DOCUMENTAȚIE

TEMA 1



NUME STUDENT: Sîrbu Cosmin-Claudiu

GRUPA: 30221(2), CTI, seria A

# CUPRINS

1. Obiectivul temei………………………………………………………………………………..3

2. Analiza problemei, modelare, scenarii, cazuri de utilizare …………………………………….5

3. Proiectare……………………………………………………………………………………….8

4. Implementare…………………………………………………………………………………...9

5. Rezultate………………………………………………………………………………………11

6. Concluzii………………………………………………………………………………………11

7. Bibliografie ……………………………………………………………………………………12

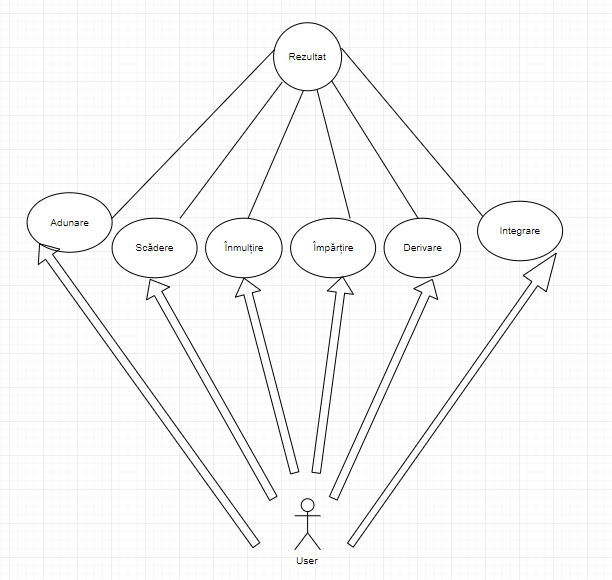
# Obiectivul temei

Tema dată are ca obiectiv crearea unui calculator de polinoame. Pentru dezvoltarea și implementarea funcțiilor necesare realizării acestui prooiect, s-a folosit limbajul de programare Java, folosind IDE-ul IntelliJ Idea de la NetBrains. Calculatorul de polinoame realizează operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare. Pentru a putea fi folosit acest calculator, este implementată o interfață GUI, prin care utilizatorul poate interacționa cu programul. Aceste indicații constituie obiectivul principal al proiectului.

Obiectivele secundare vor fi prezentate în tabelul de mai jos:

|  |  |  |
| --- | --- | --- |
| ***Obiectiv secundar*** | ***Descriere*** | ***Capitol*** |
| Implementarea clasei Polinom | În această clasă s-a declarat polinomul. Modelul/structura unui polinom, care va fi folosit în operațiile potrivite. | 3, 4 |
| Implementarea clasei Monom | Prin această clasă se crează monoame pentru realizarea polinomului. | 3, 4 |
| Implementarea Operațiilor prin clasa Operații | Implementarea operațiilor de: adunare, scădere, înmulțire, împărțire, derivare, integrare. | 4 |
| Implementarea unei Interfețe GUI | Prin interfața GUI se realizează comunicarea dintre utilizator și program: o fereastră interactivă specifică unui calculator pentru polinoame. | 4 |
| Implementarea unui Controller  ***Obiectiv secundar*** | Realizează legătura dintre GUI și Operații.  ***Descriere*** | 4  ***Capitol*** |
| Implementarea de Teste | Testare pentru verificarea operațiilor implementate. | 5 |
| Implementarea Main-ului | Rularea obiectivului principal | 1 |

# Analiza problemei, modelare, scenarii, cazuri de utilizare

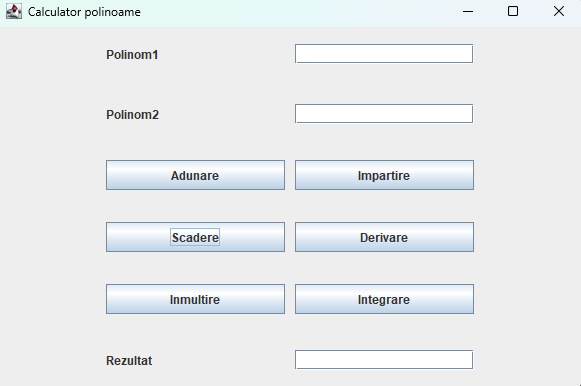
**

**Cerințe funcționale:**

1. Interfața implementată, care face legătura dintre utilizator și program permite accesul introducerii de la tastatură a două polinoame și returnarea unui alt polinom în funcție de operația aleasă.
2. Prin butoanele incluse în interfață, utilizatorul poate alege una din cele 6 operații.
3. Polinoamele sunt stocate cu ajutorul Map-urilor<Integer, Monom> .
4. Clasele implementate nu depășesc 300 de linii de cod.
5. Metodele nu depășesc 30 de linii.
6. Interfața grafică s-a realizat prin Java Swing.
7. Pentru parcurgerea colecțiilor de date s-a folosit foreach in loc de instrcuțiunea obișnuită de for.
8. Este utilizat un regex pentru alcătuirea unui polinom.
9. Pentru testarea operațiilor s-a folosit JUnit.

**Liste pași:**

1. Prin intermediul interfeței GUI utilizatorul are posibilitatea de a introduce de la tastatură două polinoame în cele 2 Text-Field-uri (Polinom 1/2). Polinoamele citite de la tastatură sunt salvate în Map.
2. Apoi utilizatorul trebuie să selecteze din interfața grafică, operația pe care dorește să o execute. Aplicația execută automat operația în funcție de butonul ales. In al 3-lea Text-Field, utilizatorul poate urmări rezultatul operației menționate anterior. Rezultatul este afișat prin intermediul funcției display() a programului. Acesta nu se oprește aici, ci la dorința utilizatorului, se poate continua prin introducerea altor polinoame, sau realizarea altor operații pe expresiile care deja au fost introduce de la tastatură. Aplicația care a fost rulată prin intermediul clasei ”Main”, se va opri din executat în momentul în care utilizatorul va închide fereastra de interfață.
3. Polinoamele au fost salvate si modelate prin intermediul clasei publice ”Polinom”, care folosește un Map format dintr-un întreg și un obiect de tip Monom <Integer, Monom>. În acest fel, fiecare element al polinomului, poate fi tratat o singură dată.
4. Clasele care au fost implementate nu depășesc niciuna numărul maxim admis de 300 de cod. Clasa cu cel mai mare număr de linii de cod este clasa ”Operații.java” care are o limită de 117 linii.
5. Metodele au fost implementate citeț și ușor de înțeles, niciuna din ele nedepășind 30 de linii.
6. S-a folosit Java Swing pentru crearea interfeței grafice. Această interfață folosește 3 Text-Field-uri, 6 Butoane și un JPanel fără de care nu s-ar putea folosi interfața. Java Swing-ul permite crearea de aplicații pentru Desktop, prin care utilizatorul interacționează cu mediul de lucru. Interfața arată în felul următor:



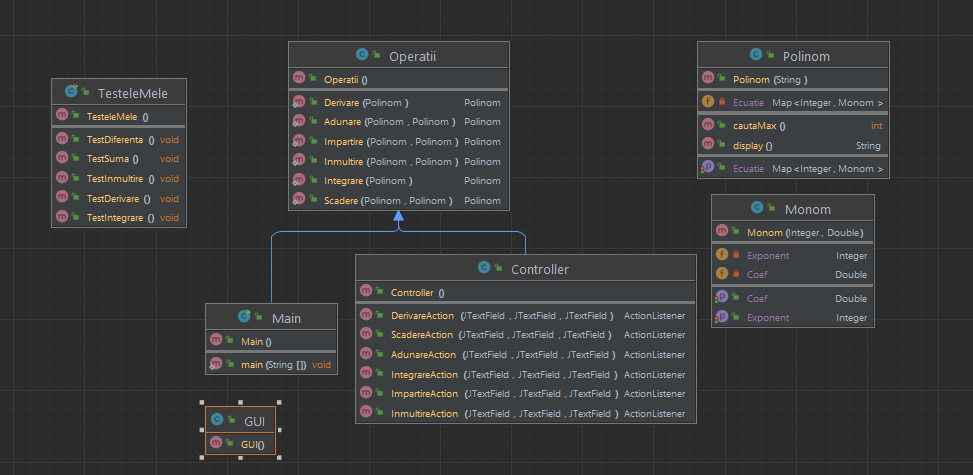
1. Pentru o tehnică de programare mai eficientă s-a preferat utilizarea instrucțiunii de buclă foreach. Codul devine mai lizibil astfel.
2. Regex-ul de tip Pattern folosit este următorul: (([-+])?\\d+\\.?\\d)+x(\\^(\\d+))?.
3. Un instrument popular pentru testarea codului și asigurarea funcționalității corecte a aplicațiilor este JUnit. Această librărie de testare unitară pentru Java permite dezvoltatorilor să scrie teste automate, detectând erorile în timpul dezvoltării. Această abordare ajută la îmbunătățirea calității codului și reduce riscul de apariție a bug-urilor în aplicație.

**Cerințe non-funcționale:**

1. Codul nu este funcțional pentru realizarea împărțirii deoarece se intră în buclă infinită și programul nu poate face față solicitării. În momentul execuției operației de împărțire, interfața se blochează și nu se mai poate face niciun calcul. O posibilă eroare a acestui cod este că la declararea Map-ului a fost folosită o structură dintr-un Integer si un Monom, în loc de un Integer și un Double.

# Proiectare

Diagrama UML a proiectului:



**Structuri de date folosite:**

Pentru a eficientiza stocarea și memorarea polinoamelor, s-a ales să se utilizeze hashmap-ul. Această abordare presupune salvarea fiecărui monom al polinomului într-o pereche de date, formată dintr-un exponent reprezentat de un număr întreg și un monom. Prin intermediul hashmap-ului, accesul rapid la informațiile necesare devine posibil, ceea ce permite manipularea polinoamelor într-un mod eficient. Utilizatorii pot, de asemenea, interacționa cu acestea cu ușurință și pot efectua operațiuni complexe fără dificultăți semnificative. Acest lucru se datorează faptului că numărul întreg din perechea <Integer,Monom> reprezintă exponentul și permite o gestionare intuitivă a polinoamelor.

**Interfețe:**

În proiectul propus, s-a decis să se folosească o interfață pentru implementarea metodelor care se ocupă de operațiile cu polinoame, în scopul de a obține o mai mare flexibilitate și adaptabilitate la nevoile proiectului. Această abordare ne permite să schimbăm implementarea algoritmilor pentru operațiile cu polinoame, fără a afecta structura fundamentală a programului. În acest fel, putem înlocui doar implementarea corespunzătoare din cadrul interfeței dacă găsim un algoritm mai eficient sau mai potrivit pentru realizarea anumitor operații cu polinoame, fără a fi nevoie să modificăm întregul program. Această metodă ne permite să ne concentrăm pe dezvoltarea algoritmilor și a operațiilor cu polinoame, în timp ce menținem structura generală a programului neschimbată. Prin urmare, utilizarea unei interfețe pentru operațiile cu polinoame ne ajută să menținem un program eficient și ușor de administrat.

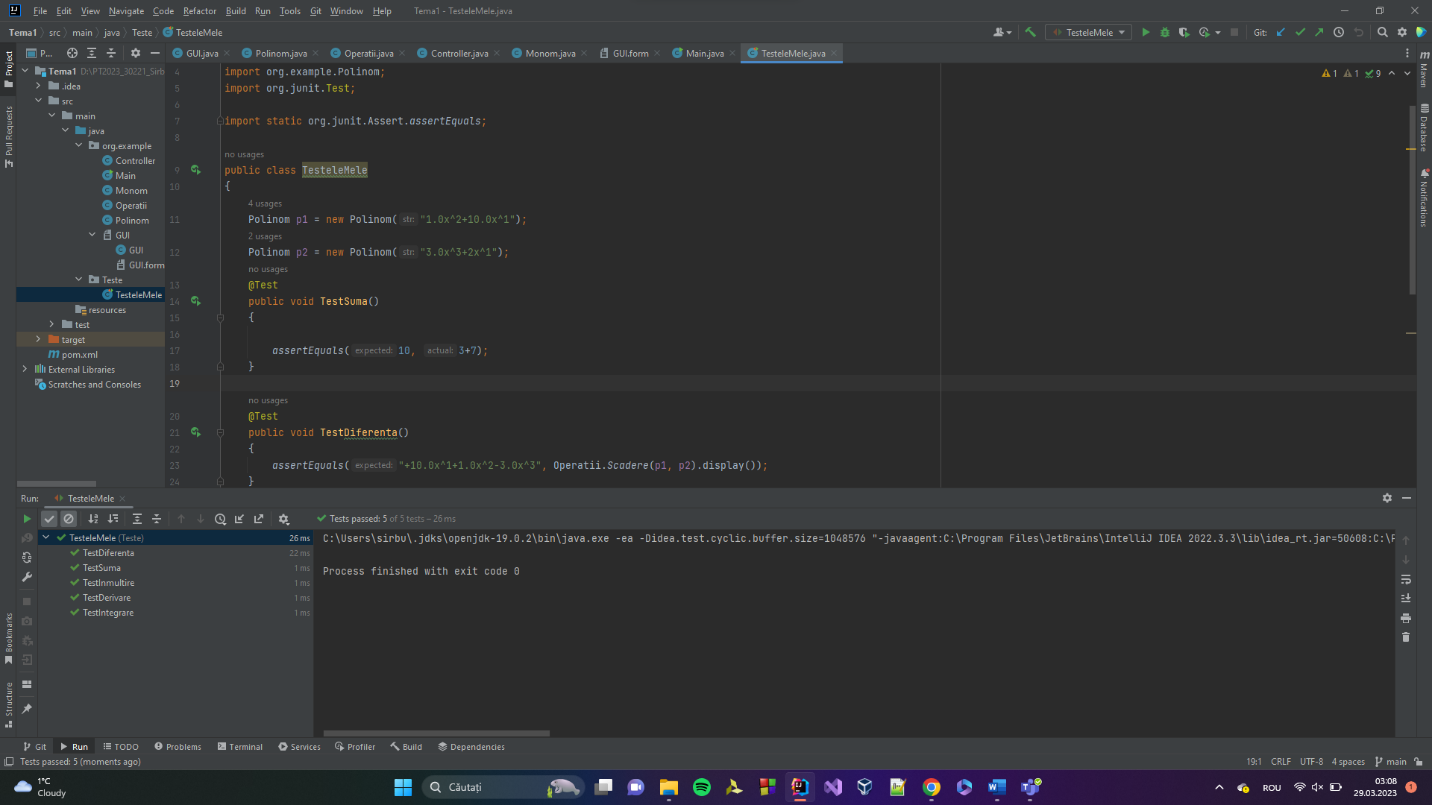
**Algoritmii folosiți:**

În cadrul proiectului nostru, am dezvoltat algoritmi critici pentru operațiile cu polinoame, precum adunarea, scăderea, înmulțirea, împărțirea, derivarea și integrarea. Deși acești algoritmi sunt de bază, aceștia sunt esențiali pentru orice program care operează cu polinoame și sunt necesari pentru realizarea unor operații mai complexe și avansate. Implementarea acestor algoritmi ne-a oferit o funcționalitate robustă și eficientă pentru operațiile de bază cu polinoame. Mai mult decât atât, acești algoritmi reprezintă o bază solidă pentru dezvoltarea unor algoritmi mai avansați în viitor, care să permită realizarea unor operații mai complexe cu polinoame. În acest fel, acești algoritmi critici reprezintă un pas important în dezvoltarea programului nostru de operare cu polinoame și sunt fundamentul pentru dezvoltarea de operații mai complexe în viitor.

# Implementare

1. Clasa “Main” extinde clasa “Operații”. Prin rularea ei are loc deschiderea interfeței cu care va lucre utilizatorul.
2. Clasa “Monom” este compusă din 2 obiecte: Exponent (de tip Integer) și Coef (de tip Double). Pentru fiecare obiect s-au creat getter-e si setter-e, care sa ajute la obtinerea dimensiunilor monomului, la cerea mai multor operații.
3. În cadrul clasei "Polinom", avem un singur câmp denumit "polinom", care este de tipul Map<Integer, Monom>. Acesta este utilizat pentru a stoca monomele care alcătuiesc polinomul. Fiecare monom este reprezentat printr-o pereche cheie-valoare formată din exponentul monomului, reprezentat de un număr întreg, și coeficientul monomului, reprezentat de un obiect de tipul Monom. Această abordare facilitează stocarea și accesarea monomelor în funcție de exponentul lor. Clasa "Polinom" are un constructor care primește un șir de caractere reprezentând un polinom. Acesta convertește șirul într-un map cu perechi cheie-valoare de tipul exponent-monom și oferă metode pentru a accesa și manipula acest map. În plus, clasa oferă metode pentru a returna un șir de caractere reprezentând polinomul, a găsi monomul cu cel mai mare exponent și a returna un șir de caractere reprezentând polinomul cu coeficienții pozitivi și exponentele pozitive. Aceste metode facilitează manipularea polinoamelor și permit utilizatorilor să acceseze informații importante despre polinoame într-un mod convenabil și eficient.
4. Clasa “GUI” este creată pentru a realiza interfața aplicației pentru calculatorul de polinoame. Această interfață a fost deja afișată și prezentată în rândurile de mai sus.
5. Clasa “Operații”, a fost implementată cu scopul de a programa operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare, pentru polinoamele pe care dorm isa le calculăm.
6. Clasa “Controller” este o clasă care face legătura dintre GUI si Operații (clasă pe care o extinde). Aici sunt implementate mai multe metode care returnează obiecte de tipul ActionListener, care fac legătura propriu-zisă program-interfață.

# Rezultate

  
 Pentru a ne asigura că aplicația noastră funcționează corect și îndeplinește specificațiile, am realizat mai multe teste utilizând utilitarul JUnit, care permite testarea unitară a codului Java. Am creat scenarii de testare pentru fiecare metodă specifică din aplicație și am verificat comportamentul acestora în diferite situații. Toate testele au trecut cu succes, ceea ce ne indică faptul că metodele implementate funcționează corespunzător, respectând specificațiile și că aplicația noastră este capabilă să execute operațiile pe polinoamele dorite. Aceste teste ne-au oferit încredere în funcționarea aplicației noastre și ne-au permis să o livrăm cu încredere către utilizatori.

# Concluzii

După finalizarea proiectului de dezvoltare a aplicației pentru calculul polinoamelor, s-au făcut o serie de concluzii importante și s-au identificat posibile îmbunătățiri. A fost constatat că specificațiile clare și respectarea cerințelor non-funcționale sunt esențiale pentru a dezvolta eficient și pentru a asigura o experiență de utilizare de calitate. Abordarea modulară și bine structurată a codului a permis dezvoltarea rapidă a funcționalităților necesare.

De asemenea, utilizarea framework-ului JUnit a fost foarte utilă pentru testarea riguroasă și automatizată a codului, astfel încât funcționalitățile implementate să fie conforme cu specificațiile.

În general, s-a înțeles că programarea modulară, testarea riguroasă și respectarea specificațiilor și cerințelor non-funcționale sunt cruciale în dezvoltarea unei aplicații de calitate.

O posibilitate de îmbunătățire ar fi adăugarea unui sistem de sugestii de completare a polinoamelor în timp real. Acest sistem ar fi capabil să sugereze completări valide pentru polinoame în timp ce utilizatorul le introduce, făcând procesul de introducere a datelor mai rapid și mai ușor de realizat. De asemenea, se poate adăuga și un sistem de notificare sau de avertizare pentru cazurile în care utilizatorul introduce date invalide sau incomplete, ajutând astfel la prevenirea erorilor și la îmbunătățirea experienței de utilizare.

# Bibliografie

<https://www.youtube.com/watch?v=9B9ygPrq8zI&t=383s>

*"Java Programming for Beginners" de Mark Lassoff*

<https://dsrl.eu/courses/pt/>