asa :


Destul de simplu pana aici.

Modeland problema, ne gandim la structura unui polinom. Putem zice ca el este impartit in monoame. Evident, un monom retine un coeficient si un exponent. Cerinta problemei ne spune ca ea trebuie realizata pentru coeficienti intregi, dar cand am ajuns la partea de integrare mi-am dat seama ca ar fi mai convenabil ca acestia sa fie totusi reali. Prin asta am adus un bonus problemei.

Acum sa vorbim despre scenariile si cazurile de utilizare. Aceasta aplicatie poate recunoaste orice polinom de o singura variabila. De preferabil structura

unui polinom sa fie ca in urmatorul exemplu : « -3*x^2 +5*x -7 » dar o sa va fac o lista cu cazurile mai deosebite, acceptate, pentru ca polinomul dumneavoastra sa fie recunoscut :

- Puteti introduce cate spatii vreti, indiferent intre cine si cine (de exemplu intre monoame, sau intre coeficientul unui monom si x etc.)
- Puteti folosi sau nu operatorul * (inmultire) sau ^ (ridicare la putere). Spun asta deoarece polinomul « -3*x^2 » va fi recunoscut indiferent daca il introduceti in acest mod sau ca il introduceti ca fiind « -3x^2 » sau « -3*x2 » sau pur si simplu « -3x2 ».
- De asemenea coeficientii monoamelor pot fi si reali « -3.1*x^2 » , dar exponentul nu !
- Nu se fac diferente intre X si x.

Alte caractere introduse de dumneavoastra, in afara de cifre, '.' (pentru numere reale), '*' , '^', '-'(pentru numere negative) si 'x' sau 'X' nu vor fi recunoscute.

Use-case : nici nu stiu daca are rost sa vorbesc despre asta. Pur si simplu utilizatorul introduce un polinom sau doua (in functie de operatia dorita) , apasa un butonul specific operatiei si boom : i se va furniza rezultatul in cazul in care polinoamele introduse respecta sintaxa mentionata mai sus. Tin de asemenea sa va atrag atentiei asupra faptului ca operatiile de adunare, scadere, inmultire, impartire se efectueaza asupra a doua polinoame, iar integrarea si derivarea vor fi aplicate doar Polinomului 1 .

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

In urmatoarele randuri o sa va prezint cum am ales sa proiectez sistemul de calcul pentru polinoame.

Vreau sa incep prin a mentiona clasele care practic « stocheaza » informatia si anume : Monom si Polinom . Prima dintre acestea contine un coeficient si un exponent. A doua contine un Hashtable de Monoame. Deci relatia dintre aceste clasa este una de tip « contine ». Am ales sa folosesc Hashtable deoarece este cel mai eficient cand vine vorba de a cauta, introduce sau sterge un monom dintr-un polinom deoarece am ales cheia ca fiind chiar exponentul. O problema generata de acest tip de Colectie a fost la impartire unde mi-a fost mai usor sa transpun Hashtable-ul de Monoame intr-un ArrayList ordonat descrescator dupa valoarea exponentului. Aceste doua clase se afla in pachetul « Model » (stiu…numele nu este prea sugestiv, o sa vedeti pe parcurs).

Al doilea lucru despre care vreau sa vorbesc este pachetul « Operatii ». El contine interfata « Operatie » ce este implementata de urmatoarele clase : « Adunare », « Scadere », « Inmultire », « Impartire », « Integrare » si « Derivare ». Acestea sunt clase SINGLETON pentru ca am nevoie de ele doar ca sa efectuez cate o operatie pe un obiect (sau doua) instanta a unui polinom si o sa imi furnizeze un rezultat de tip Polinom. Astfel, la un mument dat de timp o sa am un singur rezultat. Metoda de baza « calculeaza » primeste un parametru VARARGS asta insemnand ca numarul de argumente poate sa difere. Regula in sine dupa care se ghideaza este ca tot timpul va primi un sir de argumente. In cazul acestei aplicatii voi trimite ca argument un polinom sau doua, in functie de tipul operatiei (deoarece

adunarea, scaderea, inmultirea, impartirea necesita doua polinoame, iar integrarea si derivarea se apica unui singur polinom). Tin sa multumesc profesoarei de la laborator pentru ideile date (de la ea am pornit cu ideea de a folosi clase singleton pentru fiecare operatie si de asemenea metoda calculeaza sa aiba argument de tip varargs.

Urmatorul pachet despre care o sa vorbesc este denumit « Parsare ». El contine clasele « Expresie » si « ExpresieInvalidaException ». Prima dintre ele este tot o clasa singleton si are ca metoda de baza getPolinom. Metoda primeste ca argument un string si va trebui sa returneze un obiect de tip Polinom. Clasa « ExceptieInvalidaException » mosteneste clasa « Exceptie ».

Pachetul « UserInterface » contine clasele « Model » , « View » si « Controller ». Astfel este urmarit a se implementa patternul MVC ce ofera o foarte buna structurare a proiectului in majoritatea cazurilor. El presupune a se separa Vederea (interfata grafica ce comunica direct cu utilizatorul) de Model, ce e considerat creierul aplicatiei. El face toate operatiile si furnizeaza rezultate pentru vedere. Pentru a oferi o mai buna intelegere a acestui pattern doresc sa citez un fost profesor a corui nume nu o sa-l mentionez aici : « Modelul e de fapt aplicatia ta. Il poti folosi direct in consola. Dac ape urma doresti sa-i faci si si o interfata grafica care sa comunice cu utilizatorul e alegerea ta. » . Consider ca aici se evidentiaza foarte bine diferentele dintre Model si Vedere. Controllerul e folosit pentru a conecta aceste doua parti prin adaugarea de ascultatori partilor active, in cazul nostru celor 6 butoane. Deci el face legatura dintre Model si View.

Ultimul pachet, «Testare », contine o clasa denumita « Teste » in care, prin importarea motodelor assertTrue si assertFalse ale pachetului JUnit se testeaza fiecare metoda in parte. Astfel aceste motode de calcul ale polinoamelor sunt supuse atat la teste pozitive cat si la teste negative.

Diagrama UML a proiectului meu arata ca in figura 1. Chiard aca este o diagrama destul de simplificata, relatiile de baza se pot distinge usor. De exemplu, relatia implementare a unei interfete este definit printr-o linie punctata cu o sageata plina in varf (vezi clasele Adunare, Scadere, Inmultire, Impartire, Derivare si Integrare ce implementeaza interfata Operatie unde prin acest lucru ele sunt obligate sa ofere un corp metodei calculeaza). Intre Monom si Polinom se vede o relatie de tip « contine ». Daca o duc putin in extrema ar fi chiar o relatie de agregare. Doresc sa lamuresc acest lucru prin urmatoare intrebare : « Poate un polinom sa existe fara nici un monom ? ». Nu stiu care ar fi raspunsul vostru dar mie mi se pare absolut inutil. De aceea metoda CONSTRUCTORUL clase Monom este privat, el putand fi creat printr-o METODA DE FABRICARE. Aceasta metoda pune conditia ca monomul ce va fi creat sa aiba coeficientul diferit de 0. Din acest lucru rezulta faptul ca nu vom putea crea un monom cu coeficientul 0.

Relatiile dintre Controller, View si Model sunt urmatoarele : Modelul nu stie de Vedere si nici invers. Controlorul agrega o Vedere si un Model. Fara acestea controlorul nu ar avea nici un scop.

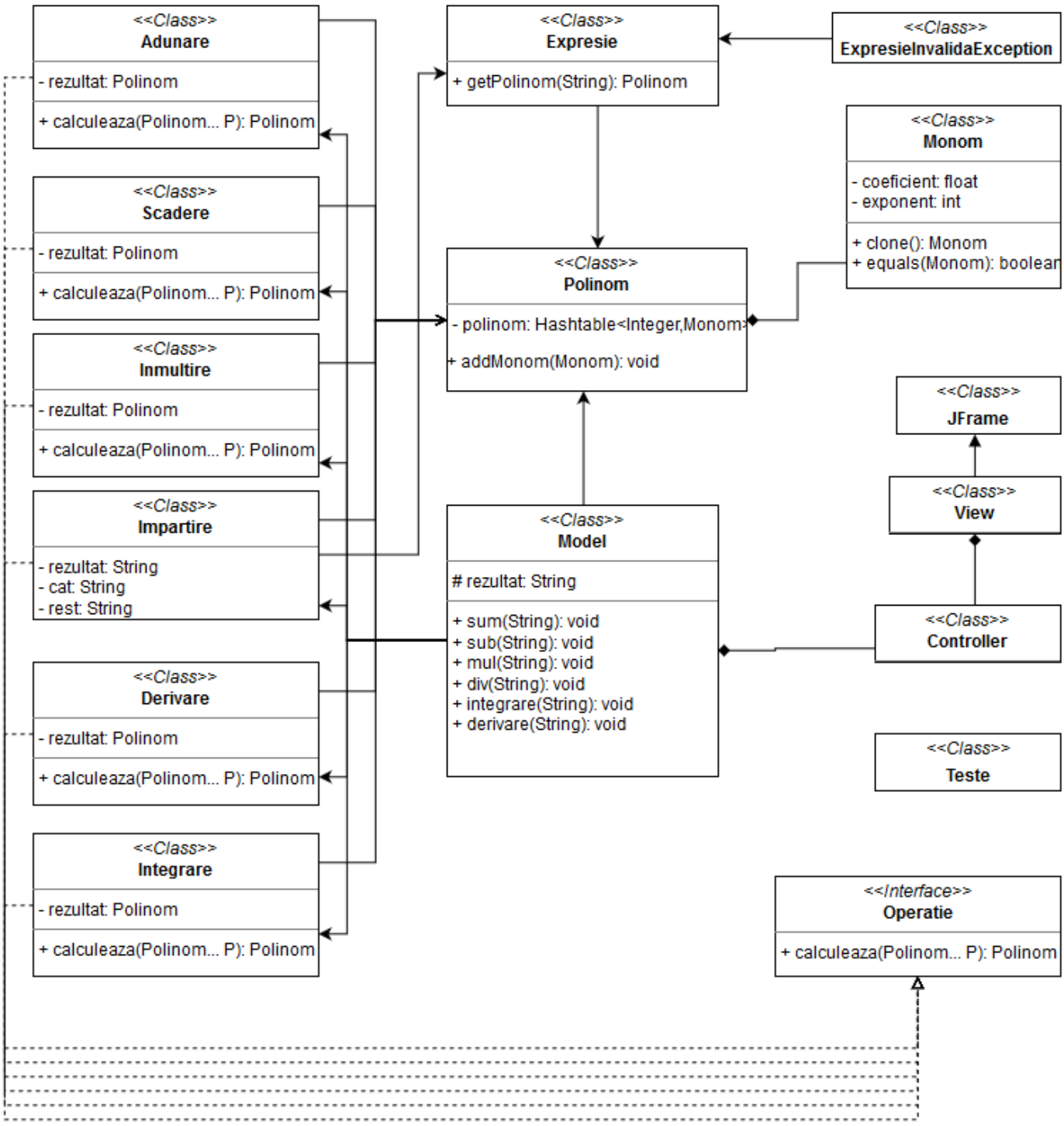


Figura 1

## 4.Implementare

Acum vreau sa va vorbesc putin despre modalitatea in care am parsat un string, transformandu-l intr-un obiect instanta a unei clase Polinom. Cred ca partea asta mi-a ocupat cel mai multi timp (in afara de intocmirea documentatiei, bineinteles). M-am bazat pe faptul ca daca o sa am o parsare buna, care imi acopera cat mai multe cazuri si imi furnizeaza polinoame valide, partea cu operatiile o sa fie floare la

ureche. Astfel ca m-am gandit cum sa scap de toate caracterele inutile si mi-am conceput eu un model ideal de monom. Acesta este in felul urmator : coeficient x exponent (ex : 2x3 sau -2x3). Deci practic fara nici un caracter de tipul * sau ^. Asa am ajuns la concluzia ca inainte sa incep parsareaa ar trebui sa elemin toate caracterele inutile si mai ales spatiile. Pe urma am observat ca delimitatorii dintre monoame pot fi doar caracterele '+'si '-'. Acum urmeaza partea in care fac split stringului initial dupa aceste doua caractere. Rezultatul va fi un sir de stringuri ce va reprezenta fiecare individual cate un monom. Deci dupa ce am un string ce reprezinta de fapt un monom imi va fi foarte usor sa extrag coeficientul si exponentul. Puri si simplu mai fac un split dupa caracterul x si astfel voi afla structura monomului. Desigur ca exista unele exceptii pe care le-am tratat individual, precum monoamele ce au exponentul 0 (acestea nu au x) si cele cu exponentul 1 (acestea nu o sa aiva exponent, si va trebui sa stabilesc eu exponentul ca fiind 1).

Pentru operatiile de baza pur si simplu parcurg cele doua polinoame si efectuez operatia specifica pe monoamele individuale.

Partea mai grea a fost la impartire, dar am biruit si acolo . Practic o sa incerc sa tot impart deimpartitul la impartitor si sa actualizez restul si catul oana cand gradul restului este mai mic decat gradul catului.

Interfata grafica este una foarte simpla, nu am mai stat sa pierd vremea cu ea doarece in principal partea de frontend al proiectelor mai mari nu se realizeaza in Java. Asadar interfata mea contine pur si simplu panoul de baza de tip ContentPane si alte 3 panouri incluse in parteaa de sus centru si jos. Parte de sus contine mesajul « Salut » pe care trebuia sa-l schimb cu ceva mai interesant, dar nu cred ca asta e ideea proiectului. Partea centrala contine trei Label-uri si trei TextField-uri corespunzatoare polinoamelor 1, 2 si rezultatului. In partea de jos apar butoanele specifice fiecarei operatii.

## 5.Rezultate

In ceea ce priveste rezultatul fiecarei operatii, acestea le-am surprins in cazuri destul de complece in clase de Testare ce foloseste JUnit. Sunt create metode de test pentru fiecare operatie in parte, inclusiv pentru parsare. In plus pentru fiecare dintre acestea am facut si teste positive, dar si negative. De asemenea am incercat sa ofer date de intrare destul de complexe pentru a analiza rezultatul, de fiecare data acesta fiind favorabil asteptarilor mele.

## 6.Concluzii

Avand in vedere ca nu prea mai am ce sa scriu in documentatie pentru a acumula numarul de cuvinte cerute, o sa creez o concluzie destul de stufoasa (sper ca doamna profesoara sa nu citeasca asta). Pe parcursul proiectarii si implementarii aceste aplicatii am acumulat informatii noi, si anume mi-am creat o viziune de ansambul asupra claselor singleton si am invatat cum sa utilizez varargs pentru metode pentru ca inainte nici macar nu auzisem de acest concept. Proiectul nu a fost foarte greu de realizat, dar a necestitat ceva timp pentru a-l implementa si mai ales pentru a trata cazurile exceptionale pentru ca la fiecare pas se ivea cate unul nou.

# 7.Bibliografie

www.google.ro

www.youtube.com

www.draw.io

https://www.mkyong.com/tutorials/junit-tutorials/

Cursurile de POO din semestrul trecut

Cateva concept despre realizarea diagramelor UML din cursul de TP din semestrul current.