DOCUMENTAȚIE

TEMA 1

NUME STUDENT: IFTENE IOAN-FLORIN

GRUPA: 30221

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 4](#_Toc95297887)

[4. Implementare 5](#_Toc95297888)

[5. Rezultate 7](#_Toc95297889)

[6. Concluzii 9](#_Toc95297890)

[7. Bibliografie 9](#_Toc95297891)

# Obiectivul temei

Obiectivul principal este de a proiecta și de a implementa un calculator pentru polinoame cu o interfață grafică, prin intermediul căreia utilizatorul să fie capabil să introducă două polinoame cu o singură variabilă și coeficienți întregi, să selecteze operația dorită (adunare, scădere, înmulțire, împărțire, derivare, integrare) și să vizualizeze rezultatul.

Obiectivele secundare care vor ajuta la atingera obiectivului principal sunt:

|  |  |
| --- | --- |
| **Obiectiv secundar** | **Capitolul în care e detaliat** |
| Descompunerea problemei în subprobleme, prin crearea a două clase, monom și polinom | 3 |
| Prelucrarea datelor de intrare în vederea extragerii celor 2 polinoame din datele introduse de utilizator de la tastatură | 4 |
| Crearea unei interfeței grafice, folosin JavaSwing | 4 |
| Utilizarea modelului arhitectural MVC (Model View Controller) | 4 |
| Implementarea operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare și salvarea rezultatului în vederea afișării în interfața grafică | 3, 4 |
| Utilizarea Junit pentru a testa corectitudinea operațiilor implementate | 5 |

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru a folosi calculatorul, utilizatorul va introduce două polinoame în câmpurile dedicate. Ele vor fi preluate ca și două stringuri, vor fi verificate și se vor crea polinoamele corespunzătoare lor. Se alege apoi operația dorită, apăsând butonul specific ei și se va afișa rezultatul, dacă totul este în regulă. Dacă cel puțin unul din polinoame nu este valid, se va afișa unul din mesajele “Polinom invalid”, "Introduceti 2 (doua) polinoame in campurile destinate!" sau "Folositi numai litera x ca si variabila pentru polinoame!", în funcție de cazul de eroare.

* Scenariul principal, în care calculatorul efectuează cu succes ceea ce se dorește este următorul:

Pasul 1: Utilizatorul deschide calculatorul pentru polinoame

Pasul 2: Utilizatorul introduce două polinoame corecte în câmpurile dedicate datelor de intrare (sau un singur polinom dacă se dorește efectuarea operației de derivare sau integrare)

Pasul 3: Utilizatorul apasă butonul corespunzător operației dorite

Pasul 4: Aplicația reușește să preia cele două stringuri din câmpurile de intrare și să formeze polinoamele

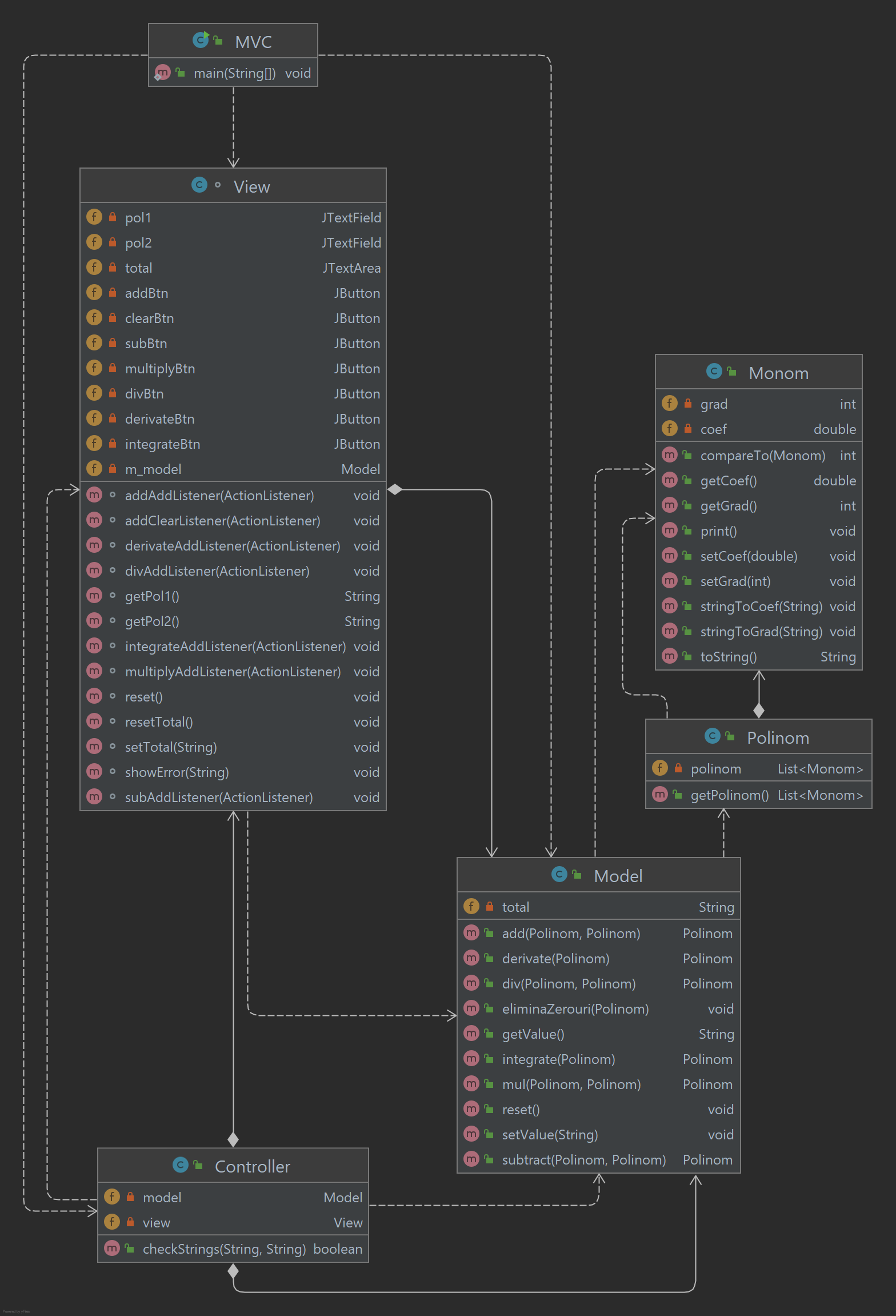
Pasul 5: Se efectuează operația cerută

Pasul 6: Rezultatul este afișat într-o zonă de text dedicată, de unde poate fi văzut de către utilizator.

* Scenarii alternative pot apărea în următoarele situații:

1. Utilizatorul introduce un singur polinom, dar dorește să efectueze o operație care necesită două polinoame (adunare, scădere, înmulțire, împărțire). În acest caz:
2. utilizatorului îi va apărea o fereastră cu mesajul "Introduceti 2 (doua) polinoame in campurile destinate!";
3. se revine la pasul 2 din scenariul principal.
4. Utilizatorul introduce cel puțin unul din cele două polinoame cu mai multe variabile, sau cu alte variabile în afară de “x”. În această situație:
5. utilizatorului îi va apărea o fereastră cu mesajul "Folositi numai litera x ca si variabila pentru polinoame!";
6. se revine la pasul 2 din scenariul principal.
7. Utilizatorul nu introduce numere întregi pentru coeficienții polinomului sau numere naturale pentru puterile variabilelor. În oricare din aceste două cazuri:
8. utilizatorului îi va apărea o fereastră cu mesajul “Polinom invalid”;
9. se revine la pasul 2 din scenariul principal.

# Proiectare

Diagrama UML de clase:

Am început proiectarea prin crearea claselor Monom și Polinom, clase care stau la baza întregii aplicații. Am folosit structura de date ArrayList în cadrul clasei Polinom, pentru a reține lista de monoame din care este alcătuit polinomul.

Am gândit apoi algoritmii necesari efectuării operațiilor care trebuie implementate:

* Pentru implementarea operației de adunare, algoritmul folosit pune ambele polinoame într-un polinom-rezultat, sortează toate monoamele în ordine descrescătoare după grad (cu ajutorul metodei *Collections.sort(),* aplicată listei de monoame corespunzătoare polinomului) și apoi le adună pe cele care au gradele egale, ținând minte doar monomul ”colector”, în care se efectuează adunarea, și eliminându-le pe celelalte din poinomul-rezultat. Același algoritm este folosit și pentru operația de scădere, diferența fiind date de înmulțirea cu ”-1” a tuturor coeficienților celui de-al doilea polinom înainte de aplicarea pașilor descriși anterior;
* Pentru operația de înmulțire, algoritmul inmulțește fiecare monom din primul polinom cu fiecare monom din al doilea polinom și salvează rezultatul într-un polinom-rezultat. Se efectuează apoi o adunare a rezultatului cu ”0”, cu scopul de a aduna între ele monoamele cu același grad, rezultate în urma înmulțirii;
* Un alt algoritm folosit este cel pentru efectuarea operației de împărțire. Am abordat această operație cu metodele de împărțire a polinoamelor învățate la matematică. Mai întâi împart primele două monoame între ele, inmulțesc rezultatul cu al doilea polinom (împărțitorul) și îl scad din primul polinom (deîmpărțitul), rezultatul fiind noul deîmpărțit. Acești pați sunt repetați până când gradul deîmpărțitului este mai mic decât gradul împărțitorului, sau până când deîmpărțitul este 0. Restul împărțirii va fi ultimul deîmpărțit obținut în cadrul algoritmului de împărțire. Dacă acesta este 0, împărțirea nu are rest;
* Pentru derivare și integrare, se parcurge fiecare monom din cadrul polinomului pentru care se efectuează operația dorită și se modifică coeficientul și gradul monoamelor, corespunzător cu operația care se efectuează, rezultatul fiind salvat într-un polinom-rezultat.

# Implementare

Pentru implementarea aplicației am folosit modelul arhitectural MVC. Astfel, am despărțit aplicația în 3 părți: Model, View și Controller. În model am implementat metodele corespunzătoare fiecărei operații, în View am creat interfața grafică, iar în Controller am descris ce trebuie să facă aplicația la apăsarea fiecărui buton.

Clasele folosite pentru realizarea aplicației și principalele metode din fiecare sunt următoarele:

1. Clasa Monom

Această clasă conține două variabile, una de tip int, pentru reținerea gradului monomului și alta de tip double, pentru coeficient. Am ales să folosesc o variabilă de tip double pentru coeficient, deși utilizatorul poate introduce doar coeficienți întregi, deoarece pot apărea situații în cadrul operațiilor de împărțire și integrare în care coeficienții rezultatului nu vor fi numere întregi, iar astfel voi putea trata și aceste cazuri în mod normal.

Pentru construirea monomului se pot da ca și parametrii gradul și coeficientul, sau un string. Constructorul care primește un string ca și parametru va verifica dacă șirul primit ca și parametru conține ”x”, ”\*” sau ”^” și tratează fiecare caz în parte pentru a extrage coeficientul și gradul polinomului care urmează a fi creat. Acest constructor folosește metodele *stringToGrad* și *stringToCoef*, metode care transformă un string, primit ca și parametru într-o varibilă de tip int și apoi o atribuie gradului, respectiv coeficientului monomului care este în curs de construire. În cazul în care cele 2 metode nu reușesc conversia stringurilor în variabile de tip intreg, se afișează un mesaj cu textul ”Polinom invalid”.

În cadrul acestei clase am implementat și o metodă *toString*, care transformă monomul într-un string de forma ”semn”+”coeficient”+”\*x”+”^grad”.

Totodată, clasa implementează *Comparable*, metoda *compareTo* fiind implementată astfel încât, atunci când este apelată metoda *Collections.sort()* pentru o listă de monoame, acestea să fie sortate în ordinea descrescătoare a gradelor.

1. Clasa Polinom

Această clasă conține o listă de monoame, implementată cu structura de date *ArrayList.* Pentru a construi un polinom, constructorul primește un string ca și parametru, îl separă în string-uri corespunzătoare monoamelor, utilizând metoda *split()* și regex-ul *”(?=\+|-)”,* pentru a extrage din șirul inițial subșirurile aflate între semnele ”+” și ”-”. Din fiecare subșir rezultat se construiește cîte un monom, care este adăugat în lista de monoame a clasei Polinom.

1. Clasa Model

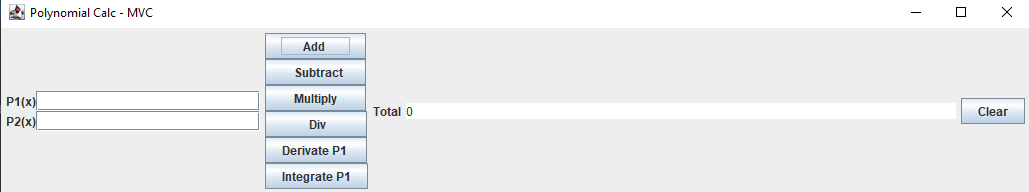
În această clasă am implementat metodele corespunzătoare operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare, pe baza algoritmilor descriși în capitolul 3. Clasa conține o variabilă de tip String, numită *total*, în care se salvează rezultatele operațiilor, sub formă de string. Această variabilă va fi folosită pentru a putea afișa rezultatul în interfața grafică. Toate metodele menționate anterior, corespunzătoare operațiilor, sunt de tip Polinom și returnează rezultatul obținut. În cadrul lor, înainte de a se returna rezultatul, acesta este salvat în variabila total, transformîndu-se polinomul în string prin apelarea metodei *toString* pentru fiecare monom din care este alcătuit polinomul.

Pe lângă metodele descrise, clasa mai conține și metoda *eliminaZerouri*, folosită pentru a elimina termenii cu coeficientul 0 din cadrul polinoamelor care sunt folosite pentru realizarea operatiilor, sau a rezultatelor obținute.

1. Clasa View

În această clasă este implementată interfața grafică a aplicației. Câmpurile clasei sunt reprezentate de două JtextField, în care utilizatorul introduce datele de intrare (cele 2 polinoame), o variabilă JtextArea, în care este afișat rezultatul, 6 butoane de tip Jbuttons, corespunzătoare operațiilor pe care le poate efectua calculatorul și încă un buton *Clear* pentru a șterge mai ușor datele introduse și rezultatul și pentru a pregăti introducerea de noi polinoame. Constructorul clasei View primeste ca și parametru un obiect de tip Model. În cadrul constructorului, se scrie rezultatul din Model în JtextArea și se aranjează câmpurile clasei într-un JPanel.

Clasa mai conține metode pentru resetarea câmpului pentru rezultat, pentru obținerea polinoamelor din JTextField, pentru afișarea unor mesaje de eroare și pentru adăugarea de Action Listener pentru fiecare buton.



1. Clasa Controller

Această clasă conține două câmpuri, unul de tip Model și celălalt de tip View.

În cadrul constructorului sunt adăugate Action Listener butoanelor din View, pe baza subclaselor definite în cadrul clasei.

Subclasele din cadrul clasei Controller sunt AddListener, SubtractListener, MultiplyListener, DerivateListener, IntegrateListener, DivListener și ClearListener, toate implementând ActionListener.

În toate aceste subclase, sunt preluate stringurile introduse de utilizator, sunt șterse spațiile goale și apoi este verificată corectitudinea datelor de intrare, cu ajutorul metodei *checkStrings*. În cadrul acestei metode se verifică dacă au fost introduse 2 polinoame (pentru operațiile care necesită acest lucru) și dacă polinoamele introduse folosesc doar litera ”x” ca și varibilă, restul verificărilor fiind efectuate în cadrul clasei Monom, după cum s-a explicat anterior. Dacă cele 2 date de intrare au trecut de metoda *checkStrings,* se construiesc cele 2 polinoame pe baza lor și, dacă totul este în regulă și polinoamele au putu fi construite fără vreun mesaj de eroare, se efectuează operația specifică butonului care a fost apăsat, prin apelarea metodei specifice, din clasa Model. Rezultatul obținut în cadrul clasei Model este preluat cu ajutorul câmpului de tip Model și afișat în câmpul specific din View.

# Rezultate

Pentru testare am folosit Junit. Scenariile pe care le-am testat sunt următoarele:

1. Adunarea/Scăderea

* test1: polinom1 = ”7”, polinom2 = ”-11”

Testează adunarea/scăderea a două polinoame de grad 0 (2 constante), al doilea fiind negativ;

* test2: polinom1 = ” 2\*x”, polinom2 = ”-11”

Testează adunarea/scăderea unui polinom de grad 1 cu o constantă negativă;

* test3: polinom1 = ”11”, polinom2 = ”2\*x”

Testează adunarea/scăderea unui polinom de grad 1 cu o constantă, constanta fiind de această dată pe prima poziție;

* test4: polinom1 = ”2\*x”, polinom2 = ”x^2-11”

Testează adunarea/scăderea a două polinoame de grade diferite (grad 1 și grad 2);

* test5: polinom1 = ” x^11+4\*x^2+2\*x+12”, polinom2 = ”x^2-11”

Testează adunarea/scăderea unui polinom de grad format din 2 cifre (11) cu un polinom de grad mai mic (2);

* test6: polinom1 = ” 2\*x”, polinom2 = ” -2\*x-11”

Testează adunarea/scăderea unui polinom format dintr-un monom de grad 1 cu un polinom format din inversul monomului din primul polinom și o constantă;

* test7: polinom1 = ” 100\*x^11-4\*x^2+2\*x-12” polinom2 = ” -100\*x^11+4\*x^2-2\*x+12”

Testează adunarea unui polinom cu coeficient și grad mare cu inversul său;

* test8: polinom1 = ” 100\*x^11-4\*x^2+2\*x-12”, polinom2 = ” 100\*x^11-4\*x^2+2\*x-12”

Testează scăderea unui polinom cu coeficient și grad mare cu el însuși;

* test9: polinom1 = ”0”, polinom2=”0”

Testează adunarea/scăderea a două polinoame formate din termenul nul;

* test10: polinom1 = ” 100\*x^11-4\*x^2+2\*x-12”, polinom2 = ”4x^2”

Testează adunarea a două polinoame de grade diferite, al doilea polinom regăsindu-se în primul, cu semn schimbat.

* test11: polinom2 = ”4x^2”, polinom2 = ” 100\*x^11+4\*x^2+2\*x-12”

Testează scăderea a două polinoame de grade diferite, primul polinom regăsindu-se în al doilea

1. Înmulțirea

* test1: polinom1 = ”5”, polinom2 = ”-11”

Testează înmulțirea a două constante;

* test2: polinom1 = ”2x”, polinom2 = ”-11”

Testează înmulțirea unui polinom de grad 1 cu o constantă;

* test3: polinom1 = ”x-1”, polinom2 = ”x-1”

Testează înmulțirea a două polinoame identice, de grad 1, pentru a verifica dacă termenii cu același grad din rezultat sunt adunați între ei;

* test4: polinom1 = ” x^3+2\*x+7”, polinom2 = ”0”

Testează înmulțirea unui polinom de gradul 3 cu 0;

* test5: polinom1 = ”x-5”, polinom2 = ”x^4+7x-2”

Testează înmulțirea unui polinom de grad 1 format din 2 monoame cu un polinom de gradul 4 format din 3 monoame;

* test6: polinom1 = ”2x”, polinom2 = ”-2x-11”

Testează înmulțirea a două polinoame de același grad, al doilea având coeficientul termenului dominant negativ;

* test7: polinom1 = ”10x^11-4x^2+2x-12”, polinom2 = ” 20x^11”

Testează înmulțirea a două polinoame de grad mare (11)

* test8: polinom1 = ”0”, polinom2 = ”0”

Testează înmulțirea a două polinoame nule;

1. Împărțirea

* test1: polinom1 = ”0”, polinom2 = ”-11”

Testează împărțirea lui 0 la o constantă;

* test2: polinom1 = ”2x”, polinom2 = ”0”

Testează împărțirea la 0 a unui polinom;

* test3: polinom1 = ”x-1”, polinom2 = ”x-1”

Testează împărțirea unui polinom la el însuși;

* test4: polinom1 = x^2-2x+1”, polinom2 = ”x-1”

Testează împărțirea unui polinom la unul din factorii lui ireductibili;

* test5: polinom1 = ”x-5”, polinom2 = ”x^4+7x-2”

Testează împărțirea unui polinom de grad 1 la un polinom de grad mai mare (4);

* test6: polinom1 = ”2x”, polinom2=”-2x-11”

Testează împărțirea a două polinoame de același grad;

* test7: polinom1 = ”x^8”, polinom2 = ”x-4”

Testează împărțirea cu rest a două polinoame de grade diferite;

* test8: polinom1 = ”x^3+2x^2+10x+7”, polinom2 = ”x-5”

Testează împărțirea cu rest a două polinoame de grade diferite;

* test9: polinom1 = ”x^8”, polinom2 = ”2x-7”

Testează împărțirea cu rest a două polinoame de grade diferite, termenii din rezultat neavând coeficienți întregi;

* test10: polinom1 = ”0”, polinom2 = ”0”

Testează împărțirea a două polinoame nule (împărțire la 0).

1. Derivarea

* test1: polinom1 = ”7”

Testează derivarea unei constante;

* test2: polinom1 = ”x”

Testează derivarea unei variabile;

* test3: polinom1 = ”4x+7”

Testează derivarea unui polinom de grad 1;

* test4: polinom1 = ”9x^5+6x^4+10x^2+4”

Testează derivarea unui polinom de gradul 5, format din 4 termeni;

* test5: polinom1 = ”100\*x^11-4\*x^2+2\*x-12”

Testează derivarea unui polinom de grad 11, cu coeficient mare (100).

1. Integrarea

* test1: polinom1 = ”0”

Testează integrarea lui 0 (rezultatul trebuie să fie o constantă);

* test2: polinom1 = ”5”

Testează integrarea unei constante;

* test3: polinom1 = ”-1”

Testează integrarea unei constante negative;

* test4: polinom1 = ”2x”

Testează integrarea unei monom de gradul 1;

* test5: polinom1 = ”5x^4+4x^3+2x+4”

Testează integrarea unui polinom de grad 4, format din 4 termeni;

* test6: polinom1 = ”9x^11+6x^4-8x^3-13”

Testează integrarea unui polinom de grad 11, rezultatul avînd coeficienți reali;

# Concluzii

Am reușit să îndeplinesc toate obiectivele acestui proiect: folosirea unui design de programare orientată pe obiect (folosirea încapsulării), utilizarea de liste în locul vectorilor și a structurilor de tip foreach, în loc de for, implementarea unei interfețe folosind Java Swing, implementarea operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare pentru polinoame, folosirea modelului arhitectural MVC, testarea folosind JUnit, utilizarea regex pentru a extrage termenii polinomului.

Rezolvând această temă, am învățat cum se folosesc expresiile regex pentru a extrage anumiți termeni dintr-un șir, cum se testează funcționalitățile unui program folosind JUnit, am înțeles mai bine modelul arhitectural MVC și am deprins folosirea foreach-ului în loc de for și a listelor în loc de vectori.

# Bibliografie

1. <https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf>
2. <https://ro.wikipedia.org/wiki/Model-view-controller>
3. <https://www.geeksforgeeks.org/split-string-java-examples/>
4. <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>
5. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split>
6. <https://www.youtube.com/watch?v=_FSXJmESFmQ>
7. <https://www.tabnine.com/code/java/methods/java.lang.String/chars>
8. <https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html>
9. <https://www.vogella.com/tutorials/JUnit/article.html>
10. <https://www.baeldung.com/junit-5>