Calculator de operații cu polinoame

Facultatea de Automatică și Calculatoare

Student: *Petrea Irina-Alexandra*

An 2, grupa 30226

. . . . . . . . . . . . . .

Cuprins:

1. **Obiectivul temei**
   1. **Obiectivul principal**
   2. **Obiective secundare**
2. **Analiza problemei**
3. **Proiectarea soluției**
4. **Implementare**
5. **Rezultate**
6. **Concluzii**
7. **Bibliografie**
8. Obiectivul temei

**Obiectivul principal** al temei este cel de a realiza o aplicație desktop pentru calculul operațiilor cu polinoame cu coeficienți întregi, de o singură variabilă. Aplicația este proiectată conform paradigmelor programării orientate pe obiecte folosind limbajul de programare Java. Punem în evidență mai multe obiective secundare, părți constituente ale obiectivului principal sus-menționat:

* **Divizarea responsabilităților pe clase și metode**, ca bună practică în proiectarea aplicației, prin implementarea modelului arhitectural MVC (Model-View-Controller), pentru a facilita înțelegerea programului și dezvoltarea ulterioară – va fi detaliată în capitolele 3 și 4
* **Realizarea diagramelor UML** (diagrame de clase, use-case, etc), pentru a facilita proiectarea ulterioară și transpunerea funcționalităților în cod – va fi detaliată în capitolele 2 și 3
* Implementarea operațiilor simple cu monoame și polinoame, anume: **adunarea, scăderea, obținerea opusului, integrarea, derivarea** – va fi detaliată în capitolul 4
* Implementarea operațiilor complexe cu monoame și polinoame, anume **înmulțirea și împărțirea** – va fi detaliată în capitolul 4
* **Proiectarea unei interfețe utilizator intuitivă și user-friendly** pentru introducerea operanzilor, selectarea operației și afișarea rezultatului – va fi detaliată în capitolul 4
* **Facilitarea introducerii operanzilor de către utilizator de la tastatură** prin folosirea matching-ului cu expresii regulate – va fi detaliată în capitolul 4.

1. Analiza problemei

Se vor prezenta funcționalitățile oferite de aplicație, precum și detalierea cazurilor de utilizare a acesteia.

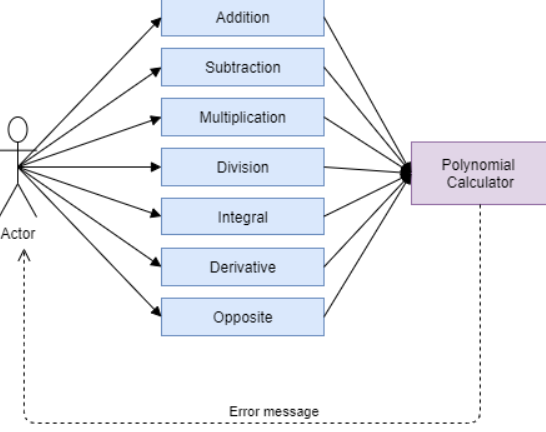
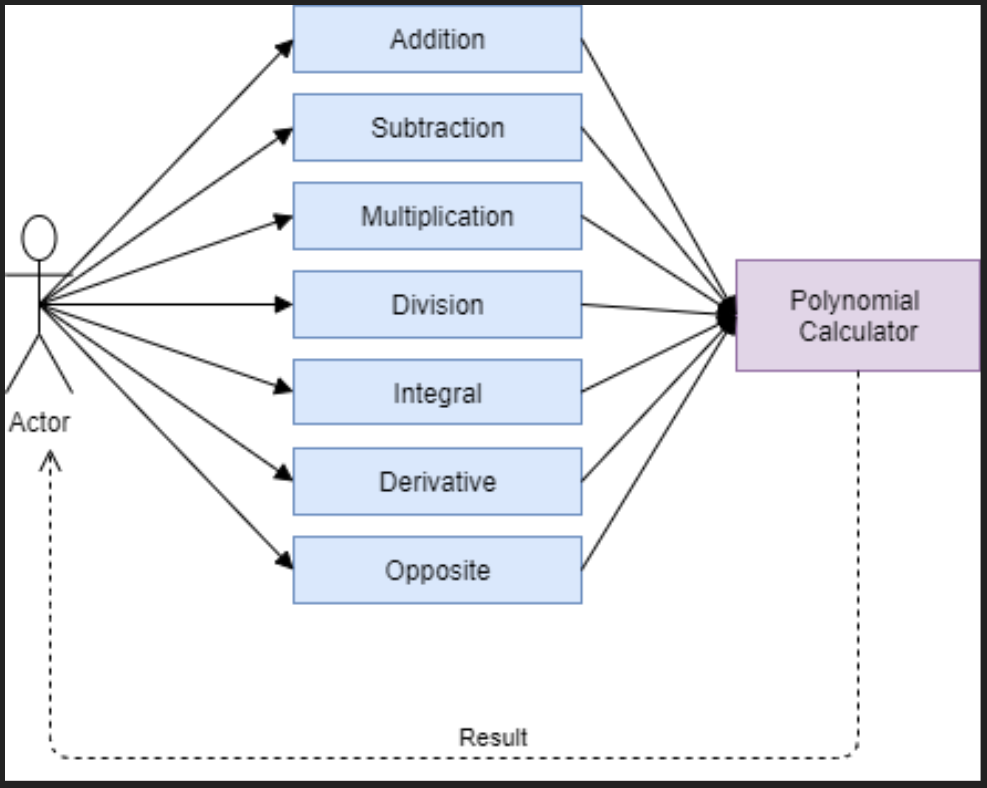
**Lista funcționalităților** întrunește totalitatea operațiilor cu polinoame pe care le conferă aplicația:

* Adunare de polinoame;
* Scădere de polinoame;
* Înmulțire de polinoame;
* Împărțire de polinoame (respectând teorema împărțirii cu rest, conform căreia rezultatul împărțirii a unui polinom P(x) la un alt polinom Q(x) îl reprezintă perechea de polinoame C(x), R(x), astfel încât P(x) = C(x) \* Q(x) + R(x));
* Obținerea integralei nedefinite dintr-un polinom;
* Obținerea formulei derivatei unui polinom;
* Obținerea opusului unui polinom (opusul lui P(x) este – P(x)).

Remarcăm și o serie de funcționalități secundare, inaccesibile utilizatorului, dar care fac posibilă existența funcționalităților principale în cadrul aplicației:

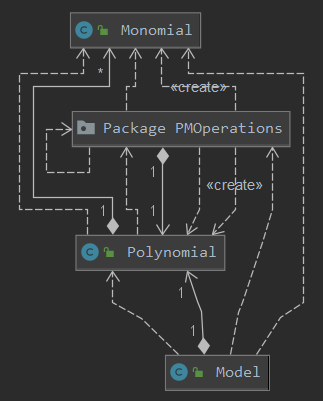
* Efectuarea operațiilor mai sus menționate pe monoame;
* Obținerea monoamelor (pereche coeficient-putere) pentru alcătuirea polinomului, din șirul/șirurile de caractere dat/e ca date de intrare, folosind funcții specializate de potrivire cu expresii regulate (Regular Expressions).

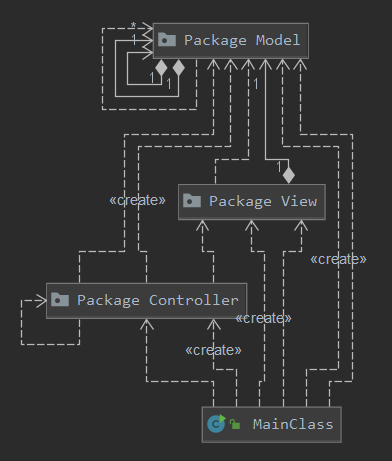
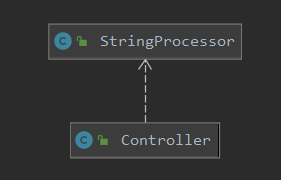
**Cazurile de utilizare** pot fi foarte ușor detaliate după diagramele use-case următoare:

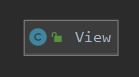


Pentru a putea obține rezultatul unei operații, utilizatorul trebuie să introducă în câmpul/câmpurile aferent/e datelor de intrare, ce pot consta în unul sau două polinoame; apoi, utilizatorul va apăsa butonul aferent operației ce se dorește a fi efectuată. Dacă datele au fost introduse corect, utilizatorul va vedea în câmpul pentru date de ieșire rezultatul operației alese, astfel că diagrama de utilizare este cea din stânga. Dacă datele nu au fost introduse corect sau au fost introduse date nevalide, diagrama use-case arată precum cea din dreapta; utilizatorul primește un mesaj de eroare care îi semnalează faptul că datele introduse sunt nevalide, iar conținutul câmpurilor este șters.

1. Proiectarea soluției

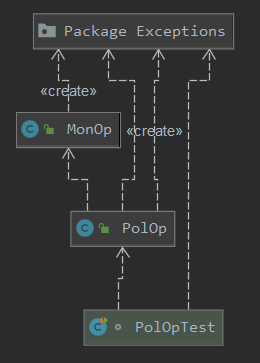
Se vor prezenta diagramele de pachete și de clase ce pun în evidență aspectele de proiectare OOP ale aplicației, precum și justificări legate de alegerile făcute în contextul proiectării.

 **Pachetul Controller**



**Pachetul View**

**Pachetul default** **Pachetul Model**



Pachetul PMOperations

Diagramele de pachete de mai sus înfățișează structura pachetelor importante din implementarea aplicației. Am ales structurarea și implementarea aplicației după **modelul arhitectural MVC – Model-View-Controller**, astfel distribuind responsabilitățile claselor din fiecare pachet după cum urmează:

* **Pachetul Model** – se ocupă cu gestiunea modelului matematic; aici sunt implementate clasele Polynomial, Monomial, precum și clasa propriu-zisă Model.
  + **Pachetul PMOperations** este un subpachet al pachetului Model și conține clase specializate în operațiile cu obiecte de tipurile Polynomial și Monomial. Clasa MonOp se ocupă cu operațiile pe monoame, în timp ce PolOp este o clasă specializată pentru operațiile cu polinoame.
    - **Pachetul Exceptions** conține excepții definite de către proiectant, specifice cazurilor de utilizare ale acestei aplicații
* **Pachetul View** – se ocupă cu aspectul interfeței utilizator, dar nu și cu interacțiunea directă dintre aceasta și utilizator; conține unica clasă View, care îndeplinește atribuțiile menționate mai devreme.
* **Pachetul Controller** – reunește cele două părți, Model și View, ocupându-se de comunicarea dintre cele două; conține clasa Controller, care asigură transmiterea mesajelor între Model și View, precum și clasa specializată StringProcessor, ale cărei atribuții vor fi prezentate în următorul capitol.

**Pachetul default** conține doar clasa MainClass, împreună cu cele 3 pachete principale: Model, View, Controller.

1. Implementare

Se va analiza fiecare clasă în parte, argumentând alegerile de implementare acolo unde este cazul.

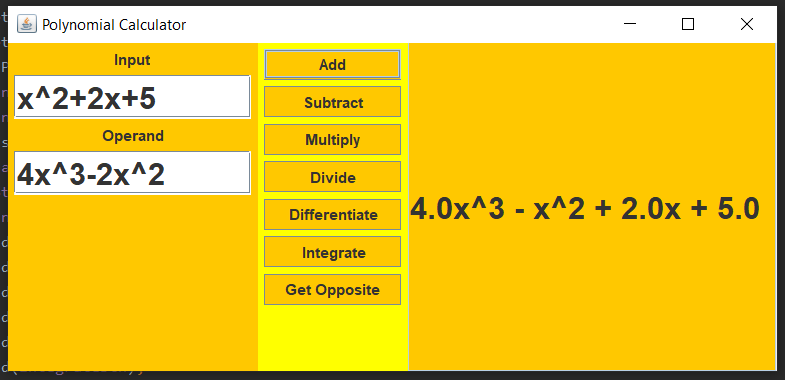
* + - 1. **Pachetul Model**
         * **Clasa Monomial** – este clasa care descrie obiectele de tip monom. Un monom mxn este caracterizat de un coeficient m de tip real și o putere n, de tip întreg.
         * **Clasa Polynomial** – este clasa care descrie obiectele de tip polinom. Un polinom este o sumă de monoame. Am implementat acest lucru folosind o listă înlănțuită de monoame, anume LinkedList, din API-ul Collections al limbajului Java. Pe lângă acest câmp, se păstrează drept proprietate și gradul polinomului, care este, prin definiție, puterea monomului cu puterea cea mai mare din polinom. O metodă importantă este metoda addMonomial, care adaugă un monom în structura polinomului, astfel încât să păstreze o ordine descrescătoare a puterilor monoamelor. Acest lucru se face iterând o dată prin structura de date, deci în timp liniar. Mai mult, dacă se dorește adăugarea unui monom la o putere deja existentă în polinom, această funcție se asigură că fiecare putere apare o singură dată, adunând monoamele de aceeași putere.
         * **Clasa Model** – folosește cele două clase Polynomial și Monomial, precum și pachetul pentru operații pe monoame și polinoame PMOperations, pentru a simula utilizarea aplicației: efectuarea de operații pe unul sau două polinoame. Ca variabile instanță avem rezultatul curent (currentTotal), un polinom, la care se poate aduna, scădea etc și polinomul aux, care se folosește în cazul împărțirii de polinoame, care poate avea 2 rezultate de tip polinom.
  + **Pachetul PMOperations** – este important să menționăm că metodele din clasele acestui pachet sunt statice, astfel nefiind nevoia instanțierii de obiecte pentru a efectua operațiile dorite. Clasele acceptă unul sau doi operanzi în funcție de operație, returnând de regulă un operand de tipul Monom, respectiv Polinom, cu excepția operației de împărțire de polinoame, unde se returnează un vector de 2 polinoame.
    - **Clasa MonOp** – clasă specializată pentru efectuarea de operații cu monoame: adunare, scădere, înmulțire, împărțire, derivare, integrare; clasa oferă și metode auxiliare, precum: obținerea opusului, verificarea de zero sau pozitivitate, copierea deep-copy, precum și generarea polinomului zero;
    - **Clasa PolOp** – clasă specializată pentru efectuarea de operații cu polinoame: adunare, scădere, înmulțire, împărțire, derivare, integrare; clasa oferă și metode auxiliare, precum: obținerea opusului, verificarea de zero, generarea polinomului zero. Metoda de adunare a fost implementată ca o interclasare, întrucât ne putem baza mereu pe ordonarea descrescătoare a polinoamelor în funcție de putere. Metoda de scădere a fost implementată foarte simplu, ca o adunare între primul operand și opusul celui celui de-al doilea operand. Înmulțirea s-a făcut înmulțind fiecare monom din primul operand cu fiecare monom din al doilea operand, adăugând rezultatul înmulțirii monoamelor la rezultatul final; din nou, ne bazăm pe abilitatea funcției addMonomial din clasa Polynomial de a menține ordinea descrescătoare a puterilor în lista înlănțuită de monoame și de a păstra unicitatea puterilor prin adunarea termenilor asemenea (prin termeni asemenea înțelegem monoame cu aceeași putere). Împărțirea polinoamelor s-a realizat după algoritmul din [1], prin adaptarea pseudocodului la limbajul Java și la structurile de date, metodele folosite. Împărțirea se face conform teoremei împărțirii cu rest a două polinoame, returnând un vector de alte două polinoame, quotient și remainder, care respectă relația p1 = p2\*quotient + remainder, unde p1 și p2 sunt operanzii împărțirii, iar operația dorită este p1/p2. Operațiile de derivare și integrare se fac pe baza proprietății de liniaritate a acestor două operații: derivarea/integrarea sumei este egală cu suma derivatelor/integralelor, deci rezultatul se construiește prin parcurgerea listei de monoame și adăugarea la rezultat a monoamelor pe care se efectuează operația dorită, de derivare sau integrare. In această clasă sunt incluse și supraîncărcări de metode precum cea de înmulțire, putându-se efectua înmulțirea cu scalar, cu monom și cu polinom, cu același nume de metodă.
    - **Clasa OperationException** din pachetul Exceptions este o exceptie definita de catre proiectant, pentru a putea depista anumite erori in efectuarea operatiilor, precum: impartirea la monomul sau polinomul nul, impartirea unui polinom cu grad mic la un polinom cu grad mai mare, etc.

2. **Pachetul Controller**

* + - **Clasa Controller** – această clasă întrunește componentele Model și View, facilitând comunicarea dintre ele. Doar prin constructor, această clasă definește ascultători pentru componentele din View cu care interacționează utilizatorul (butoanele ce vor declanșa efectuarea operației), iar pe baza acestor interacțiuni va apela anumite metode din Model, pentru a putea întoarce și afișa un rezultat în View. În acest fel, clasa Model, care implementează modelul matematic, nu știe de prezența clasei View.
    - **Clasa StringProcessor** – această clasă conține metode specializate pentru procesarea șirurilor de caractere, fiind capabile să extragă polinoame din șiruri de caractere, pe baza unor tiparuri, cu ajutorul expresiilor regulate (Regular Expressions sau RegEx). Metodele de procesare sunt statice, ca să nu fim nevoiți să instanțiem un obiect din această clasă pentru a procesa șirurile de caractere. Metoda principală o reprezintă metoda toPolynomial, care acceptă ca parametru un șir de caractere (String), urmând să returneze un obiect de tipul Polynomial, corespunzător șirului de caractere dat. Ca parte din preprocesarea șirului de caractere, se elimină orice spațiu alb din interiorul acestuia. Șirul de caractere de intrare este așteptat să respecte un tipar de genul: sumă de polinoame de forma mxn, xn, mx, x, m, unde m este număr întreg, iar n este număr natural. Prin matching cu o expresie regulată, se identifică 5 tipuri de grupuri (captured groups, cu paranteze rotunde, ca în [2]), corespunzătoare monoamelor de tipurile mxn, xn, mx, x, m. Aceste grupuri se capturează cu semn, urmând apoi să extragem semnul (prin variabila sign, care poate fi 1 sau -1, extragerea se face tot pe baza unui matching cu o expresie regulată), apoi să extragem coeficienții. Extragerea coeficienților se face după aducerea fiecărui grup (monom) la forma mxn (de exemplu, 2x va fi 2x1, x va fi 1x1, iar constanta 10 va fi 10x0). După extragerea coeficienților și a semnului, se poate instanția un obiect de tipul Monomial, care va fi adăugat la rezultatul de tip Polynomial. Drept documentație și unealtă de test, am folosit [3], respectiv [4].

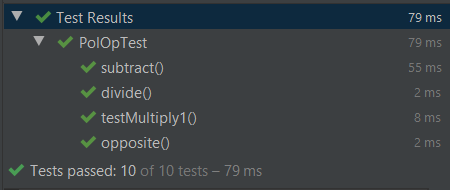
3. **Pachetul View**

* **Clasa View** – această clasă se ocupă cu interfața grafică a aplicației, cea cu care interacționează utilizatorul. Utilizatorului i se pun la dispoziție două zone pentru introducere text (introducerea operanzilor), precum și butoane, la acționarea cărora se va efectua operația aferentă. Dacă operația dorită este ilegală (conform descrierii clasei PolOp), utilizatorului i se va afișa un mesaj de eroare, iar casetele de text își vor șterge conținutul. Altfel, dacă operația este legală, în zona de date de ieșire va fi afișat rezultatul operației. Dacă operația dorită are nevoie de un singur operand (derivarea, integrarea, calcularea opusului), atunci se va lua în considerare doar ceea ce este scris în prima casetă de text.



1. Teste și rezultate

Pentru testarea acestei aplicații, s-a folosit unealta de testare unitară **JUnit**. S-a testat numai clasa de operații pe polinoame PolOp, întrucât în testarea acesteia au fost implicate toate celelate metode importante (procesarea de șiruri de caractere din clasa StringProcessor, precum și operațiile pe monoame din clasa MonOp).



1. Concluzii

În urma realizării acestei teme, am dobândit cunoștințe cu privire la utilizarea expresiilor regulate în prelucrarea de șiruri de caractere. De asemenea, am aprofundat cunoștințele legate de modelul arhitectural Model-View-Controller, metodele de testare unitară folosind JUnit, realizarea diagramelor UML ca ajutor în proiectarea aplicației, precum și realizarea unei documentații cuprinzătoare.

Drept posibilități ulterioare de dezvoltare, la această aplicație se pot adăuga validatori, astfel încât utilizatorul să fie atenționat exact unde a greșit (de exemplu, o greșeală de scriere). De asemenea, se poate dezvolta aplicația așa încât să permită operațiile cu polinoame având coeficienți reali. Se mai poate adăuga opțiunea de găsire a rădăcinilor polinomului, sau de calcul al unor transformate (Laplace, Fourier etc)

1. Bibliografie

[1] <https://rosettacode.org/wiki/Polynomial_long_division>

[2] <http://courses.washington.edu/css162/stark/Polynomial/Polynomial.java>

[3] <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>

[4] <https://regex101.com/>