**Documentatie pentru tema 1 la materia**

**Tehnici de programare fundamentale**

**1.Obiectivul temei:**

Obiectivul principal al acestei teme constă în elaborarea unui calculator polinomial în Java prin intermediul căruia se pot realiza diferite operații matematice pe unul sau două polinoame, interacțiunea cu utilizatorul realizându-se prin intermediul unei interfețe grafice.

Obiective secundare:

1. Analiza problemei și a operațiilor necesare de implementat. Se analizează cerințele programului și se notează funcționalitățile acestuia si interacțiunea cu utilizatorul (2. Analiza problemei)
2. Crearea unei diagrame UML pentru elaborarea unei implementări pentru clase si a interacțiunii dintre acestea. Desenarea claselor împreună cu relațiile dintre acestea permit la o înțelegere mai bună a structurării programului și a logicii din acesta. (3. Proiectare)
3. Creearea unor clase pentru reprezentarea polinoamelor și a monoamelor. Polinoamele vor fi reprezentate ca o mulțime de monoame. (4. Implementare)
4. Elaborarea structurării programului după modelul MVC (Model-View-Controller). Pentru a avea un cod mai ușor de inteles, se va folosii structura MVC pentru a știi exact unde se găsește fiecare funcționalitate a programului. (3. Proiectare, 4.Implementare)
5. Scrierea metodelor in clasa Operations pentru operațiile pe polinoame. Clasa Operations este folosită pentru reprezentarea logicii de calcul din program. (4. Implementare)
6. Scrierea codului pentru interfața GUI a programului în clasa GUI. Clasa GUI va conține toate componentele folosite pentru interfața grafică a calculatorului polinomial (butoane, casete text etc.) (4. Implementare)
7. Scrierea claselor interne de tip ActionListener in clasa Listeners si adăugarea acestor funcționalități butoanelor create în clasa GUI. (4. Implementare)
8. Crearea clasei Main în care se vor creea instanțe pentru clasele GUI si Listenres ce vor putea permite folosirea calculatorului polinomial. Clasa Main are rolul de-a desemna locul de început al programului. (4. Implementare)

**2.Analiza problemei:**

Diagram

Description automatically generated

Interacțiunea din cadrul programului este una foarte simplă. Singura persoană ce interacționează cu programul este un utilizator oarecare ce este capabil să scrie polinoame, să le modifice și să apeleze operații matematice ce se folosesc de polinoamele scrise si care returnează în final un nou polinom ca și rezultatul operației. Toate operațiile sunt apelate prin butoane din interfața grafică, iar polinoamele sunt scrise de către utilizator sau program în casete text. In cazul în care utilizatorul imtroduce un polinom scris greșit, atunci în caseta în care ar trebuii să apară rezultatul va apărea mesajul “polinom inexistent” sau ”polinom invalid”.

Use Case: Suma dintre polinoame

Actor principal: Utilizatorul

Scenariu in caz de success:

1. Utilizatorul introduce corect polinoamele
2. Utilizatorul apasă butonul “Adunare”
3. Suma dintre polinoame este realizată
4. În caseta text pentru rezultat apare polinomul format prin însumare

Scenariu in caz de eșec:

1. Utilizatorul introduce un polinom scris greșit sau lipsă
2. Se revine la pasul 1

Use Case: Diferența dintre polinoame

Actor principal: Utilizatorul

Scenariu în caz de success:

1. Utilizatorul introduce corect polinoamele

2. Utilizatorul apasă butonul “Scadere”

3. Diferența dintre polinoame este realizată

4. In caseta text pentru rezultat apare polinomul format prin diferență

Scenariu in caz de eșec:

1. Utilizatorul introduce un polinom scris greșit sau lipsă

2. Se revine la pasul 1

Use Case: Înmulțirea dintre polinoame

Actor principal: Utilizatorul

Scenariu în caz de success:

1. Utilizatorul introduce corect polinoamele

2. Utilizatorul apasă butonul “Inmultire”

3. Înmulțirea dintre polinoame este realizată

4. În caseta text pentru rezultat apare polinomul format prin înmulțire

Scenariu în caz de eșec:

1. Utilizatorul introduce un polinom scris greșit sau lipsă

2. Se revine la pasul 1

Use Case: Împărțirea dintre polinoame

Actor principal: Utilizatorul

Scenariu În caz de success:

1. Utilizatorul introduce corect polinoamele
2. Utilizatorul apasă butonul “Impartire”
3. Împărțirea dintre polinoame este realizată
4. În caseta text pentru rezultat apare polinomul format prin împărțire

Scenariu in caz de esec:

1. Utilizatorul introduce un polinom scris greșit sau lipsă
2. Se revine la pasul 1

Use Case: Derivarea unui polinom

Actor principal: Utilizatorul

Scenariu în caz de success:

1. Utilizatorul introduce corect polinomul 1
2. Utilizatorul apasă butonul “Derivare”
3. Derivarea polinomului este realizată
4. În caseta text pentru rezultat apare polinomul format prin derivare

Scenariu în caz de eșec:

1. Utilizatorul introduce un polinom scris greșit sau lipsă
2. Se revine la pasul 1

Use Case: Integrarea unui polinom

Actor principal: Utilizatorul

Scenariu în caz de success:

1. Utilizatorul introduce corect polinomul 1
2. Utilizatorul apasă butonul “Integrare”
3. Integrarea polinomului este realizată
4. În caseta text pentru rezultat apare polinomul format prin integrare

Scenariu în caz de eșec:

1. Utilizatorul introduce un polinom scris greșit sau lipsă
2. Se revine la pasul 1

**3.Proiectare:**

Diagram

Description automatically generated

Proiectul în sine este format din 4 pachete: Data Models ce conține clasele ce descriu datele folosite în cadrul programului, Controller ce conține clasa pentru descrierea comportamentului interfeței grafice, View care conține clasa folosită pentru creearea interfeței grafice și Model care conține clasa ce implementează operațiile matematice.

Graphical user interface, text

Description automatically generated

Pachetul Model conține o singură clasă Operations care conține metodele statice folosite pentru realizarea operațiilor matematice pe polinoame. Toate aceste metode returneaza un polinom si primesc unul sau două polinoame.

Diagram

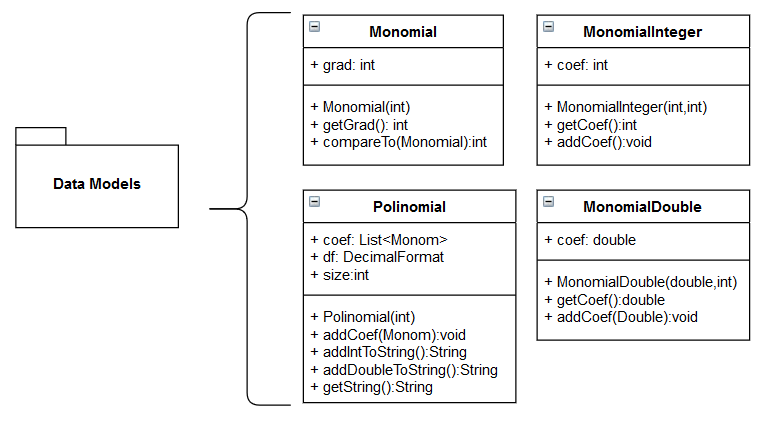
Description automatically generated

Pachetul View conține o singură clasă GUI care creează interfața grafică pentru program. Această clasă prezintă metode de returnare a mesajelor scrise in casetele text pentru polinoame, scrierea rezultatului într-o casetă text si adăugarea de ActionListeners butoanelor.

Graphical user interface

Description automatically generated

Pachetul Controller conține o singură clasă Listeners ce are rolul de-a oferii ActionListeners interfeței grafice trimise în constructor și de convertire a textului din casetele text in polinoame. În Listeners se regăsesc mai multe clase interne ce extind interfața ActionListener si care au rolul de-a permite apelarea operațiilor pe polinoame prin intermediul butoanelor din interfața grafică.



Pachetul Data Models conține clasele ce reprezintă polinoamele folosite în calculele din Operations. Clasa Polinomial conține o listă de obiecte Monomial, iar din clasa Monomial se extind alte 2 clase: MonomialInteger și MonomialDouble.

Diagram

Description automatically generated

Din diagrama UML a programului se observă mai multe asocieri speciale între clase. Clasele MonomialInteger și MonomialDouble sunt subclase ale clasei Monomial, Monomial care implementează interfața Comparable<Monomial> și care este agregat clasei Polinomial datorită existenței în Polinomial a listei de obiecte de tip Monomial. Clasa GUI este agregată clasei Listeners care, în sine, conține 5 clase interne ce extind interfața ActionListener. Interfața ActionListener este folosită pentru adăugarea de funcționalitate componentelor de interfața grafică. Clasele interne din clasa Listeners apelează metode din clasa Operations, metode ce se folosesc de obiecte de tip Polinomial. În clasa Main se regăseste metoda Main unde sunt declarate căte un obiect din clasa GUI și Listeners ce vor reprezenta calculatorul polinomial.

**4.Implementare:**

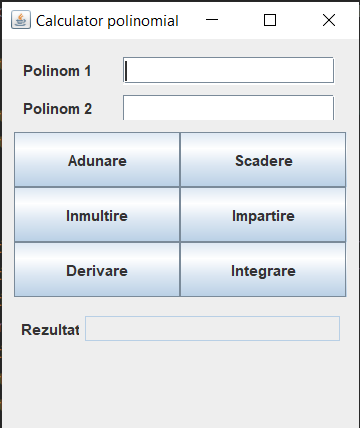
Clasa Monomial are rolul de-a reprezenta un monom, componentă de polinom. Aceasta implementează interfața Comparable. Clasa Monomial are un singur atribut grad ce este de tip private int și care reprezintă puterea monomului din polinomul din care face parte. Clasa deține un singur constructor care primeste un parametru de tip int ce va da valoare atributului grad. Monomial implementează două metode: getGrad si compareTo. Metoda compareTo din interfata Comparable primește ca și parametru un obiect de tip Monomial si returnează rezultatul metodei Integer.compare() ce primește ca și argumente gradul de la obiectul metodei compareTo și cel trimis ca parametru.

Clasa MonomialInteger extinde clasa Monomial, aceasta avand în plus un atribut și două metode. Atributul în plus este coef de tip private int ce reprezintă valoarea de tip întreg a coeficientului monomului . Clasa are un singur constructor ce primeste 2 parametrii de tip int, apeleaza constructorul super-clasei si adauga valoare atributului coef. Metoda addCoef primeste un parametru de tip int si adună valoarea acestuia la atributul coef.

Clasa MonomialDouble este aproape identică cu clasa MonomialInteger, diferențele fiind lipsa unei metode și tipul atributului coef care este private double.

Clasa Polinomial are rolul de-a fi o reprezentare a unui polinom. Această clasă are ca și atribute coef ce este o listă de obiecte Monomial și df ce este un obiect de tip DecimalFormat ce urmează pattern-ul ”0.0”. Constructorul clasei nu primește niciun parametru și inițializează lista coef ca fiind un ArrayList. Clasa implementează 4 metode: addCoef, getString, addIntToString și addDoubleToString. Metoda addCoef primește ca și parametru un obiect m de tip Monomial și verifică în lista coef dacă există un membru ce are același grad ca și m: dacă au același grad atunci valoarea coeficientului lui m este adunată la cea a monomului existent in listă, altfel obiectul m este adăugat în lista coef. Metoda createString va returna un String ce va fi o reprezentare a polinomului descris de clasa Polinomial. Această funcție apelează una din funcțiile addDoubleToString sau addIntToString ce primesc ca parametru un monom și șirul respectiv și care vor returna String-ul trimis cu monomul adăugat. Monomul respectă un pattern de forma ”SemnCoeficient\*X^Grad”. Alegerea se face în funcție de tipul obiectului Monomial: MonomialInteger sau MonomialDouble. Singurele diferențe între metodele addIntToString și addDoubleToString este clasa de monom și faptul că pentru coeficienții reali se folosește formatarea de la atributul df. Dacă in lista coef există un singur Monomial ce are coeficientul egal cu 0 atunci getString va returna un String sub forma “0”. Există totuși situații speciale pentru cazul în care coeficientul este 0, coeficientul este 1, primul monom are coeficientul pozitiv și puterea monomului este 1 sau 0.

Clasa GUI are rolul de-a creea interfata grafică a programului.



Aceasta extinde clasa JFrame și are ca atribute obiecte de tip private final JTextField pentru polinoamele ce vor fi folosite în calcule și rezultatul calculelelor și un ArrayList de obiecte de tip JButton. În constructorul clasei mai sunt declarate 3 etichete Jlabel pentru fiecare polinom și 3 panouri: pentru polinoamele 1 și 2 cu layout null, pentru butoane folosind GridLayout și pentru polinomul rezultat folosind FlowLayout. De asemenea, clasa mai prezintă metodele: getPolinomial1 și getPolinomial2 ce returnează textul din casetele polinomial1 și polinomial2, setResult ce scrie în caseta result String-ul primit, visible ce va face frame-ul GUI vizibil și addActionListener ce primește ca și parametrii un obiect a de tip ActionListener și un obiect but de tip int și care va adăuga butonului cu indexul but din listă ActionListener-ul a.

Clasa Listeners îndeplinește rolul de asignare de obiecte de tip ActionListeners la butoanele unui obiect de tip GUI, obiect ce este prezent ca și atribut clasei. În constructor este trimis ca și parametru un obiect din clasa GUI care este asignat atributului v și după aceea sunt asignate fiecărui buton din v câte o clasă internă a clasei Listeners, clase ce implementează interfața ActionListener. În clasă se regăsesc 2 metode: returnPolinomials și returnMatcher. Metoda returnMatcher va primii ca și parametru un String text și, folosind clase din pachetul regex, va verifica parametrul text pe baza unui pattern și va returna un obiect din clasa Matcher. Dacă textul transmis este gol sau conține caractere nepotrivite unui polinom, atunci va fi aruncată o excepție cu un mesaj corespunzător. Metoda getPolinomials va primii ca și parametru un număr de tip int (1 sau 2), va extrage String-ul din caseta text polinom1 sau polinom2, va apela metoda getMatcher pentru a primii un Matcher potrivit polinomului și va introduce într-un obiect de tip Polinomial obiecte din clasa Monomials ce se vor forma în funcție de grupurile regăsite în matcher. Obiectul din clasa Polinomial format va fi returnat.

Fiecare din clasele interne din Listeners urmează același principiu: este apelată metoda returnPolinomials pentru a obține polinomul 1 și eventual polinomul 2, este apelată una din metodele din clasa Operations ce corespunde operației clasei Listener și rezultatul este scris în caseta text result. În cazul în care returnMatcher a aruncat o excepție de tip Exception, aceasta va fi prinsă și mesajul acesteia va fi scris în caseta text result.

Clasa Operations are rolul de-a furniza metode statice ce corespund operațiilor matematice pe polinoame. Toate acestea primesc ca și parametrii unul sau două obiecte de tip Polinomial și returnează un obiect de tip Polinomial.

Metoda sumSub realizează atât operația de sumă, cât și de diferență. Pe lângă cele două polinoame, metoda mai primește și o variabilă op de tip short care poate fi 1 sau -1. Inițial cele două polinoame sunt sortate folosind metoda Collections.sort() și sunt inițializate două variabile i și j de tip int. Urmând un algoritm similar cu cel de merge de la mergesort, sunt comparate puterile monoamelor de pe pozițiile i și j din ambele polinoame. Dacă sunt egale atunci în lista obiectul suma este introdus un monom cu puterea corespunzătoare și coeficient care este suma dintre coeficienții celor două elemente. În caz contrar în polinomul rezultat este introdus monomul cu gradul mai mic. După ce se ajunge la capătul uneia dintre liste, toate celelalte monoame ale celeilalte liste sunt introduse în polinomul rezultat. Atunci când elementul unei liste este inserat in polinomul rezultat, indexul corespunzător listei este incrementat. Atunci când un element din polinomul 2 este implicat în inserare, coeficientul acestuia este înmulțit cu valoarea lui op care arată dacă se realizează operație de sumă sau diferență. În final se returnează polinomul rezultat.

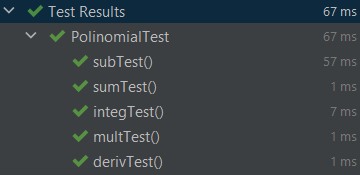
Metoda multiplication realizează operația de produs. Cele două polinoame primite sunt parcurse de câte o buclă foreach() prin intermediul căreia în polinomul rezultat vor fi introduse monoame unde coeficientul va fi format din produsul dintre coeficienții monoamelor participante din cele două polinoame, iar puterea va fi formată din suma puterilor celor 2 monoame. În final se returnează polinomul rezultat.

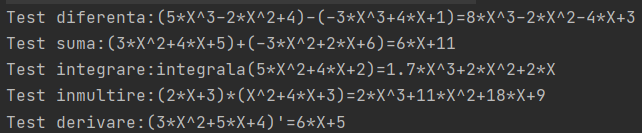
Metoda derivation realizează operația de derivare. Aceasta va primii un obiect de tip Polinomial și va parcurge printr-un foreach() fiecare monom din listă. În polinomul rezultat vor fi introduse monoame de tip MonomialInteger unde coeficientul va fi produsul dintre coeficientul monomului curent și valoarea atributului grad, iar puterea va fi egala cu atributul grad decrementat. În final se returnează polinomul rezultat.

Metoda integration realizează operația de integrare. Aceasta va primii un obiect de tip Polinomial și va parcurge printr-un foreach fiecare monom din listă. În polinomul rezultat vor fi introduse monoame de tip MonomialDouble sau MonomialInteger unde coeficientul va fi coeficientul polinomului trimis împărțit la valoarea atributului grad incrementat, iar puterea va fi egala cu atributul grad incrementat. În final se returnează polinomul rezultat.

**4.Rezultate:**

Parcurgând diferite teste simple s-a putut observa funcționarea operațiilor matematice pe polinoame. În toate testele au fost declarate unul sau două polinoame, le-au fost introduse monoame, a fost creat un polinom ce ar trebuii sa conțină rezultatul din urma operației, s-a reținut într-un alt polinom rezultatul operației apelate și, în final, printr-un assertEquals au fost comparate reprezentarile prin String returnate de metoda getString ale polinomului rezultat din operație și al polinomului cu rezultatele deduse. Egalitatea vine din faptul că pentru valorile puterilor și ale coeficienților sunt egale pentru String-uri identice.





**5.Concluzie**

Pentru elaborarea unui calculator polinomial este nevoie de anticiparea apariției de cazuri speciale ce pot să afecteze lizibiliatea sau corectitudinea rezultatului scris: semnul coeficientului monomului, existența unui singur monom cu coeficient 0, lipsa de polinoame scrise etc. Utilizatorul este nevoit să respecte o structură standard pentru scrierea polinoamelor, acest lucru însemnând evitarea scrierii de spații libere, introducerea de alte caractere litere înafară de ”X” și ”x”, introducerea de operatori în locații nepotrivite etc.

Când vine vorba de lucruri noi învățate, această temă nu a adus foarte multe contribuții. Exceptând învățarea lucrului pe Gitlab, cel mai important lucru învățat a fost folosirea claselor din pachetul regex. Acestea au putut permite citirea și verificarea corectitudinii polinoamelor scrise de către utuilizator, acestea fiind verificate dacă respectă un anumit pattern.

Îmbunătățiri posibile pentru program ar fi înfrumusețarea interfeței grafice, elaborarea operației de împărțire, crearea unui pattern pentru o libertate mai mare în scrierea polinoamelor și eventual îmbunătățirea anumitor algoritmi precum cei de la operații.

**6.Bibliografie:**

* Swing

<https://docs.oracle.com/javase/tutorial/uiswing/>

* Java Regular Expressions

<https://docs.oracle.com/javase/tutorial/essential/regex/>

* Junit

<https://www.vogella.com/tutorials/JUnit/article.html>

* Java naming conventions

<https://google.github.io/styleguide/javaguide.html>