

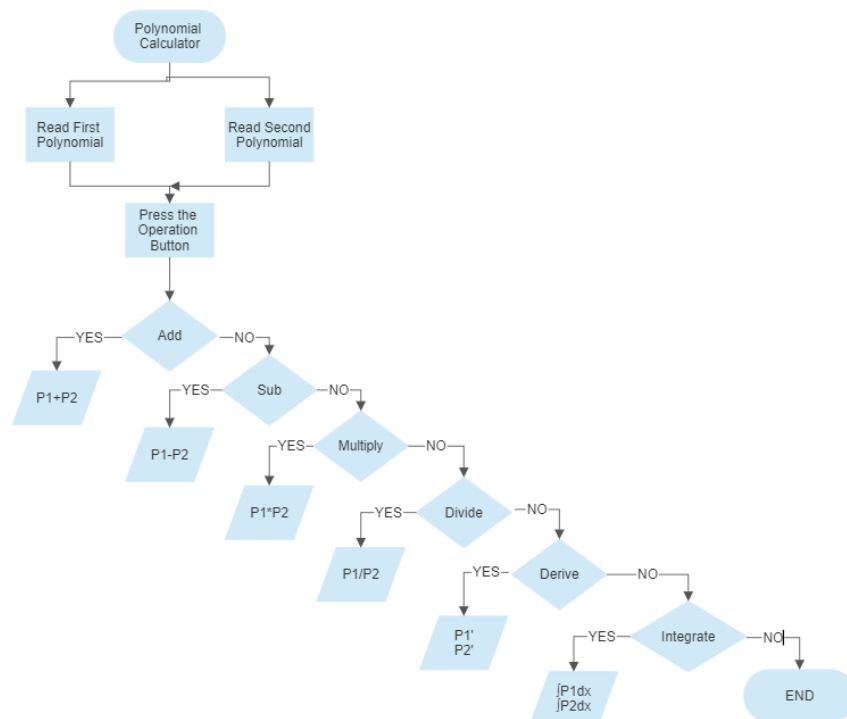
# DOCUMENTATIE

## TEMA 1

NUME STUDENT: Ciofu Dinuța-Dumitrița  
GRUPA: 30221

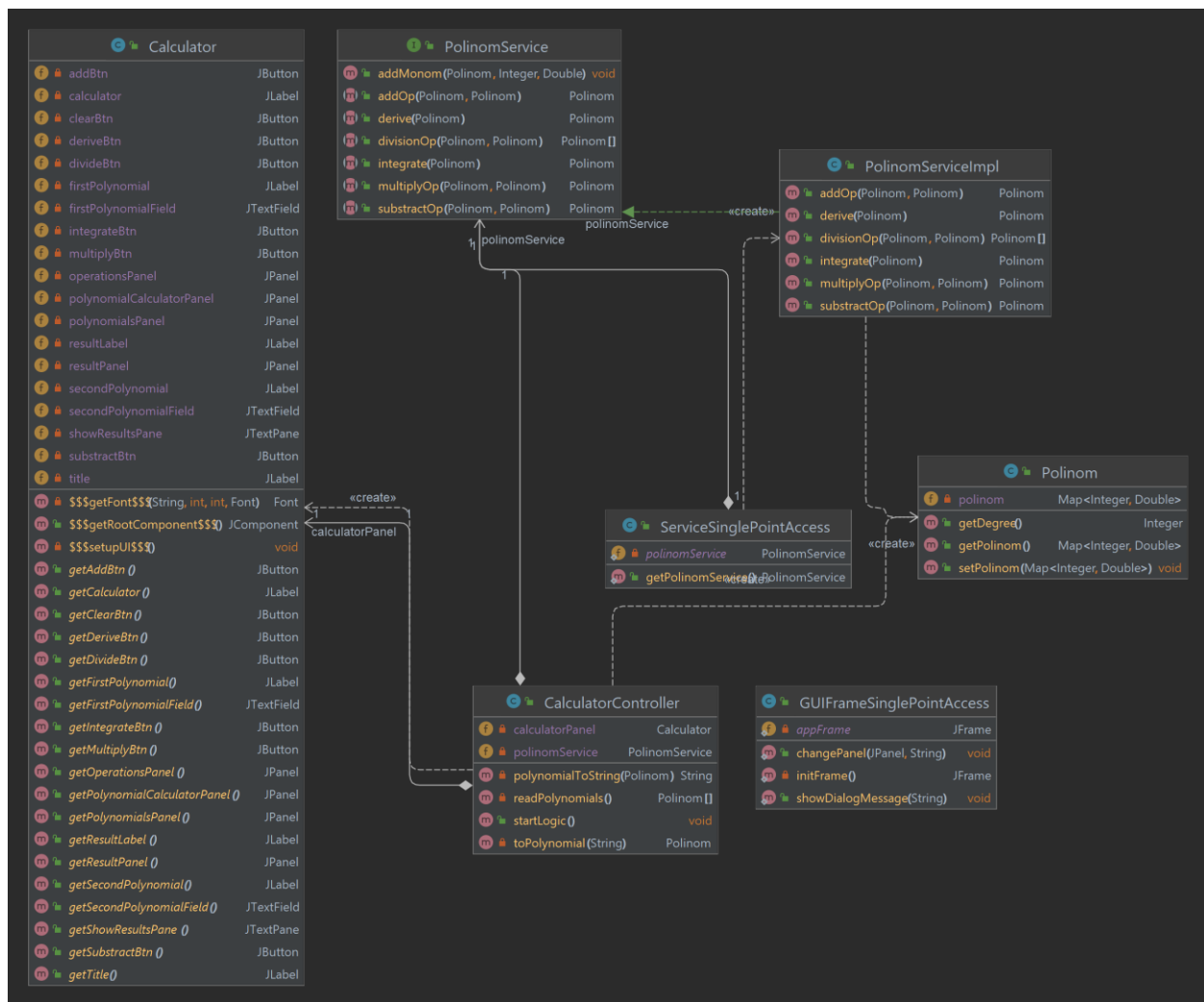
# CUPRINS

1.	Obiectivul temei.....	3
2.	Analiza problemei, modelare, scenarii, cazuri de utilizare .....	3
3.	Proiectare .....	4
4.	Implementare .....	5
5.	Rezultate .....	6
6.	Concluzii.....	8
7.	Bibliografie .....	9



### 3. Proiectare

UML (Unified Modeling Language) este un limbaj standard de modelare a sistemelor de software, utilizat în mod obișnuit pentru a reprezenta grafic structura, comportamentul și interacțiunile dintre diferite componente ale unui sistem.



- Clasa „Polinom”: are o singură variabilă membru, polinom. Polinoamele sunt stocate ca o mapare între exponenții termenilor și coeficienții lor. De exemplu, polinomul  $x^3 + 2x^2 - 3x^1 + 4$  va fi reprezentat ca o mapare  $\{3 \rightarrow 1, 2 \rightarrow 2, 1 \rightarrow -3, 0 \rightarrow 4\}$ . Această clasă conține câteva metode de acces pentru a accesa și manipula polinoamele.

- Interfața „PolinomService”: definește operațiile ce se pot face cu polinoamele, și anume adunare, scădere, înmulțire, împărțire, derivare și integrare. Aceste operații sunt implementate în clasa „PolinomServiceImpl”, care implementează interfața PolinomService.
- Pe lângă acestea, aplicația include și o clasă „GUIFrameSinglePointAccess”, care oferă o interfață grafică de utilizator pentru a interacționa cu aplicația.
- Clasa „ServiceSinglePoinAccess” are o singură metodă statică (getPolinomService) care returnează instanța clasei „PolinomService”.
- Clasa „Calculator” este responsabilă de crearea interfeței vizuale a calculatorului de polinoame. Aici se găsesc elemente de interfață grafică și metodele care le returnează.
- „CalculatorController”: această clasă conține logica principală de polinoame și este responsabilă de comunicarea dintre model (app.model.Polinom), service (app.service.PolinomService) și view (app.view.Calculator). În această clasă de găsesc metode care transformă un șir de caractere într-un polinom (toPolynomial), care citește și returnează cele două polinoame introduse de utilizator (readPolynomials), precum și o metodă publică (startLogic) care pornește interacțiunea utilizatorului cu calculatorul de polinoame.

## 4. Implementare

Pentru a implementa acest calculator, s-a creat o clasă CalculatorController, care se ocupă de gestionarea interacțiunii utilizatorului cu interfața și de efectuarea operațiilor aritmetice cu polinoame. Această clasă conține câteva câmpuri de referință, precum un obiect de tip Calculator (care reprezintă interfața grafică a calculatorului), un serviciu PolinomService (care se ocupă de efectuarea operațiilor cu polinoame), precum și metode pentru conversia între string-uri și obiecte de tip polinom.

Metoda „toPolynomial(String s)” primește un șir de caractere și îl convertește într-un obiect de tip Polinom, apoi adaugă fiecare monom în obiectul respectiv, folosind metoda „addMonom()” din PolinomService. Această metodă folosește o expresie regulată pentru a separa fiecare monom din șirul de caractere și o serie de condiții pentru a identifica coeficientul și exponentul fiecărui monom.

Metoda „polynomialToString(Polinom p)” primește un obiect de tip Polinom și îl convertește într-un șir de caractere, apoi îl returnează. Această metodă parcurge mapa de monomi ai polinomului și adaugă fiecare monom în șirul de caractere, în funcție de coeficientul și exponentul acestuia.

Metoda „readPolynomyals()” primește valorile introduse de utilizator în cele două câmpuri de text, apoi le convertește în obiecte de tip Polinom, folosind metoda „toPolynomial()”.

Metoda „startLogic()” este metoda principală a clasei CalculatorController. Aceasta inițializează interfața grafică a calculatorului, setează ascultători de evenimente pentru butoanele de adunare, scădere, înmulțire, împărțire, derivare, integrare și clear, apoi afișează rezultatul într-un câmp de text din interfață. Metoda afișează o casetă de dialog în cazul în care utilizatorul introduce date invalide.

## 5. Rezultate


Această implementare conține un set de teste JUnit pentru a verifica dacă operațiile polinomiale din PolinimService funcționează așa cum se așteaptă. JUnit este o bibliotecă de testare pentru Java, care permite dezvoltatorilor să creeze și să ruleze automat teste unitare pentru a verifica dacă codul lor este corect.

În clasa OperationsTest, se folosesc șase teste de operații: adunarea, scăderea, înmulțirea, împărțirea, derivarea și integrarea a două polinoame. În fiecare test se creează două polinoame, se efectuează operația specificată și se verifică dacă rezultatul este cel așteptat.

Testarea automată este importantă în dezvoltarea software-ului deoarece permite dezvoltatorilor să descopere rapid erori și bug-uri în codul lor. Prin adăugarea de teste de unitate la codul lor, aceștia pot fi siguri că modificările pe care le fac nu rup funcționalitatea existentă a aplicației. De asemenea, teste bine scrise permit dezvoltatorilor să lucreze cu mai multă încredere la refactorizarea codului sau la adăugarea de funcționalități noi, știind că au un set de teste care să le garanteze că nimic nu a fost stricat.

În plus, pe lângă testarea unitară, utilizatorul poate testa direct operațiile prin intermediul interfeței grafice.

ADUNARE



POLYNOMIAL CALCULATOR

First Polynimial

Second Polynomial

Add Subtract


Multiply Divide

Derive Integrate

Result:  $+3.0x^3-7.0x^2+3.0x^1-2.0$

Clear

ÎMPĂRȚIRE



POLYNOMIAL CALCULATOR

First Polynimial

Second Polynomial

Add Subtract

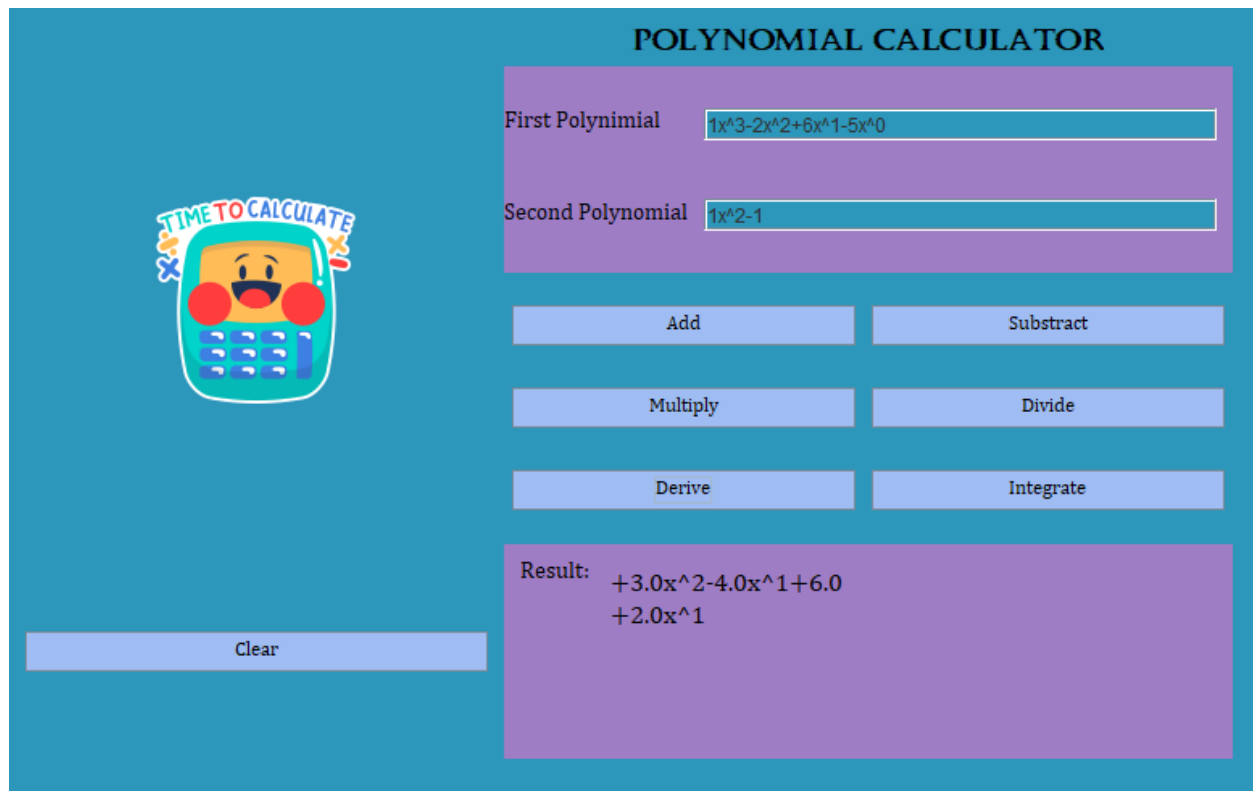
Multiply Divide

Derive Integrate

Result:  $+1.0x^1-2.0$   
 $+7.0x^1-7.0$

Clear

## DERIVARE



The screenshot shows a web-based application titled "POLYNOMIAL CALCULATOR". On the left, there is a cartoon calculator character with a face and the text "TIME TO CALCULATE" above it. The main interface has a purple background. It features two input fields: "First Polynimial" (note the typo) containing the expression  $1x^3-2x^2+6x^1-5x^0$  and "Second Polynomial" containing  $1x^2-1$ . Below these are four buttons: "Add", "Subtract", "Multiply", and "Divide". At the bottom of the button section are "Derive" and "Integrate". A "Clear" button is located on the left side. The "Result:" section displays the output:  $+3.0x^2-4.0x^1+6.0$  and  $+2.0x^1$ .

## 6. Concluzii

În concluzie, aplicația de calculator de polinoame dezvoltată în cadrul acestei teme de laborator oferă utilizatorului posibilitatea de a efectua operații aritmetice cu două polinoame și operații matematice precum derivare și integrare. Polinoamele sunt introduse în interfața grafică a aplicației și interpretate utilizând expresii regulate și o structură de date bazată pe TreeMap. Aplicația are o arhitectură orientată pe obiecte și include mai multe clase, cum ar fi Polinom, PolinomService, GUIFrameSinglePointAccess și CalculatorController. Scopul final al aplicației este de a oferi o interfață simplă și intuitivă pentru utilizatori, astfel încât să se poată efectua operațiile dorite cu ușurință.

Totodată, în cadrul temei "calculator de polinoame", am învățat cum să implementez o aplicație de calcul al polinoamelor, care oferă utilizatorului posibilitatea de a efectua operații aritmetice cu două polinoame și operații matematice precum derivare și integrare. În general, implementarea acestei aplicații ne-a ajutat să îmbunătățim abilitățile de programare orientată pe obiecte, să înțelegem cum să implementăm o interfață grafică simplă și intuitivă pentru utilizator și să învățăm cum să utilizăm expresiile regulate și structurile de date pentru a interpreta și manipula polinoame.



## 7. Bibliografie

1. *Regex for polynomial expression* - <https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression>
2. *JUnit* - <https://mvnrepository.com/artifact/junit/junit/4.13.2>
3. *JUnit 5 User Guide* - <https://junit.org/junit5/docs/5.0.2/user-guide/index.pdf>