

FACULTATEA: Automatică și Calculatoare

SPECIALIZAREA: Calculatoare și Tehnologia Informației

DISCIPLINA: Tehnici de programare fundamentală

PROIECT: Calculator de polinoame

**Buțiu Cristian**

**An universitar:2020-2021**

**Grupa: 30227**

**Profesor laborator: Dan Mitrea**

**Cuprins:**

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

Obiectivul principal al temei este de a implementa un calculator de polinoame care să poată efectua operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare.

* Folosirea conceptelor de bază a Programării Orientate pe Obiect(moștenire, încapsulare, etc.).
* Crearea calculatorului de polinoame folosind modelul conceptual logic MVC(Model-View-Controller).
* Folosirea design-ului Programării Orientate pe Obiect – definirea apropiată

a claselor Polinom si Monom ca rezultat al descompunerii problemei.

* Folosirea expresiilor Regex pentru extragerea coeficientilor si puterilor polinomului.
* Folosirea standardului Java de numire a pachetelor, claselor, metodelor, atributelor.
* Folosirea JUnit pentru testarea funcțiilor care realizează operațiile matematice.

1. **Analiza problemei**

Analiza unei probleme reprezintă cel mai important pas către rezolvarea acesteia. Acest pas presupune o examinare amănunțiă și o identificare a unei soluții optime pentru problema dată. Ideiile la nivel conceptual trebuie transformate într-o formă finală dorită, adica codul Java.Primul pas este identificarea pachetelor si claselor care vor fi folosite pentru aplicație, iar ulterior furnizarea de metode potrivite pentru fiecare dintre aceste clase.

Modelarea datelor este și ea un aspect important pe care nu trebuie să îl omitem.În cazul de față avem 2 posibilități prin care utilizatorul poate introduce date.În primul caz se introduce doar primul polinom pentru operațiile de derivare și integrare, în cel de-al doilea caz utilizatorul introduce ambele polinoame care vor fi folosite pentru operațiile de: adunare, scădere, înmulțire, împărțire.

Deoarece utilizatorul furnizeaza date de intrare de tipul String, dar în aplicație se va lucra cu obiecte de tip Polinom și Monom, vom avea nevoie de metode precum stringToPolinom() care transforma String-ul primit ca si parametru intr-un polinom și polinomToString() care transforma polinomul intr-un String. Metoda toString() va fi suprascrisa si in clasa Monom si in clasa Polinom.

1. **Proiectare**

Pentru ușurarea construirii polinoamelor, acestea trebuie introduse sub forma ax^n + bx^(n-1) + ... + cx^1 + dx^0(chiar dacă exponentul monomului este 1 sau 0 și chiar dacă coeficientul unui monom este 1).

Modelul conceptual de proiectare este MVC(Model-View-Controller), așadar clasele sunt împărțite în 4 pachete:primul pachet este Model care are clasele Polinom si Monom,al doilea View care are clasele View si Main si al treilea Controller care are clasa Controller, iar al 4-lea pachet este Test unde se află o singură clasă, cea de testare cu JUnit.

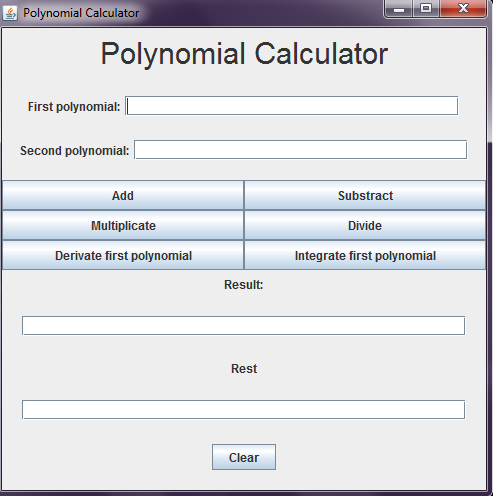
Clasa monom are 2 atribute, primul este coeficientul monomului de tip double, iar al doilea puterea acestuia de tip int. Am ales tip-ul coeficientului double deoarece, deși monoamele au coeficienți întregi, de cele mai multe ori operațiile de integrare și împărțire vor scoate un rezultat care are coeficienți reali. Constructorul clasei Monom

are la rândul său 2 argumente, coeficientul și puterea. Clasa Polinom are un atribut, care este o lista de monoame si un constructor vid. Încapsularea stă la baza acestui proiect, pentru accesarea atributelor dintr-un pachet extern am construit gettere si settere.

Operațiile matematice sunt implementate în clasa Controller. Metodele folosite sunt:

inmultireMonom, impartireMonom, stringToPolinom, adunareSauScadere – unde se face adunarea sau scaderea, integrare, derivare, inmultire, impartire.

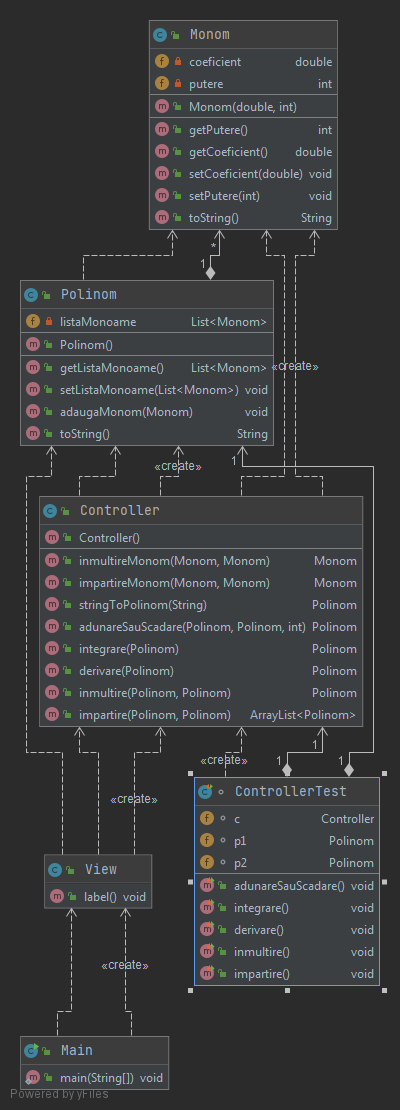
Interfața grafică a proiectului a fost implementată în clasa View din pachetul View.Design –ul interfeței a fost ales astfel încât orice utilizator al acesteia să înțeleagă cum se utilizează acest calculator. Aplicația are 4 textfield–uri. În primele două se introduc cele 2 polinoame, iar următoarele două reprezintă rezultatul operației alese, respectiv restul operației pentru operația de împărțire. Între aceste două textfield-uri se află 6 butoane pentru cele 6 operații care pot fi alese de utilizator. La finalul ferestrei se află un buton de „Clear” care permite ștergea informația din toate texfield-urile.Mai jos este un screenshot cu interfața grafică a proiectului.



Cel mai complex algoritm folosit în această aplicație este cel de împărțire. Acest algoritm folosește și metoda împărțireMonom, dar și algoritmul de scădere. Pentru acest algoritm am presupus ca primul polinom introdus va avea intotdeauna gradul mai mare decat cel de al doilea, și că polinoamele vor fi introduse cu gradul fiecarui monom in ordine descrescatoare. Voi explica algoritmul de împărțire a două polinoame.

1. Impartim primul monom din primul polinom la primul monom din al doilea polinom.
2. Inmultim catul obtinut cu cel de al doilea polinom si scadem din primul polinom acest rezultat
3. Repetam pasul 1 considerand restul de la punctul 2 ca noul deimpartit pana cand gradul primului polinom este mai mic decat al polinomului 2.

Mai jos este diagrama UML a proiectului.

.

4. **Implementare**

* Clasa Monom

Această clasă are două atribute de tip private: double coeficient si int putere. Avem un contructor care primește ca argument coeficientul si puterea. Sunt implementate atat metodele de get si de set pentru cele două atribute, dar si metoda toString(), metodă suprascrisă. Această metodă transformă monomul intr-un String, astfel incat sa poata fi afisat in textfied-ul pentru rezultat. În această metodă trebuie să avem in vedere că dacă coeficientul este pozitiv, atunci trebuie să afisam un + inaintea coeficientului la afisare. Daca coeficientul este negativ afisam fara + inaintea coeficientului.

* Clasa Polinom

Denumita sugestiv, clasa Polinom are ca scop reprezentarea la nivel de cod Java a unui polinom. Aceasta clasa are un atribut, acesta este lista de monoame si este de tip private. Definirea lui a fost facuta astfel: private List<Monom> listaMonoame = new ArrayList<>(). Avem un constructor vid. De asemenea avem si metodele de getListaMonoame, de setListaMonoame si metoda adaugaMonom care permite adaugarea unui monom la o lista de monoame. Suprascrierea toString() nu putea sa lipseasca nici din aceasta clasa. Metoda transforma polinomul intr-un String pentru a putea fi afisat. Initial vom avea un String care este gol, urmand ca mai apoi sa adaugam pe rand monoame folosind functia toString din clasa Monom. Metoda nu are niciun parametru. In cadrul acestei metode luam pe rand toate monoamele din polinom, apelam functia toString din Monom pe fiecare Monom si concatenam la Stringul gol Stringul returnat din metoda toString din clasa Monom. La final returnam String-ul care a fost construit.

* Clasa Controller

Aceasta clasa include toate metodele care efectueaza operatii pe polinoame sau pe monoame.Aceasta clasa nu are niciun atribut. Metodele continute de aceasta clasa sunt:

1. inmultireMonom() – aceasta metoda primeste parametrii doua monoame si returneaza un monom care reprezinta produsul celor doua monoame. Vom folosi aceasta functie, care va fi apelata succesiv, in metoda principala, cea de inmultire().
2. ImpartireMonom – aceasta metoda primeste 2 monoame si returneaza un monom care reprezinta impartirea primului monom la cel de al doilea. Vom folosi aceasta functie in metoda principala impartire().
3. adunareSauScadere – aceasta metoda primeste 3 parametrii: primul polinom, al doilea polinom si o variabila de tip int denumita nr, care are valoarea 0 daca dorim sa adunam si 1 daca dorim sa scadem cele 2 polinoame. Am ales sa unesc aceste doua opearatii intr-o singura functie deoarece am observat o mare similaritate intre codurile celor doua operatii, dorind sa am cat mai putin cod duplicat. Am parcurs ambele polinoame in acelasi timp, pornind de la primul monom(am considerat ca gradul monoamelor in polinom este descrescator) din fiecare si am adaugat intr-un al treilea polinom monom-ul cu puterea mai mare. Am trecut la urmatorul monom in polinomul din care am pus in polinomul ce la final va fi returnat. În cazul in care polinoamele aveau aceeai putere doar adunam sau scadeam coeficientii(depinde de valoarea lui nr).
4. inmultire() – aceasta metoda primeste ca parametrii cele doua polinoame si returneaza un obiect de tipul Polinom care este rezultatul obtinut in urma inmultirii celor doua polinoame primite. Algoritmul functioneaza astfel: Parcurgem lista de monoame a primului polinom, monom cu monom, si la fiecare parcurgere parcurgem si toata lista celui de-al doilea polinom, monom cu monom, si inmultim monomul din primul polinom cu toate monoamele din cel de-al doilea, rezultatul fiecarei inmultiri fiind adaugat intr-un polinom intermediar. La iesirea din bucla interioara adunam polinomul intermediar la polinomul final care va fi returnat la finalul metodei.
5. derivare() – aceasta metoda primeste un singur parametru, primul polinom furnizat de utilizator. Aceasta metoda returneaza tot un polinom care reprezinta derivarea polinomului dat. Parcurgem lista de monoame din polinom si intr-un polinom intermediar adaugam un nou monom format. Coeficientul noului monom va fi = coeficientul monomului polinomului \* puterea monomului polinomului, iar puterea va fi putearea monomului polinomului – 1. La final vom returna polinomul intermediar.
6. integrare() – aceasta metoda primeste un singur parametru, tot primul polinom furnizat de catre utilizator. Aceasta metoda returneaza tot un obiect de tip polinom care reprezinta integrarea polinomului furnizat ca parametru. Metoda este operatia inversa a derivarii. Parcurgem intreaga lista de monoame al polinomului dat ca parametru, monom cu monom, si intr-un nou polinom final, adaugam monom-ul creat astfel: coeficient = coeficient / (exponent + 1), iar exponent = exponent + 1.
7. stringToPolinom()-aceasta metoda primeste ca parametru un String, acesta fiind de fapt input-ul utilizatorului. Presupunem ca utilizatorul furnizeaza un polinom care are forma: ax^n+bx^(n – 1)+...+cx^1 + dx^0, folosim un algoritm de tip pattern / matcher si metoda split(). Astfel, extragem String-uri partiale care reprezinta String-ul unui monom. Ulterior, împărțim monomul în două String-uri, folosind regex. Prima parte reprezentând coeficientul, iar a doua parte exponentul. Deoarece impartim string-ul dupa caracterul ‘^’ in primul String ni se va lua si caracterul ‘x’ asa ca vom taia ultimul caracter al sau. Cele doua String-uri trebuie transformate in double, respective in int. Pentru acest lucru folosim functia parseDouble(), respective parseInt(), prima urmand a fi aplicata pe coefficient, iar a doua pe putere. Adaugam aceste 2 valori intr-un Monom care va fi adaugat la polinomul final prin metoda adaugaMonom. Cand s-a terminat de split-uit intreg String-ul dat de utilizator, inseamna ca am ajuns la final si returnam polinomul construit. În cazul in care utilizatorul nu furnizeaza un polinom de forma: ax^n+bx^(n-1)+...+cx^1+dx^0, vom afisa un mesaj de eroare “Introduceti polinoame valide.”. Aceasta verificare am facut-o in cadrul unei exceptii block try and catch.

* Clasa View

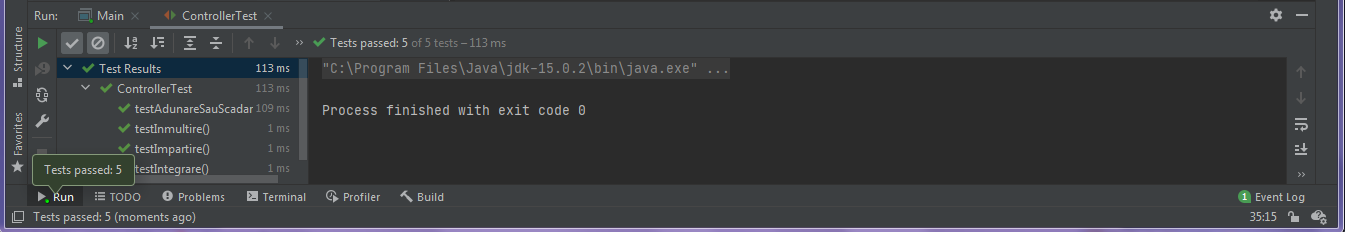
Aceasta clasa reprezinta intreaga interfata grafica, construita folosind Java Swing. Nu are niciun atribut, iar singura metoda continuta este metoda label in care avem declarate toate componentele. Componentele sunt conectate in asa fel incat sa obtinem un design simplu de inteles de catre orice utilizator. Interactiunea dintre interfata si codul aplicatiei este facut prin actionListener(), metode care sunt adaugate pe fiecare dintre cele 7 butoane si sunt declansate la momentul click-ului pe ecran al butoanelor respective. In cadrul acestei metode este specificată funcția matematică apelată pentru a ajunge la rezultatul dorit pe textfield-ul de afișare. Luarea valorii din inputuri se face prin metoda .getText(), iar punerea valorii in output se face prin metoda .setText().Clasa View are 4 textfield-uri: 2 pentru polinoamele furnizate de utilizator si 2 pentru rezultat si rest-ul impartirii.

* Clasa ControllerTest

Aceasta clasa are 3 atribute, primul este un obiect de tipul controller, iar celelalte sunt 2 polinoame care urmeaza a fi testate in fiecare dintre metodele de test. Avem metode de test pentru fiecare metoda matematica aplicata pe polinoame, dar testarea adunarii si scaderii se face in aceeasi metoda. Testarea a fost efectuata utilizand JUnit. Metodele continute de aceasta clasa sunt: testAdunareSauScadere(), testDerivare(), testIntegrare(), testInmultire(), testImpartire().

**5.Rezultate**

Mai jos avem testarea in IntelliJ cu JUnit. Toate testele date au fost validate.



**6.Concluzii**

Chiar daca am rezolvat tema cu 2 saptamani intarziere, consider ca am invatat multe din aceasta tema. Am realizat faptul ca pentru a aborda o problema ai nevoie de o analiza amanuntita si concreta. De-a lungul implementarii solutiei mele, am realizat ca pentru a avea o aplicatie cat mai aproape de perfectiune trebuie sa treci prin tot felul de probleme. Te confrunti cu cazuri speciale si erori pentru ca acest calculator sa functioneze pe orice date introduse de catre utilizator. Ca dezvoltari ulterioare, ar fi implementarea unui regex mai complex care sa ajute introducerea polinomului mai usor si a afisarii mai prietenoase cu utilizatorul. Daca coeficientul este 1 sa nu mai trebuiasca sa fie scris si daca exponentul este 1 sa nu trebuiasca sa se scrie x^1 sau daca exponentul este 0 sa nu se mai scrie x-ul.

**7. Bibliografie**

<https://www.jetbrains.com/help/idea/create-tests.html>

<https://plugins.jetbrains.com/plugin/15124-uml-generator>

<https://www.emathhelp.net/calculators/algebra-1/polynomial-calculator/?p=2x%5E4-2x%5E2%2B6x%5E1-5x%5E0&q=1x%5E2-1x%5E0>

<https://en.wikipedia.org/wiki/Polynomial>

<https://www.w3schools.com/java/java_regex.asp>

<https://stackoverflow.com/questions/23377044/java-arraylist-add-overwriting-values>