Tehnici de Programare – Tema 1

*Calculator de Polinoame*

Popovici Ruxandra

*Grupa 30227*

1. Obiectivul Temei

Obiectivul principal al proiectului de fata a fost proiectarea si implementarea unui calculator de polinoame ce foloseste o interfata grafica dedicata prin intermediul careia un utilizator va putea introduce de la tastatura polinoame, alege operatiile matematice pe care doreste sa le efectueze pe acestea si, intr-un final, va putea vizualiza rezultatul calculelor.

Obiectivele secundare ale proiectului:

* Analizarea obiectivelor proiectului si a problemelor legate de implementare prin modelarea de scenarii corespunzatoare cazurilor de utilizare, astfel incat sa identificam corect cerintele;
* Proiectarea calculatorului astfel incat toate necesitatile utilizatorului sa fie indeplinte;
* Implementarea solutiei optime descoperite prin scrierea de cod;
* Testarea calculatorului, observandu-i si analizandu-i concret comportamentul pentru mai multe cazuri, luand in considerare scenarii diferite de folosire si diferite date de intrare.

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

Prin analiza problemei intelegem examinarea cerintelor, potentialelor probleme si a diferitelor modalitati in care aplicatia noastra ar putea fi folosita, pusa la incercare: introducerea de date (a polinoamelor) care va duce la generarea unui raspuns corespunzator din partea calculatorului in functie de corectitudinea lor.

Dorim ca utilizatorul sa fie atentionat cu privire la validitatea structurilor introduse ca cei doi operanzi ai operatiilor ce ii sunt puse la dispozitie (structura adecvata a unui polinom, existenta a cel putin unuia dintre ei pentru efectuarea anumitor calcule, etc.).

De asemenea, programul trebuie sa functioneze in mod adecvat si sa genereze raspunsul dorit, urmand principiul diagramei urmatoare:

Rezultat

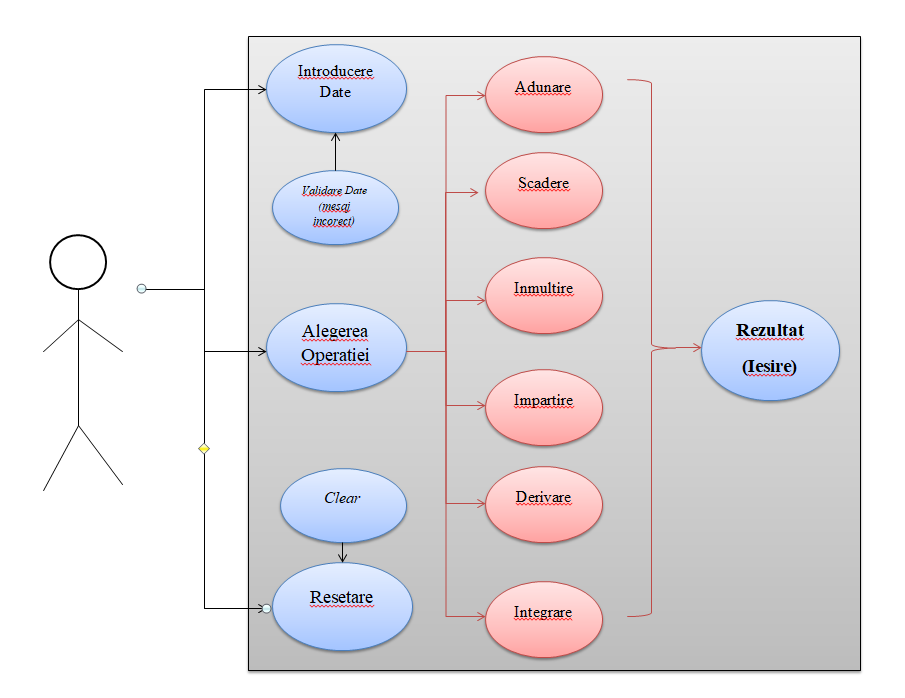
OPERATIE

Al Doilea Polinom

Primul Polinom

*Calculator de Polinoame*

*~Schema generala a proiectului~*

Diagrama Use Case demostreaza mai bine functionalitatea si ideea intregului proiect:

**User**

Dupa cum putem observa si mai sus, intregul sistem reprezinta aplicatia creata, in care User-ul reprezinta, de fapt, actorul / utilizatorul, care se asteapta sa primeasca un raspuns / rezultat dupa ce actioneaza asupra lui.

* Cerinte functionale ale calculatorului:
* Trebuie sa permita utilizatorului sa introduca date (cele doua polinoame);
* Trebuie sa permita utilizatorului sa selecteze operatia dorita si trebuie sa o si duca la implinire;
* Trebuie sa permita utilizatorului vizionarea rezultatului cerut – rezultatul operatiilor.
* Altele:
* Calculatorul trebuie sa fie intuitiv si usor de folosit de catre utilizator.
* Sa genereze un rezultat adecvat in caz de success sau sa afiseze un mesaj de eroare in caz ca datele primite nu au respectat cerintele si formatul cautat.
* Scenariu de utilizare:

1. Utilizatorul introduce primul polinom de la tastatura: ex. “x^3 + 2x^2 - 3”;
2. Utilizatorul introduce cel de-al doilea polinom de la tastatura: ex. “x^4 + 2x^3 – 8x^2”;
3. Utilizatorul apasa butonul de “Add” (“Adunare”), selectand operatia de adunare.
4. Calculatorul implementat efectueaza operatia de adunare si afiseaza rezultatul;
5. Utilizatorul iese din aplicatie.

Un scenariu alternativ de utilizare:

1. Utilizatorul introduce date gresite de la tastatura: ex. “afasfaasg”;
2. Aplicatia afiseaza un mesaj de eroare si sterge datele introduse;
3. Se revine la primii pasi din scenariul anterior.
4. Proiectare *(decizii de proiectare, diagrame UML, structuride date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)*

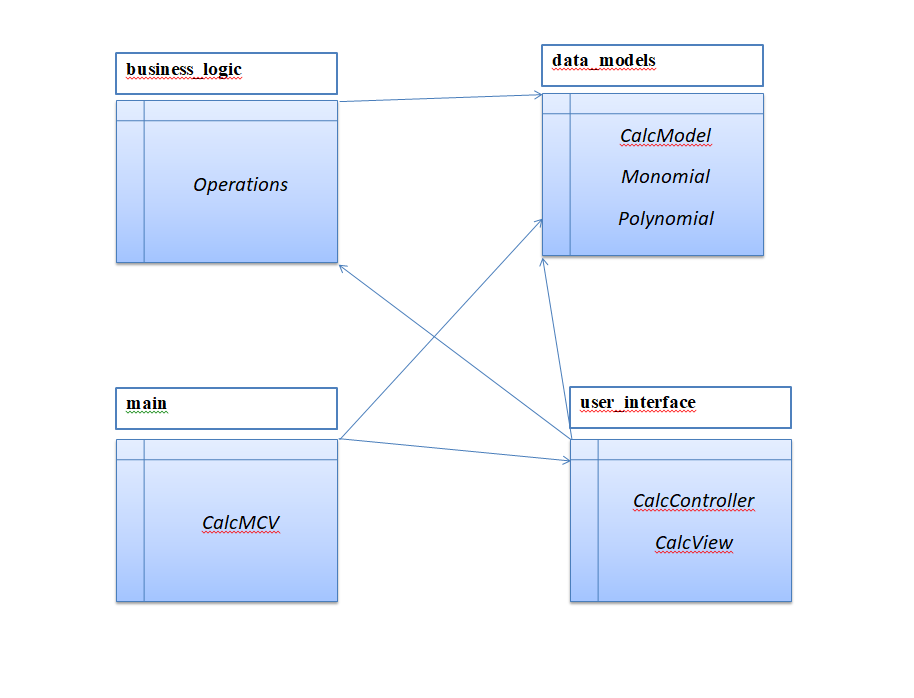
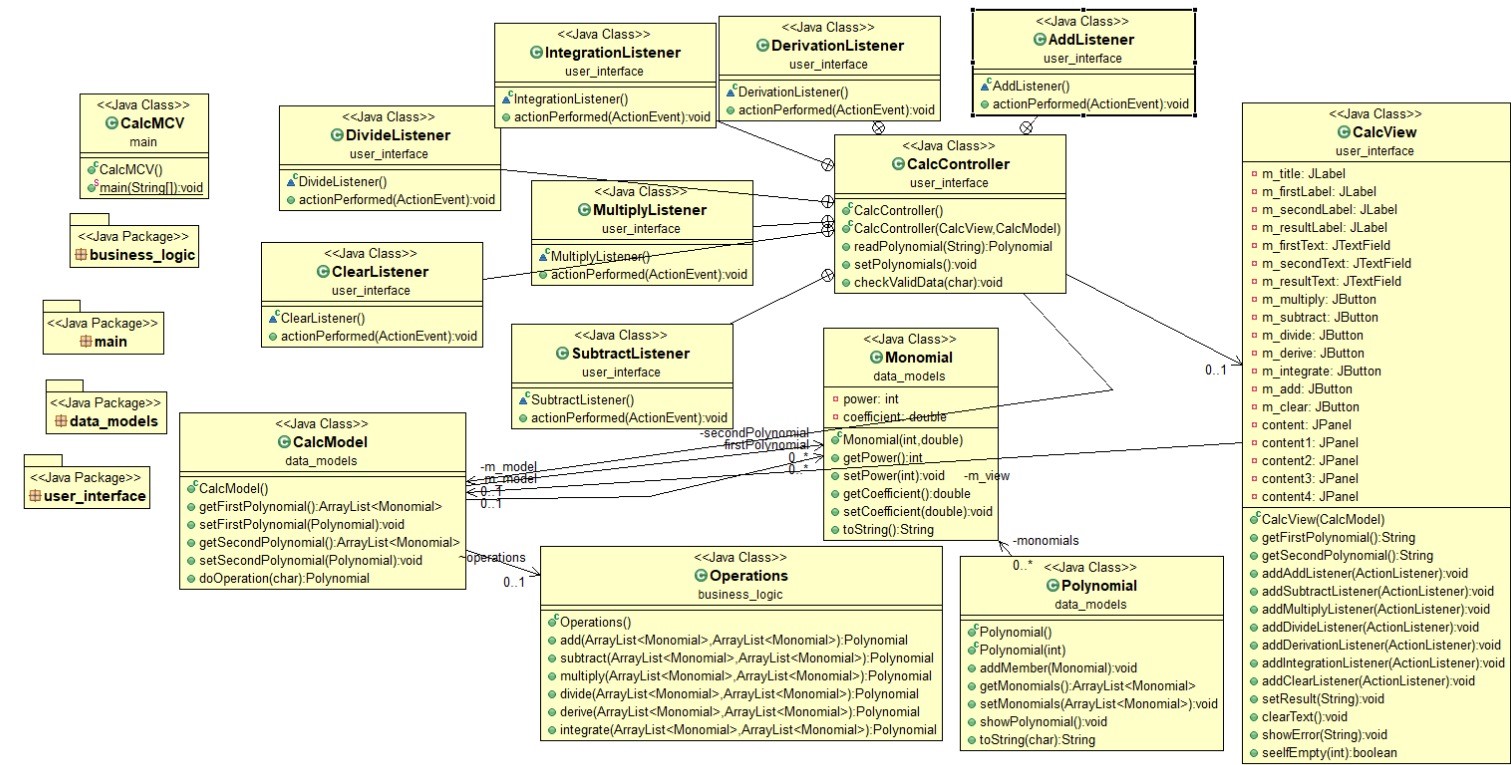
Proiectarea in limbaj OOP a presupus mai multi pasi, in functie de deciziile luate: am urmarit implementarea folosind modelul MCV (Model-View-Controller), fiind un model tip sistem interactiv si repartizarea lor in pachete, asa cum se poate observa si in diagrama de mai jos:

Diagrama UML a proiectului ajuta la o mai buna si amanuntita intelegere ai analiza a claselor, pachetelor, metodelor implementate in fiecare din acestea, cum se “leaga” si “comunica” intre ele:



Structura de tip MCV separa fiecare clasa in functie de functionalitatile metodelor din ele.

“data\_models” contine clase ce reprezinta “structurile” datelor la care se asteapta aplicatia: Monom, Polinom si doi operanzi de tip Polinom pentru efectuarea de operatii.

“user\_interface” le contine pe cele care au de a face cu interactiunea dintre utilizator si calculatorul / aplicatia propriu-zisa: GUI, JFrame, butoanele cu ActionListenere-le respective, etc..

“business\_logic” are ca si clasa chiar logica operatiilor ce au loc la cererea utilizatorului, implementarea acestora in functie de datele primite.

“main” reprezinta clasa care porneste aplicatia si care leaga toate celelalte pachete si clase impreuna.

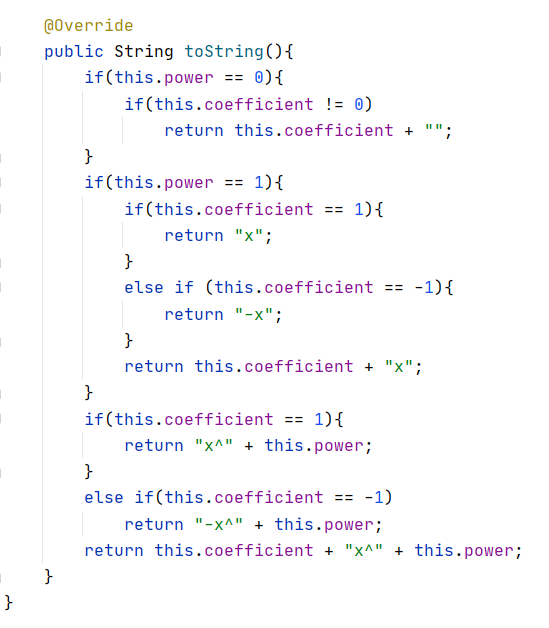
1. Implementare

* *Pachetul “data\_models”*

***Clasa “Monomial”***

Petntru o mai usoara interpretare a intregii structuri de tip “polinom” (al clasei “Polynomial”), am ales definirea unei structuri de tip monom de forma “ax^b”, unde “a” este coeficientul de tip double, “x” este variabila, iar “b” este exponentul / puterea.

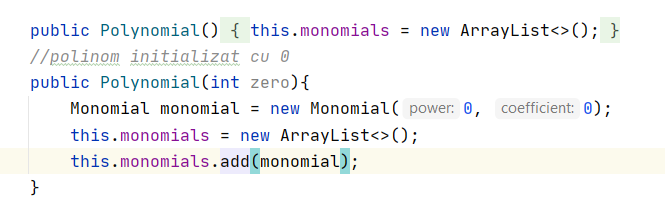
Inafara de constructor, getter si setter, am implementat, suprascriind metoda “toString”, o functie de afisare a monomului ce va ajuta la “estetica”: nu vom mai afisa in polinom termeni care nu exista (a = 0), coeficienti sau puteri care nu isi au sensul in contextul vizionarii rezultatului (a = 1, b = 0 sau b = 1).

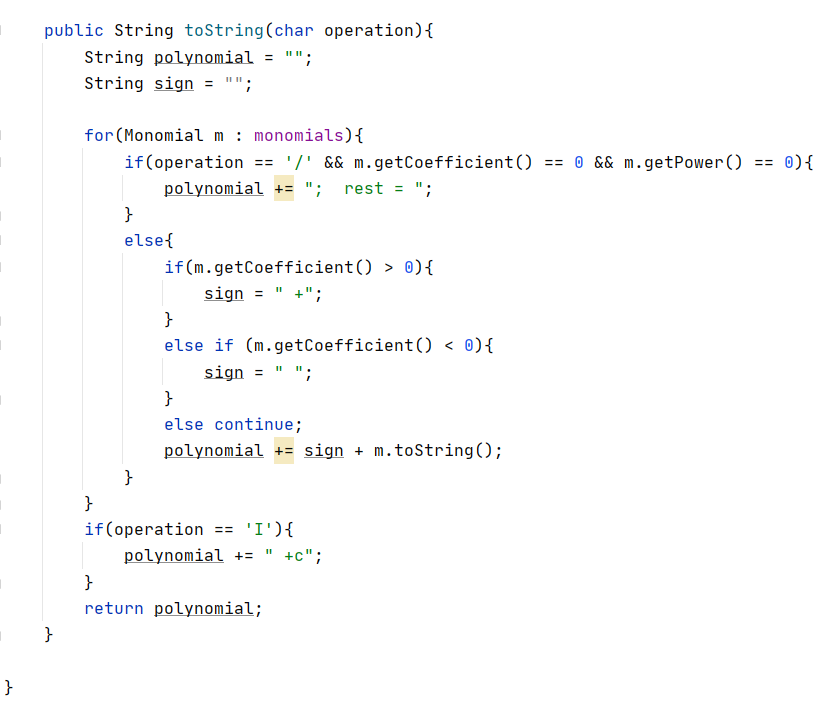


***Clasa “Polynomial”***

Aceasta implementeaza structura propriu-zisa pe care se bazeaza intreg proiectul: am ales reprezentarea polinomului sub forma unui ArrayList de monoame datorita usurintei cu care putem atat accesa termeni, cat si adauga, sterge, parcurge, etc.

Am ales implementarea a 2 constructori, printre care cel care primeste ca argument un “int zero” va fi folosit in a initializa un polinom cu un singur monom de tip “0x^0”, pentru a ajuta la calcule si pentru a evita eventuale erori.



Metoda “toString” din aceeasi clasa nu mai face suprascriere, astfel incat primeste ca parametru un “char operation”, care, in functie de operatia aleasa de utilizator, va formata rezultatul in mod corespunzator: pentru impartire, daca avem un termen de tip “0x^0”, il va inlocui cu “rest =” (mai multe la metoda “divide”), pentru integrare, va adauga un “+c” la final, iar in functie de coeficient, va adauga ori un “+”, ori un spatiu liber(“-” ca exista deja de la variabila “double coefficient” a monomului).

***Clasa “CalcModel”***

Contine cei doi operanzi primiti de la utilizator cu care se vor face calculele si un obiect de tip Operations care le va efectua.

Metoda “doOperation” primeste ca parametru un “char operation”, folosindu-l intr-un switch care va trasmite pe “fisrtPolynomial” si “secondPolynomial” metodelor corespunsatoare operatiei. Va returna un nou polinom (obiect de tip Polynomial).

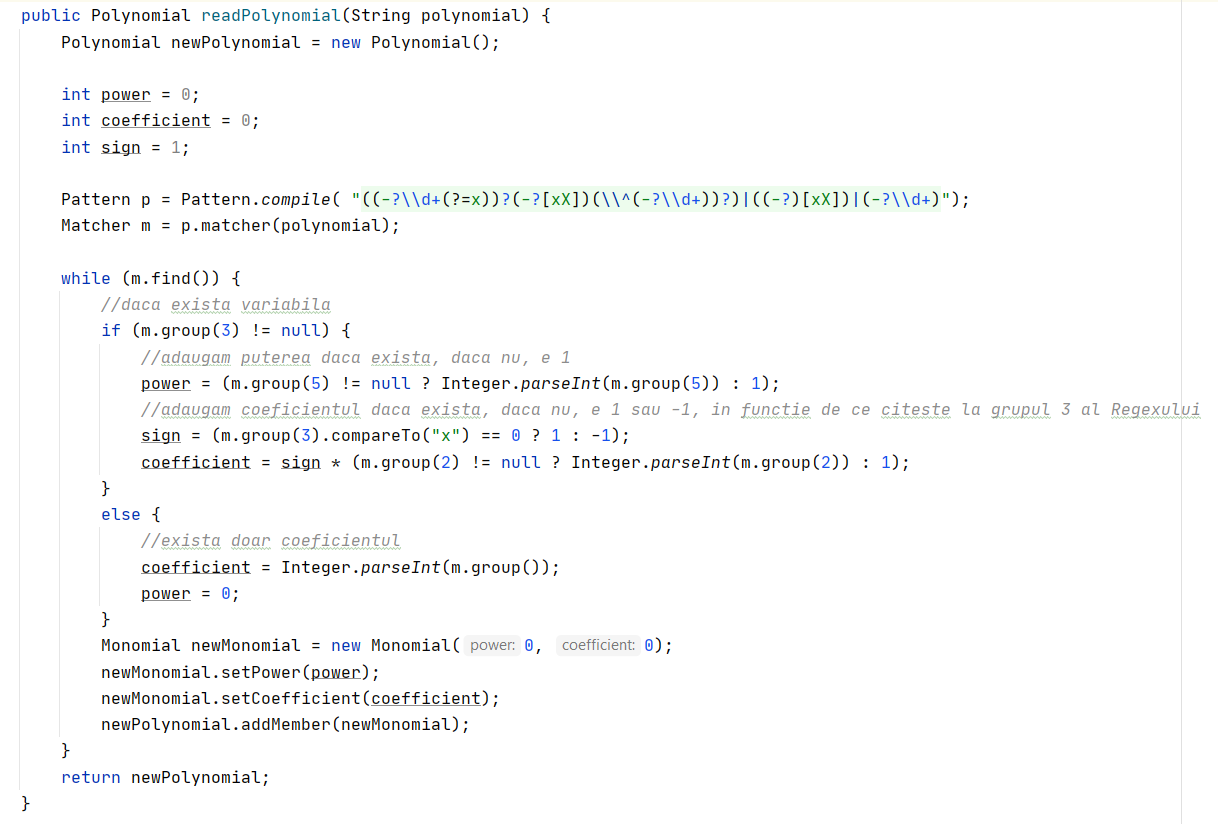


* *Pachetul “user\_interface”*

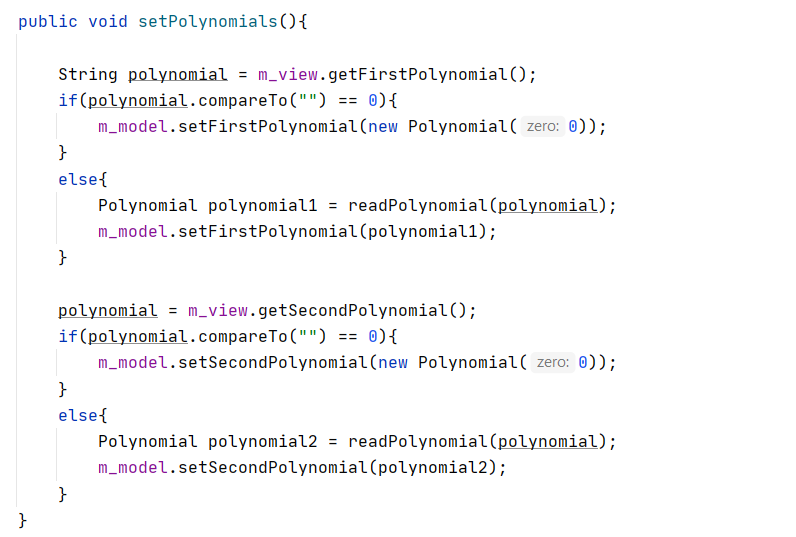
***Clasa “CalcController”***

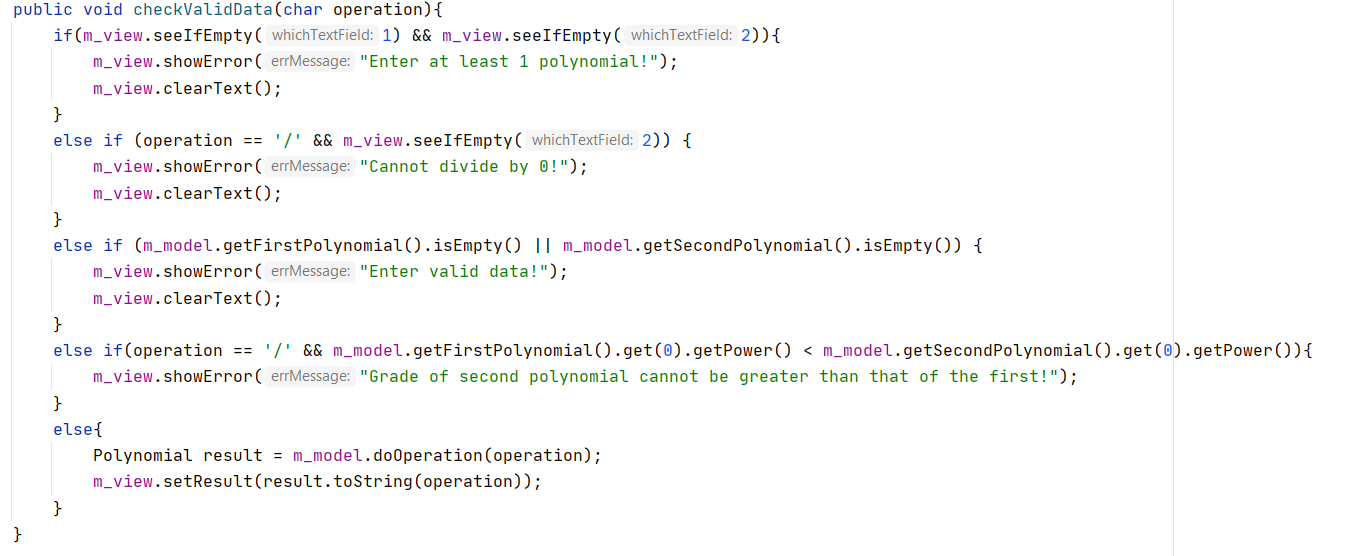
Contine un obiect de tip “CalcModel” si unul de tip “CalcView”.

Aici se va face citirea polinoamelor prin metoda “readPolynomial” care primeste ca parametru un “String polynomial” – sirul de caractere introdus de catre utilizator in CalcView (TextField) – ce va fi “tradus” cu ajutorul pachetului RegEx. Se verifica daca “polynomial” verifica modelul (pattern-ul) *"((-?\\d+(?=x))?(-?[xX])(\\^(-?\\d+))?)|((-?)[xX])|(-?\\d+)"* care va imparti sirul in mai multe grupuri si va extrage informatiile necesare:



Cat timp sirul, grupul gasit va respecta pattern-ul propus de RegEx, un nou monom va fi introdus intr-un polinom care va fi, la sfarsit, returnat.

Metoda “setPolynomials” va citi ambele polinoame si le va trimite mai departe la m\_model, unde vor fi folosite pentru calcule, iar metoda “checkValidData” va seta rezultatul numai dupa ce va verifica corectitudinea datelor introduse de utilizator atat in mod general cat si in raport cu operatia (ex. trebuie sa avem si un al doilea termen pentru a putea efectua impartirea). 



Constructorul acestei clase va adauga pentru fiecare buton al CalcView-ului (GUI-ului) un ActionListener – contine, deci, cate o clasa separata pentru fiecare operatie in parte.

***Clasa “CalcView”***

Aici se va crea interfata grafica cu care va interactiona utilizatorul.

Prin intermediul ei, utilizatorul nu trebuie sa se gandeasca cum se va ajunge / cum trebuie sa se ajunga la un rezultat, el doar va introduce date (prin TextField-uri), va trimite comenzi (prin butoane) si va primi rezultate, adica un raspuns de la aplicatie.

Diferite metode, mai ales cele de la sfarsit, ajuta programatorul in a face legatura cu celelalte clase si componente ale MCV-ului (transmiterea datelor, adica a celor doua polinoame, ori verificarea existentei lor – *ex. getFirstPolynomial, seeIfEmpty* -, adaugarea de actionListener-e corespunzatoare – *ex. addMultiplyListener* -, afisarea de erori in functie de raspunsul primit de la Controller – *ex. showError* -, etc.):



Toate acestea sunt posibile datorita faptului ca “CalcView” mosteneste clasa JFrame:



astfel incat constructorul clasei noastre creeaza insasi fereastra cu care va interactiona utilizatorul (interfata). Vom folosi tot felul de obiecte din pachetele “swing” si “awt” de tip JLabel, JTextField, JButton, si JPanel de care un user se va folosi pentru transmiterea de date si informatii, iar un programator pentru prelucrarea si modelarea lor.



Am impartit Frame-ul in mai multe panouri (panel-uri) pentru o mai buna organizare a componentelor din GUI, folosind metode de tip setLayout, add, pack, setAlignment.



* *Pachetul “business\_logic”*

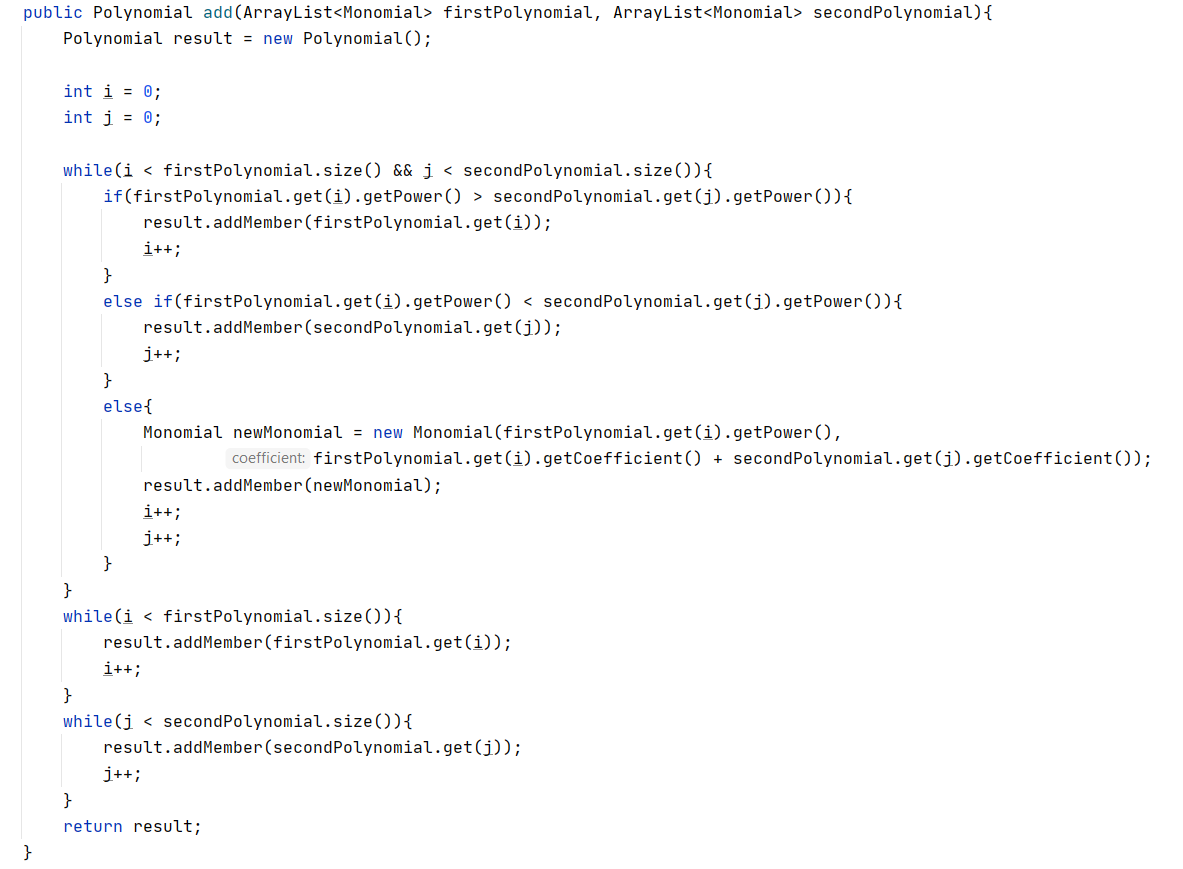
***Clasa “Operations”***

Aceasta clasa rezolva problema operatiilor ce trebuie efectuate asupra celor doua polinoame; contine numai metode care vor returna un obiect nou de tip “Polynomial”, fiecare primind ca parametri doua ArrayList-uri “firstPolynomial” si “SecondPolynomial”.

1. Metoda “add”

Folosindu-ne de doi indecsi, i si j, vom parcurge cele doua liste de monoame si vom hotari cum si ce vom adauga in rezultat (result) in functie de exponentul / puterea fiecarui monom in parte: vom compara un monom din primul si un monom din al doilea: cel ce este mai mare va fi adaugat primul in rezultat si se va trece la urmatorul termen din acesta (i++ sau j++). Cand doi termeni vor avea exponenti egali, vom crea un nou obiect de tip “Monomial” a carui exponent va fi egal cu cel gasit (in implementare a fost ales “firstPolynomial” ca fiind sursa) si coeficient, cu suma dintre cei doi ai monoamelor.

“while”-urile de la sfarsit au rolul de a adauga in rezultat termenii la care nu s-a mai ajuns(daca am atins capacitatea unuia din polinoamele sursa, tot e posibil ca in celalalt sa mai avem monoame ce trebuie luate in considerare).

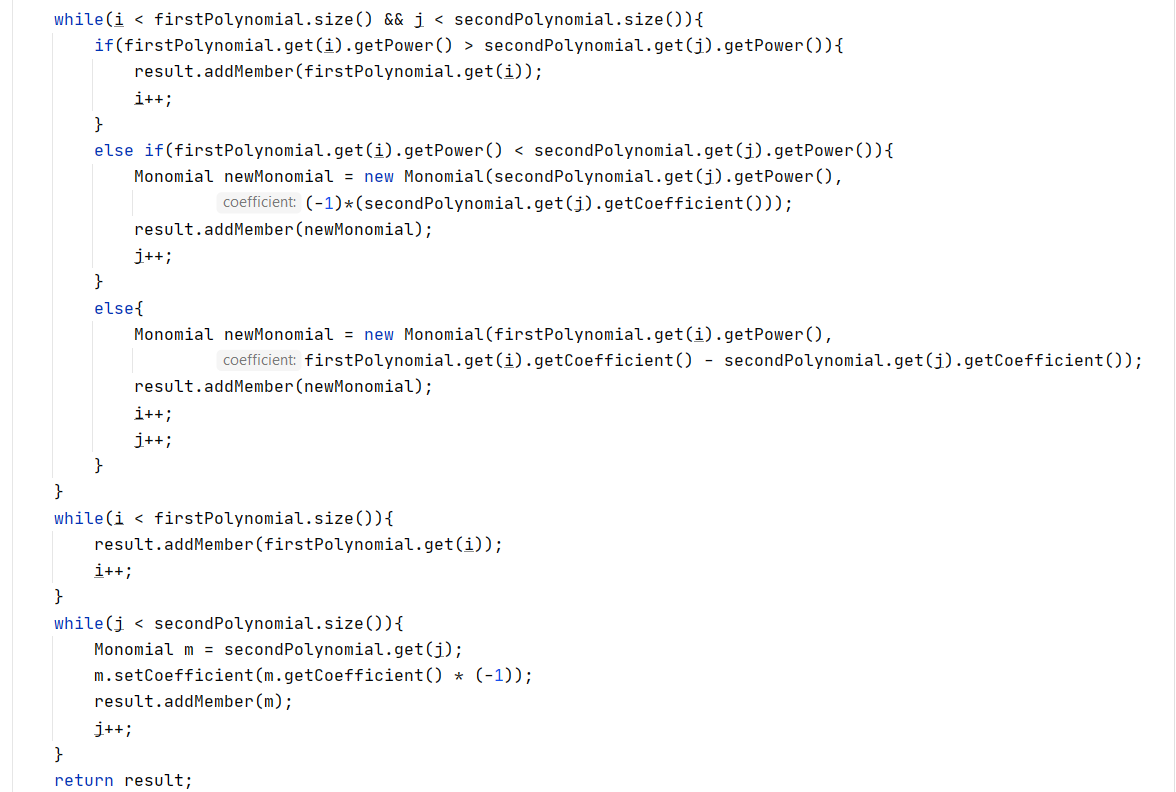


1. Metoda “subtract”

Asemanatoare metodei add, singura diferenta fiind semnul cu care se va lua in considerare cel de-al doilea polinom:

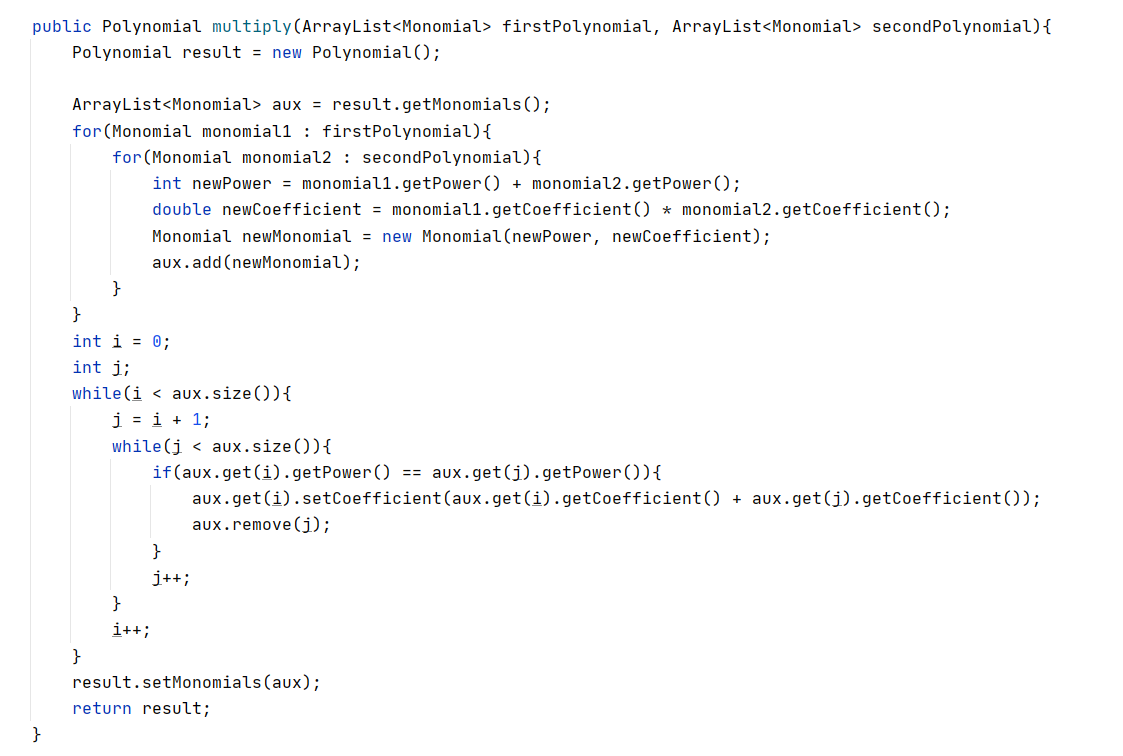
Daca exponentul acestuia este mai mare decat al primului, cream un nou monom al carui coeficient va fi -1 \* cel al termenului in discutie.

De asemenea, in cazul unei egalitati, noul monom al rezultatului va avea coeficientul egal cu diferenta dintre cel al primului si cel de-al doilea.



1. Metoda “multiply”

Se foloseste de ideea de “for in for”: inmultim fiecare termen din primul polinom, cu fiecare termen din al doilea, dupa care facem o noua parcurgere a rezultatului, un “while in while”, prin care verificam duplicate ale exponentului / puterii: daca exista, le adunam, iar al doilea termen va fi eliminat,



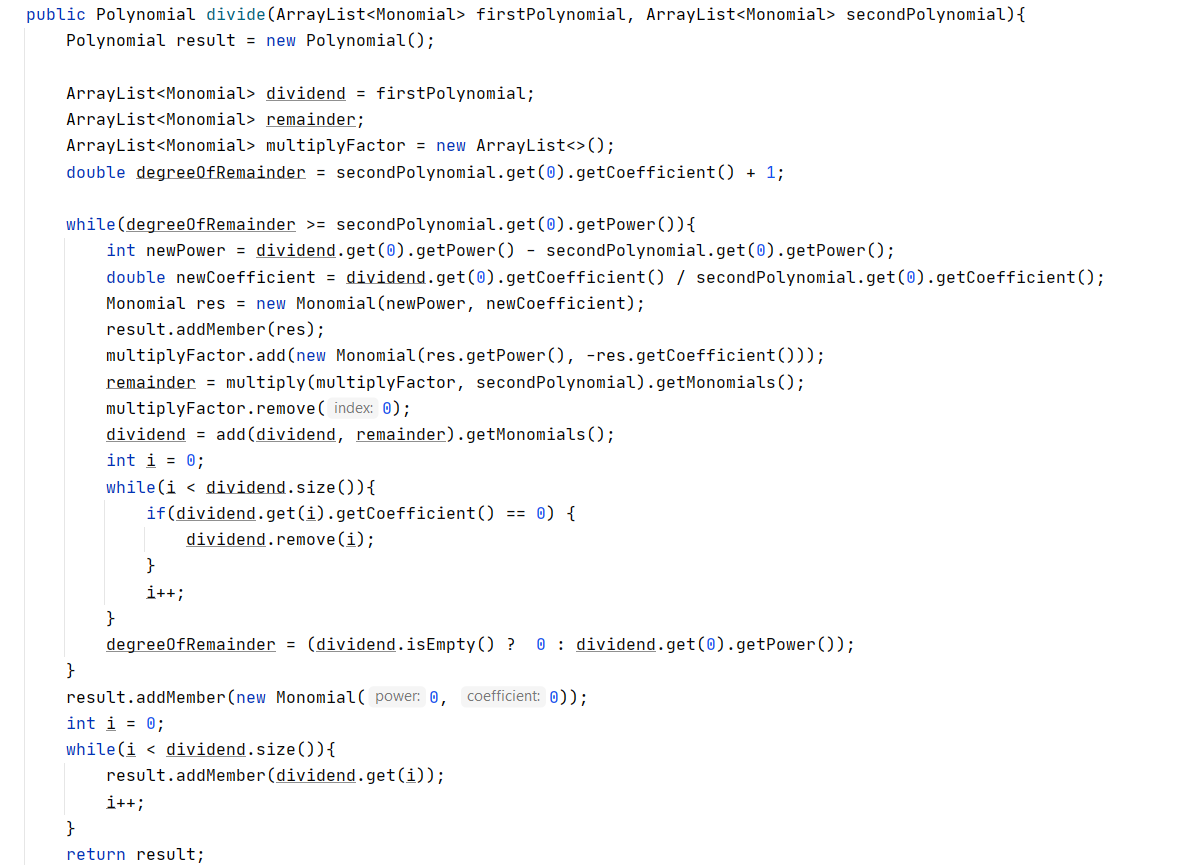
1. Metoda “divide”

Aceasta metoda a folosit algoritmul propus in suportul de curs:

Impartire treptata a “dividend”(“firstPolynomial” initial) cu primul termen al celui de-al doilea polinom din lista de parametri ai functiei (initial), dupa care cu primul termen al catului care ramane, inmultirea cu semn schimbat al acestuia (multiplyFactor) cu “secondPolynomial” – rezultat stocat in remainder si scaderea lor pentru aflarea noului impartitor(dividend), pana cand gradul (exponentul) catului (dividend) este mai mic decat cel al “secondPolynomial”.

Rezultatul va fi format prin adaugarea de la inceput al unui nou monom care este, practic, pentru fiecare pas, rezultatul impartirii.

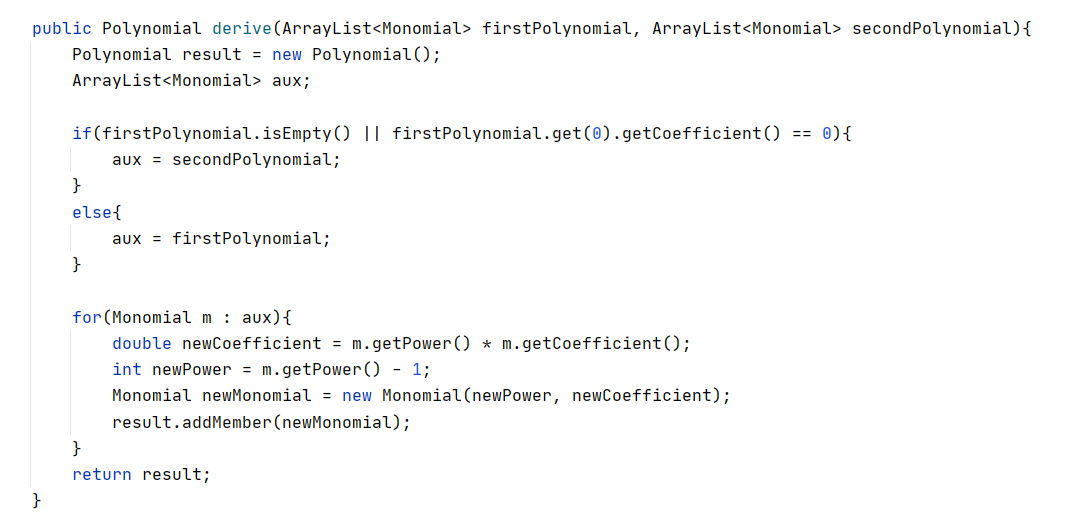
Fiindca fiecare functie din aceasta clasa trebuie sa returneze un singur polinom, am alse ca implementare sa stochez atat catul, cat si restul in acelasi polinom; am reusit asta prin separarea celor doi cu ajutorul unui monom de tipul “0x^0” = “new Monomial(0, 0)”. Afisarea din clasa “Polynomial” va pune in locul acestuia un string de tipul “ rest = ”.



1. Metoda “derive”

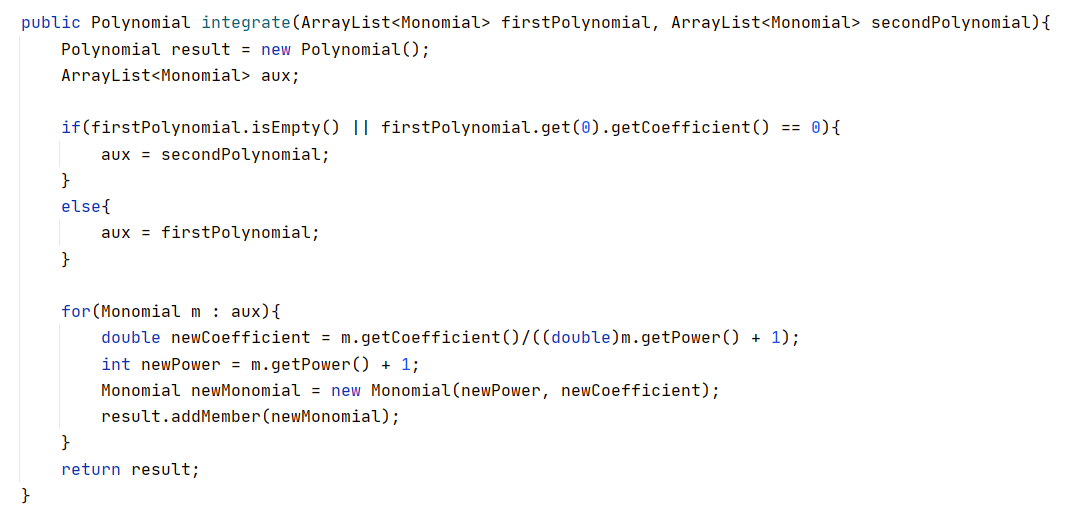
Am permis acestei metode sa se foloseasca, in primul rand, de primul polinom pentru a face operatia de derivare, iar in cazul in care acesta nu a fost introdus (este inexistent, de tipul “0x^0”), sa-l considere pe cel de-al doilea la calcule.

Rezultatul a fost creat prin parcurgerea polinomului luat ca si parametru, crearea de noi monoame cu valori ce sa respecte regulile de derivare si adaugarea acestora in “result”.



1. Metoda “integrate”

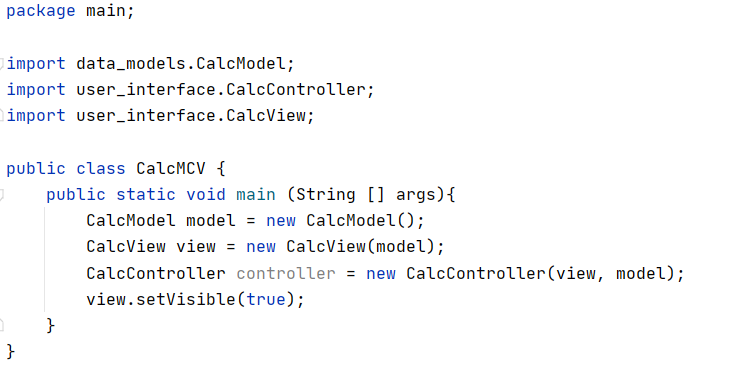
Urmeaza acelasi principiu ca si metoda de mai sus “derivate”, avand doar alte valori pentru exponent si coeficient, valori corespunsatoare regulilor de integrare:



* *Pachetul “main”*

***Clasa “CalcMCV”***

Aceasta clasa porneste aplicatia si permite lucrul cu GUI, facand legatura intre toate celelalte clase si stabilind relatiile care vor duce la buna functionare a interfetei grafice si a programului din spatele acesteia.



1. Rezultate

Am testat aplicatia folosind JUnit si creand clasa PolTest care va verifica corectitudinea fiecarei operatii implementate in parte.

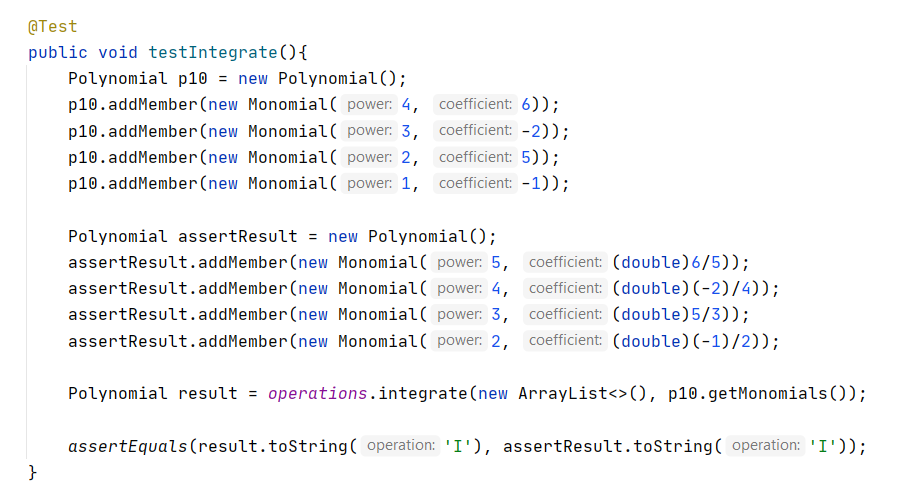
Am creat, deci, 6 metode diferite, in care am dat ca si parametri metodelor respective polinoame create ori prin constructori ai clasei “Monomial”, si prin metoda “add” a “Polynomial”, ori prin metoda clasei CalcController “readPolynomial”, ce foloseste RegEx-ul pentru citirea unui polinom, “extragerea” acestuia dintr-un String.

“assertEquals” este metoda care verifica corectitudinea calculelor facute, prin verificarea egalitatii dintre assertResult (polinom creat manual, egal cu rezultatul operatiilor dintre cele doua polinoame) si result (polinomul returnat de metoda corespunzatoare).

Exemplu pentru impartire:



Exemplu pentru integrare:



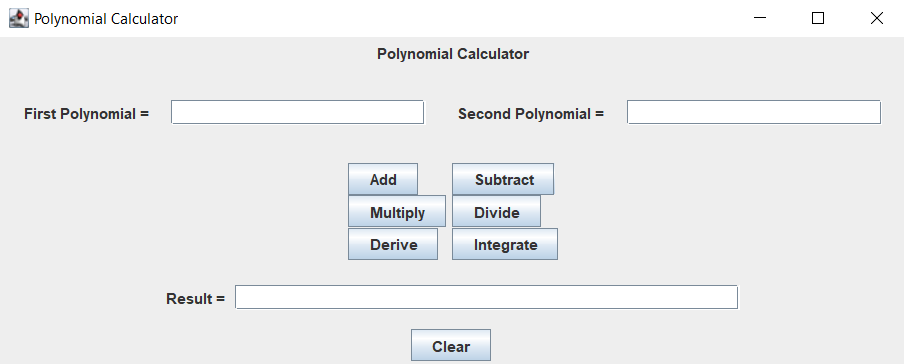
1. Concluzii

Prin implementarea acestui proiect s-a urmarit atat lucrul cu GUI-ul, cat si intelegrea relatiilor dintre clase, al mostenirii de metode, al lucrului cu liste, organizarea codului pentru o mai buna si usoara intelegere si crearea, per ansamblu, a unei aplicatii utile si usor de folosit pentru un eventual utilizator.

Noutati pentru mine, fata de ce am invatat semestrul trecut, a fost lucrul cu RegEx-ul, JUnit, reusind, de asemenea, sa ma familiarizez cu GitLab si cu proiectele de tip Meaven.

Consider ca diverse metode ar fi putut fi simplificate si scrise in mai putine linii de cod. Doua exemple ar fi “checkValidData” in care verific corectitudinea datelor introduse si, la pachet, cele doua metode “toString” din clasele “Monomial” si “Polynomial”, unde cu siguranta exista o varianta mai usoara si mai eleganta de afisare dar care, la momentul respectiv, mi-a scapat.

Varinata finala a proiectului (GUI-ul):



1. Bibliografie

*Operatii cu polinoame:*

[*https://en.wikipedia.org/wiki/Polynomial\_long\_division*](https://en.wikipedia.org/wiki/Polynomial_long_division)

*Regex:*

[*https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression*](https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression)

*JUnit:*

[*https://www.baeldung.com/junit-5?fbclid=IwAR39Om7OF7lZzhNzm3-3cNbMOJmgNlu7bA5uD5aQXXejnh7PDhtamxzjJoU*](https://www.baeldung.com/junit-5?fbclid=IwAR39Om7OF7lZzhNzm3-3cNbMOJmgNlu7bA5uD5aQXXejnh7PDhtamxzjJoU)

[*https://www.jetbrains.com/help/idea/testing.html*](https://www.jetbrains.com/help/idea/testing.html)

[*https://www.wyzant.com/resources/answers/627688/how-do-i-pass-in-a-polynomial-function-in-java*](https://www.wyzant.com/resources/answers/627688/how-do-i-pass-in-a-polynomial-function-in-java)

*FUNDAMENTAL PROGRAMMING TECHNIQUES*

*ASSIGNMENT 1 – SUPPORT PRESENTATION -Teams-*