**Calculator de polinoame**

Nume: Brehuescu Andrei

Grupa: 30224

Facultatea: CTI An2

**Calculator de polinoame**

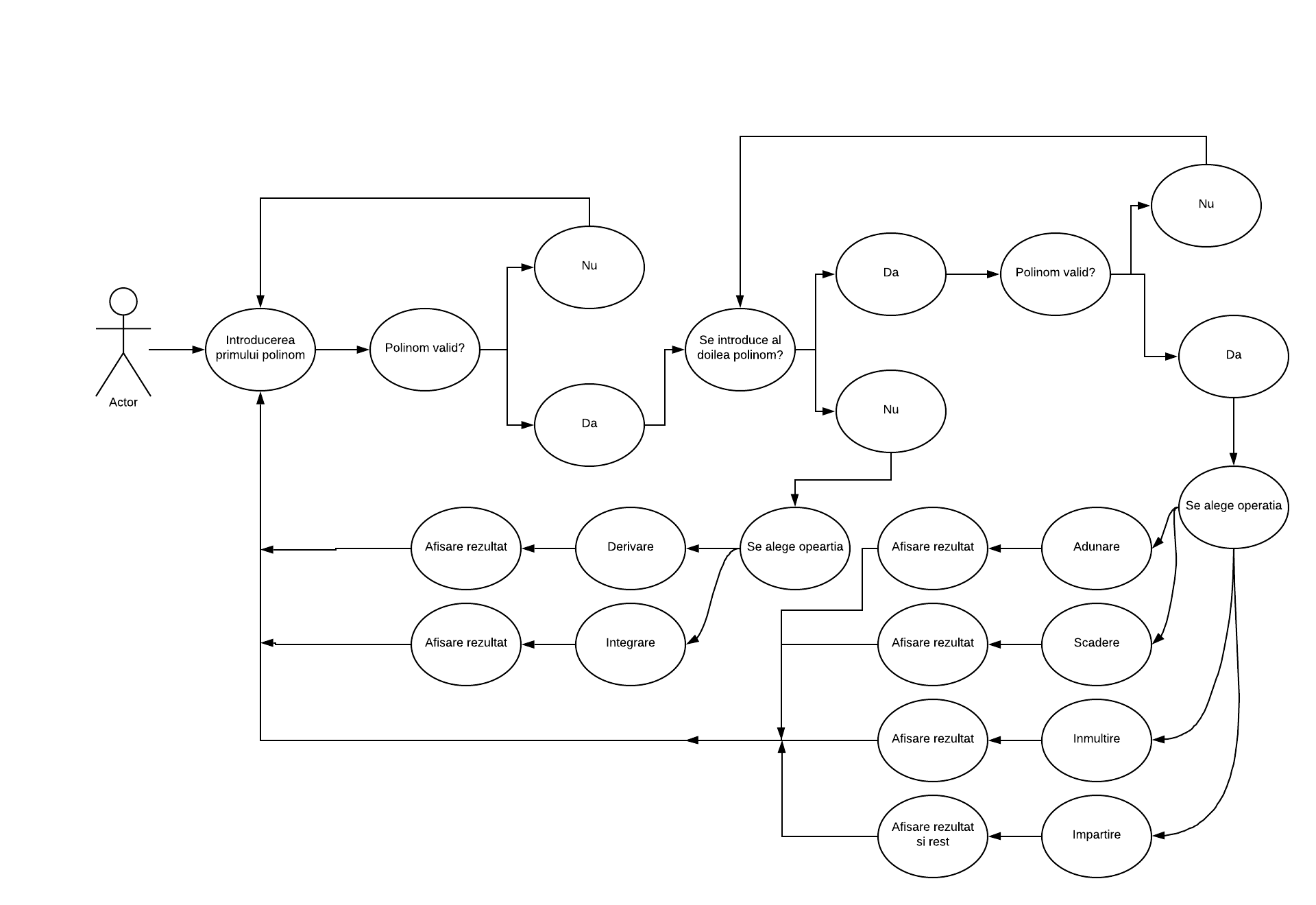
**1. Obiectivul lucrarii**

Lucrarea prezintă proiectarea și implementarea unui calculator de polinoame cu o interfață grafică prin care utilizatorii pot introduce polinoame și pot alege operația dorită (adunare, scădere, inmulțire, impărțire, derivare sau integrare).

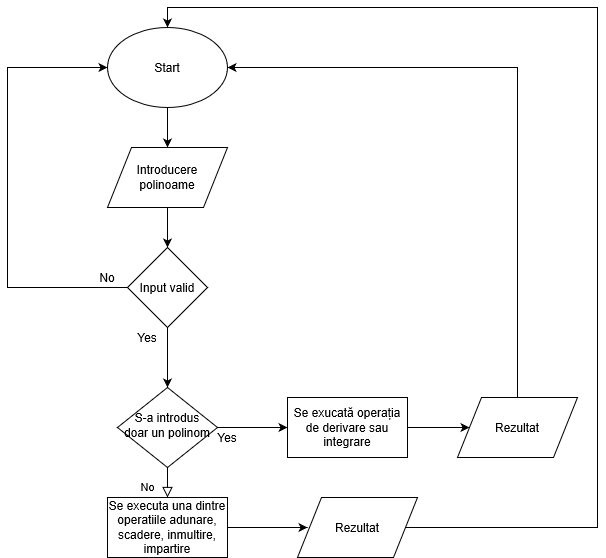
Obiectivele secundare sunt :

* implementarea unor algoritmi eficienți pentru fiecare operație (adunare, scadere, etc.)
* crearea unei interfețe grafice ușor de utilizat, cat mai intuitive
* identificarea datelor eronate ce ar putea fi introduse de către utilizator
* generarea rezultatului corect și a restului(in cazul imparțirii)
* afișarea rezultatelor pe interfața grafică

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



Calculatorul de polinoame poate primi la intrare unul sau doua polinoame in funcție de operația cerută. In cazult operațiilor de derivare și integrare doar câmpul corespunzător primului polinom va trebui completat, iar pentru celelalte operații vor trebui introduse doua polinoame. În cazul introducerii unor date eronate ( se va considera doar ‘x’ drept necunoscută în cadrul polinoamelor ), va apărea un mesaj avertizând faptul că a fost introdus un set de date invalide, iar operația selectată nu va fi realizată. Calculatorul acceptă introducerea polinoamelor în orice ordine a gradelor.

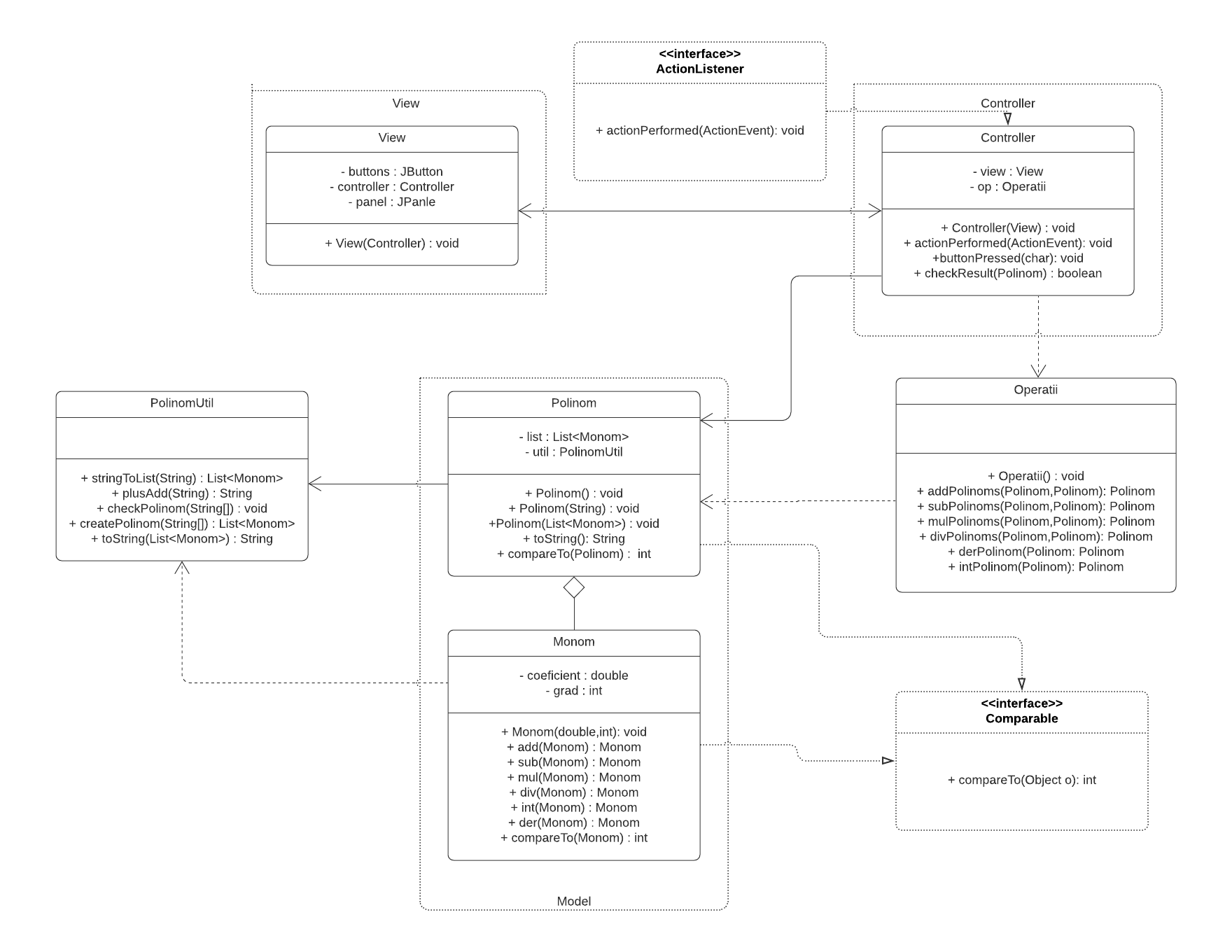
****

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

Proiectatea este bazătă pe modelul MVC astfel având la baza clasa View care se ocupă de intefața grafică, respectiv aranjarea butoanelor, a câmpurilor pentru text si de adaugarea componentei funcționale a acestora ( actionListener ) . Clasa controller se ocupă de gestionarea acțiunilor ce au loc pe interfața grafică (introducerea textului, apăsarea butoanelor) și in funcție de butoanele apăsate si datele introduse apelează metode din partea de Model .

Modelul este alcătuit din clasele Polinom și Monom și se ocupă de stocarea polinoamelor într-o formă utilizabilă de clasa Operații unde sunt descrise metodele corespunzatoare operațiilor.

Structurile de date folosite sunt reprezentate prin clasa Monom, Polinom si RezDiv. În cadrul polinomului, un monom este un termen reprezentat prin coeficientul lui x si gradul acestuia, polinoamele sunt reprezentate de o lista de monoame, iar RezDiv este o clasă pentru a furniza rezultatul operației de imparțire, clasa conține doua polinoame ( rezultatul si restul ) .

Interfețele folosite sunt Comparable și ActionListener. Interfața Comparable este utilizată in cadrul clasei Monom pentru a realiza sortarea listei de Monoame din cadrul polinomului . Interfața ActionListener este utilizată in clasa responsabilă de controlul butoanelelor de pe interfața grafică .

4. **Implementare**

Clasa View conține ca atribute principale 6 butoane, unul pentru fiecare operație, 2 câmpuri pentru preluarea datelor introduse, 2 polinomane, și doua câmpuri de text pentru afișarea rezulatului și a restului, când este cazul. Clasa view extinde JFrame, iar in constructorul acestei clase sunt așezate componentele interfeței grafice folosindu-se ca mod de aranjare GridLayout. Acesta clasa contine si o instanță a Clasei controller utilizat pentru a se adapta un model MVC si pentru dependența dintre clase.

Clasa Controller conține ca atribute o instanță a clasei View si o instanță a clasei Operații. Clasa implementează interfața ActionListener și suprascrie metoda actionPerformed, unde sunt controlate acțiunile în funcție de butoanele apăsate. În metodata actionPerformed se verifică daca este apăsat unul dintre butoanele corespunzătoare operațiilor (adunare, scădere, înmulțire, impărțire, integrare sau derivare) și este apelată o alta metodă( buttonPressed ) care, in funcție de butonul apăsat, va verifica datele introduse și va calcula rezultatul și restul cu metodele din clasa Operatii. In clasa Controller a mai fost definită o metodă care verifica daca rezultatul este 0 sau nu pentru evitarea excepțiilor generate de accesarea unei liste null.

Clasa Monom conține ca atribute un coeficient de tip float si un grad de tip int. Coeficientul este de tip float deoarece este necesar la operația de imparțire. Această clasă conține si operațiile pe monoame( adunarea a doua monoame, scăderea, etc), necesare simplificării complexității opearțiilor pe polinoame și implementează interfața Comparable ( cu suprascrrierea metodei compareTo ) pentru generarea unui polinom corespunzător celui introdus de la tastatură, dar in ordinea descrescătoare a gradelor. In cazul metodelor ce realizează operații pe monoame acestea primesc ca parametru un alt Monom ( cu excepția derivării si integrării) si realizează operațiile corespunzătoare, returnând un nou monom pentru a nu modifica datele de intrare.

Clasa Polinum conține ca atribut o listă de monoame și o referință la un obiect de tipul PolinomUtil, utilizată pentru conversii din șir de caractere în listă de polinoame și invers. Polinomul poate primi ca parametru pentru constructor atât o listă de monoame pentru instanțierea directă a polinomului, dar și un șir de caractere caruia i se vor aplica conversii pentru generarea listei.

In cadrul clasei PolinomUtil se realizează conversia din șir de caractere in listă de monoame (polinom) prin următorii pași:

* se proceseaza sirul de caractere și se inlocuiește semnul ‘-‘ cu ‘+-‘
* se desparte șirul de caractere într-un șir de cuvinte , separarea se face dupa semnul ‘+’
* fiecare cuvânt va fi procesat individual și va genera un monom pentru listă
* dupa generarea listei , aceasta va fi sortată prin apelul metodei Collection.sort() , care sortează in functie de cum a fost definită metoda compareTo din clasa Monom( in acest caz in ordinea descrescătoare a gradului)

Clasa polinom implementeaza interfata Comparable , cu suprascrierea metodei compareTo pentru a usura compararea polinoamelor, utilizata in metoda de impartire.

Clasa PolinomUtil conține atât metode pentru verificarea datelor introduse de utilizator, iar în cazul in care sunt valide se vor face conversii pentru generarea unui polinom, cât si metode pentru generarea unui șir de caractere pe baza unui polinom pentru a putea fi afișat pe interfața grafică în câmpurile pentru text. Procesul de conversie din șir de caractere în listă de monoame se face cu ajutorul mai multor metode, fiecare metoda având un singur rol. Am ales definirea mai multor metode pentru un singur proces (cel de conversie) pentru a putea avea un cod cât mai ușor de ințeles și de urmărit in cazul apariției diferitelor erori. Prima metodă necesară acestei conversii , plusAdd, procesează un șir de caractere astfel încat unde se găseste caracterul‘-‘, aceasta va fi înlocuit de caracterele “+-“, iar noul șir este returnat pentru următoarea procesare. Pe baza noului șir de caractere se va genera un șir de cuvinte cu ajutorul metodei split(String regex) care împarte șirul de caractere în cuvinte dupa un separator specificat prin regex ( în acest caz ‘+‘). După generarea șirului de cuvinte va fi apelată o metodă de verificare pentru a identifica posibilele date eronate introduse de către utilizator ( o altă variabilă necunoscută inafară de x, coeficienți care nu sunt numere întregi sau reale, grade care nu sunt numere întregi), metoda aruncând o excepție in cazul identificarii unei astfel de erori introduse de catre utilizator. În cazul in care datele sunt valide va fi apelată o noua metodă, createPolinom, care primește ca parametu șirul de cuvinte și returnează lista de monoame pentru polinom. Această metodă parcurge fiecare cuvant și preia coeficientul și gradul, pentru a crea un monom și a-l adăuga în listă. Odata creată lista de monoame aceasta este sortată si returnată polinomului.

Clasa WrongInputException extinde clasa Exception și conține un mesaj. Clasa este utilizată doar in cazul in care utilizatorul introduce date eroanate, metodele aruncând o excepție de acest tip.

Clasa RezDiv conține ca atribute doua polinoame, rezultatul și restul împărțirii a două polinoame. S-a ales definirea unei clase noi pentru stocarea acestui rezultat pentru a respecta o manieră de programare orientată pe obiecte.

Clasa Operații conține metodele corespunzătoare fiecarei dintre operațiile polinoamelor. Metodele responsabile de adunarea, scăderea si înmulțirea polinoamelore primesc ca parametru de intrare doua polinoamele și returnează un polinom. Operația de împarțire primește ca parametru de intrare doua polinoame și returneaza un obiect de tipul RezDiv . Metodele pentru integrare și derivare au ca parametru de intrare un singur polinom și returnează un polinom rezultat.

Operațiile de adunare și scădere ( metodele addPolinoms și subPolinoms ) sunt implementate asemănător. Se folosește o manieră de “ interclasare ” a celor doua polinoame. Se parcurg împreuna ambele liste de moanoame fiecare cu in iterator diferit de la cel mai mare grad către cel mai mic. În cazul în care gradul unui monom din cele doua liste este mai mare, acesta va fi adaugat în lista rezultatului și iteratorul corespunzător incrementat, dar dacă gradele sunt egale se realizează operația, de adunare sau scădere, pe monoame, iar monomul rezultat este trecut în listă.

Operația de înmulțire este realizată prin parcurgerea monom cu monom a unui singur polinom, iar pentru fiecare monom, acesta este înmulțit cu toate monoamele din celălalt polinom pentru generarea unui polinom nou. Acest polinom nou este adunat la un polinom rezultat.

Operația de derivare este realizată prin parcugerea listei de monoame și înmulțind coeficientul cu gradul, dupa care gradul este decrementat. Daca gradul este deja 0, atunci monomul va fi șters .

Operația de integrare este inversa operației de derivare, se parcurge fiecare monom iar gradul este incrementat cu unu și coeficientul este împartit la această valoarea .

Opearația de împarțire este realizată astfel: monomul cu gradul cel mai mare al deîmpărțitului este împărțit la monomul cu gradul cel mai mare al împărțitorului. Monomul rezultat va fi adăugat în lista rezultatului și înmulțit cu împărțitorul. Rezulatul acestei înmulțiri va fi scăzut din deîmpărțit iar gradul acestuia va scădea. Procesul este repetat cât timp gradul deîmpărțitului este mai mare sau egal cu gradul împărțitorului. La finalul acestei operații va fi returnat un obiect de tip RezDiv care conține cele doua polinoame, restul si rezultatul împărțirii.

La finalul oricarei operații pe polinoame are loc o eliminare a termenilor care au valoare ceficientului egala cu 0 din lista de monoame.

Clasa TestPolynomials conține un test, realizat in JUnit 5, al tuturor operațiilor. Clasa conține metoda setUp() care inițializează polinoamele înaintea fiecărei metode de test.

Pentru interfața grafică, clasa View extinde clasa Jframe . In această clasă ca atribute private sunt butoanele responsabile pentru fiecare operație și câteva campuri de text pentru rezultat , rest și evidențierea locațiilor unde vor apărea rezultatele. Frame-ul principal conține un singur Panel, acesta având setat layout-ul de tip GridBagLayout. Utilizând acest tip de layout, panel-ul va fi împarțit in celule, urmând a fi specificata fiecare coordonata a celulei înainte de a se adăuga conținut în aceasta. Pentru fiecare buton sau text adăugat într-o celulă am specificat, unde era cazul, diferite aspecte folosind GridBagConstraints, care permite spațierea față de celulele alăturare, coordonatele celulelor, pe câte rânduri si coloane se poate extinde o singura componentă cât și alinierea unui obiect aflat in celulă. Interfața coține 6 butoane, unul pentru fiecare operație, un buton pentru curățarea câmpurilor de text, 2 câmpuri de text pentru introducerea polinoamelor, 2 etichete pentru afișarea rezultatelor și 4 etichete necesare specificarii locației pentru introducerea și citirea datelor .

**5. Rezultate**

Pentru testarea cu JUnit au fost deja definite doua polinoane in metoda setUp din cadrul clasei TestPolynomials. Pentru veficarea altor polinoame trebuie schimbat atât șirul de caractere care reprezintă polinomul din metoda setUp() cât și șirul de caractere cu care este comparat rezultatul operațiilor din fiecare metodă de test.

Pentru testarea prin interfața grafică se vor introduce 2 polinoame daca se dorește una din opeartâțiile: adunare, scădere, împărțire sau înmulțire , sau un singur polinom daca se dorește una din operațiile: derivare sau integrare. După introducerea polinoamelor se acționează butonul operației corespunzătoare . În cazul introducerii de date invalide va apărea o fereastră care va specifica cauza erorii ( date eroante, câmpurile polinoamelor goale sau ambele câmpuri completate deși se apase pe operatia de derivare sau integrare ) .

**6. Concluzii**

Prin realizarea aceste teme am învățat ce înseamna modelul MVC și mi-am îmbunătățit modul de definire al claselor astfel încât să corespundă cât mai mult unei maniere de programare orientată pe obiecte. De asemenea, am învațat sa folosesc GridBagLayout împreuna cu GridBagConstraints pentru crearea unei interfețe grafice mai complexă.

Am înteles atât importanța tratării tuturor cazurilor posibile și definirea scenariilor cât și importanța unei interfețe grafice cât mai intuitive pentru ușurarea folosirii de către utilizator .

**7. Bibliografie**

* <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>
* <https://www.tutorialspoint.com/java/java_regular_expressions.htm>
* <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>
* <https://stackoverflow.com/>
* <https://google.github.io/styleguide/javaguide.html>
* <https://www.lucidchart.com/>