DOCUMENTATIE

TEMA *1*

NUME STUDENT: Coșarcă Ioan-Cristian

GRUPA: …30227…

CUPRINS

1. Obiectivul temei

Se va prezenta obiectivul principal al temei printr-o fraza si un tabel sau o lista cu obiectivele secundare. Obiectivele secundare reprezintă pasii care trebuie urmați pentru indeplinirea obiectivului principal. Fiecare obiectiv secundar va fi descris si se va indica in care capitol al documentației va fi detaliat.

Obiectivul acestui proiect / acestei teme este realizarea în Java a unei aplicații cu interfață utilizator dedicată prin care utilizatorul poate insera polinoame și care să permită execuția operațiilor de adunare, scădere, înmulțire, împărțire, derivare și integrare pentru polinoame cu coeficienți reali și puteri numere pozitive.

Sub-obiective:

- Analiza problemei, modelare, scenarii, cazuri de utilizare (Capitolul 2)
- Proiectare (Capitolul 3)
- Implementare (Capitolul 4)
- Rezultate (Capitolul 5)

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

Se va prezenta cadrul de cerinte functionale formalizat si cazurile de utilizare ca si diagrame si descrieri de use-case. Descrierile use-case-urilor se vor face sub forma unui flow-chart ori sub forma unei liste conținând pașii execuției fiecărui use-case.

Cerințe funcționale:

- Calculatorul de polinoame ar trebui să îi permită utilizatorului să insereze / introducă polinoame prin intermediul interfeței
- Calculatorul de polinoame ar trebui să îi permită utilizatorului să aleagă operația dorită prin intermediul interfeței
- Calculatorul de polinoame ar trebui să afișeze rezultatul operației selectate în cadrul interfeței
- Calculatorul de polinoame ar trebui să permită efectuarea operației de adunare între două polinoame
- Calculatorul de polinoame ar trebui să permită efectuarea operației de scădere între două polinoame
- Calculatorul de polinoame ar trebui să permită efectuarea operației de înmulțire între două polinoame
- Calculatorul de polinoame ar trebui să permită efectuarea operației de împărțire a polinomului de grad mai mare la polinomul de grad mai mic
- Calculatorul de polinoame ar trebui să permită efectuarea operației de derivare asupra primului polinom
- Calculatorul de polinoame ar trebui să permită efectuarea operației de integrare asupra primului polinom

Cerințe non-funcționale:

- Calculatorul de polinoame ar trebui să fie intuitiv și ușor de folosit de către utilizator
- Calculatorul de polinoame ar trebui să permită corectarea / ștergerea polinomului introdus de utilizator
- Calculatorul de polinoame ar trebui să permită introducerea unui polinom în cadrul interfeței, câte unul pe rând

Use Case: Adună Polinoame

Actor Primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează două polinoame în interfața grafică

2. Utilizatorul apasă butonul “Add”

3. Calculatorul de polinoame efectuează adunarea celor două polinoame și afișează rezultatul în câmpul “Result”

Secvență alternativă: Polinoame Nevalide

- Utilizatorul inserează polinoame incorecte
- Scenariul se întoarce la pasul 1

Use Case: Scade Polinoame

Actor Primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează două polinoame în interfața grafică

2. Utilizatorul apasă butonul “Subtract”

3. Calculatorul de polinoame efectuează scăderea celor două polinoame și afișează rezultatul în câmpul “Result”

Secvență alternativă: Polinoame Nevalide

- Utilizatorul inserează polinoame incorecte
- Scenariul se întoarce la pasul 1

Use Case: Înmulțește Polinoame

Actor Primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează două polinoame în interfața grafică

2. Utilizatorul apasă butonul “Multiply”

3. Calculatorul de polinoame efectuează înmulțirea celor două polinoame și afișează rezultatul în câmpul “Result”

Secvență alternativă: Polinoame Nevalide

- Utilizatorul inserează polinoame incorecte
- Scenariul se întoarce la pasul 1

Use Case: Împarte Polinomul de grad mai mare la Polinomul de grad mai mic

Actor Primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează două polinoame în interfața grafică

2. Utilizatorul apasă butonul “Divide”

3. Calculatorul de polinoame efectuează împărțirea polinomului de grad mai mare la cel de grad mai mic și afișează rezultatul în câmpul “Result”

Secvență alternativă: Polinoame Nevalide

- Utilizatorul inserează polinoame incorecte
- Scenariul se întoarce la pasul 1

Use Case: Derivata Polinomului

Actor Primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează un polinom în prima căsuță din interfața grafică

2. Utilizatorul apasă butonul “Derivative”

3. Calculatorul de polinoame efectuează derivarea primului polinom și afișează rezultatul în câmpul “Result”

Secvență alternativă: Polinoame Nevalide

- Utilizatorul inserează polinoame incorecte
- Scenariul se întoarce la pasul 1

Use Case: Integrala Polinomului

Actor Primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează un polinom în prima căsuță din interfața grafică

2. Utilizatorul apasă butonul “Integeration”

3. Calculatorul de polinoame efectuează integrarea primului polinom și afișează rezultatul în câmpul “Result”

Secvență alternativă: Polinoame Nevalide

- Utilizatorul inserează polinoame incorecte
- Scenariul se întoarce la pasul 1

3. Proiectare

*Se va prezenta proiectarea OOP a aplicației, diagramele UML de clase si de pachete, structurile de date folosite, interfețele definite si algoritmii folosiți (*daca e cazul)

Proiectarea aplicației calculatorului respectă arhitectura Model, View, Controller. Aceasta separă aplicația în 3 zone: procesare, ieșire și intrare.

Model – încapsulează datele și funcționalitățile

View – se ocupă de implementarea interfeței cu utilizatorul

Controller – fiecare View are asociat un Controller, acesta primește informațiile de intrare sub formă de evenimente; evenimentele sunt convertite în apeluri de funcții sau proceduri, care sunt apoi trimise către Model sau către View

Componenta Model este alcătuită din două clase: CalcPolinomModel (pentru a asigura reprezentarea polinoamelor ca și ArrayList – uri de monoame și care implementează operațiile asupra acestora) și Monom (clasa care construiește și asigură reprezentarea unui monom și care efectuează operațiile asupra acestuia).

Diagrame UML de clase:

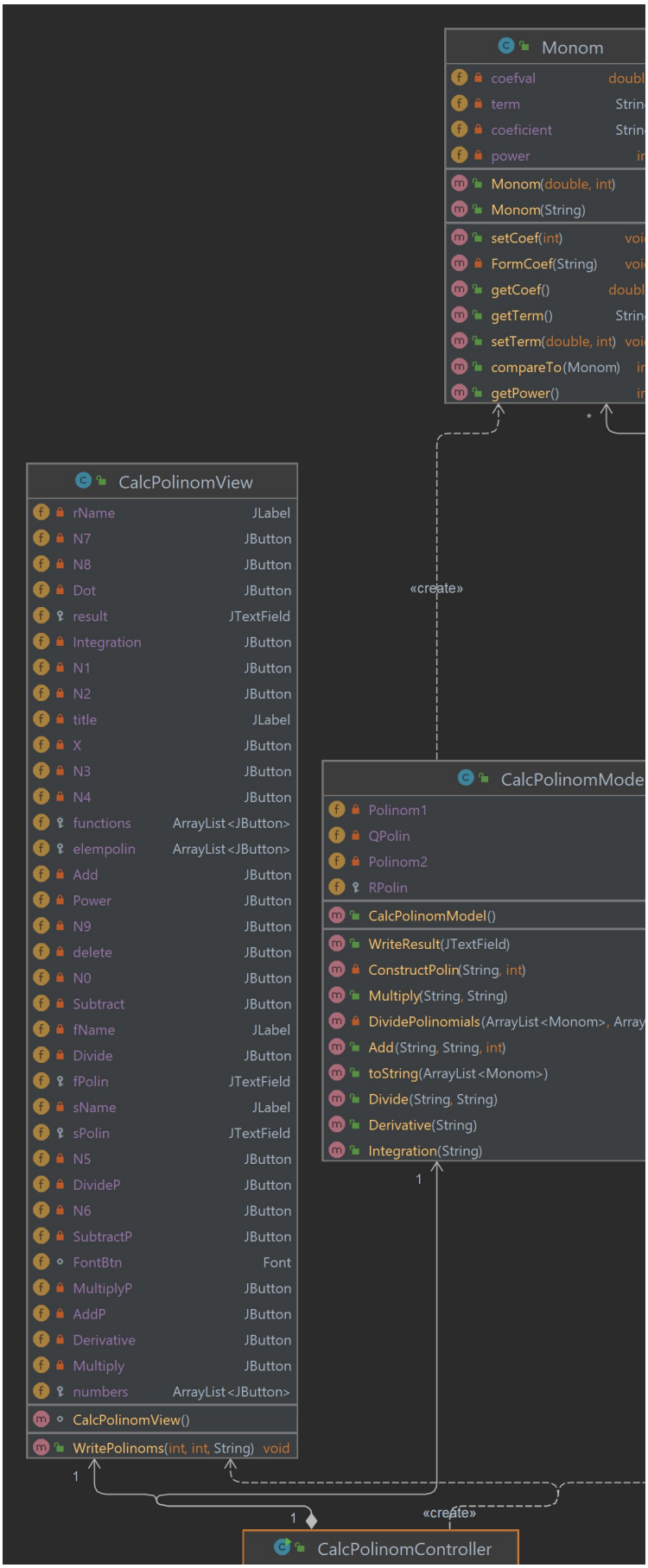
În cadrul aplicației avem următoarele 4 clase:

- CalcPolinomView
- CalcPolinomController
- CalcPolinomModel
- Monom

Între clasa CalcPolinomController și CalcPolinomView, respectiv între clasa CalcPolinomController și CalcPolinomModel putem remarca o agregare simplă: un controller are exact o vedere și un controller are exact un model.

Între clasa CalcPolinomModel și Monom se pot observa în schimb relații de asociere multiplă, prin intermediul variabilelor declarate în model: Polinom1, Polinom2, RPolin și QPolin. Acestea fiind declarate ca și ArrayList – uri ale clasei Monom, putem spune că listele au zero sau mai multe Monoame fiecare.

De asemenea, avem relații de dependență între CalcPolinomController și CalcPolinomView, CalcPolinomController și CalcPolinomModel, respectiv CalcPolinomModel și Monom.


Monom
coefval
term
coeficient
power
Monom(double, int)
Monom(String)
setCoef(int)
FormCoef(String)
getCoef()
getTerm()
setTerm(double, int)
compareTo(Monom)
getPower()
*
«create»
CalcPolinomView
rName JLabel
N7 JButton
N8 JButton
Dot JButton
result JTextField
Integration JButton
N1 JButton
N2 JButton
title JLabel
X JButton
N3 JButton
N4 JButton
functions ArrayList<JButton>
elempolin ArrayList<JButton>
Add JButton
Power JButton
N9 JButton
delete JButton
N0 JButton
Subtract JButton
fName JLabel
Divide JButton
fPolin JTextField
sName JLabel
sPolin JTextField
N5 JButton
DivideP JButton
N6 JButton
SubtractP JButton
FontBtn Font
MultiplyP JButton
AddP JButton
Derivative JButton
Multiply JButton
numbers ArrayList<JButton>
CalcPolinomView()
WritePolinoms(int, int, String) void
CalcPolinomMode
Polinom1
QPolin
Polinom2
RPolin
CalcPolinomModel()
WriteResult(JTextField)
ConstructPolin(String, int)
Multiply(String, String)
DividePolinomials(ArrayList<Monom>, Array
Add(String, String, int)
toString(ArrayList<Monom>)
Divide(String, String)
Derivative(String)
Integration(String)
1
1
1
«create»
CalcPolinomController

Am folosit structurile de ArrayList pentru a grupa diverse butoane în cadrul interfeței grafice și a le putea atribui proprietăți mai ușor. De asemenea, am utilizat ArrayList și pentru a reprezenta un Polinom ca și un șir de Monoame.

Clasa Controller "CalcPolinomController" implementează interfețele ActionListener și MouseListener. Interfața ActionListener asigură răspunsul aplicației în momentul în care un buton asupra căruia s-a pus un ActionListener este apăsat prin intermediul metodei pe care o implementează, actionPerformed. Interfața MouseListener are o serie de metode pe care le implementează ca răspuns pentru diverse operații cu mouse-ul. Dintre acestea, metoda pe care o folosim explicit este mouseClicked. În cadrul ei, specificăm ca doar căsuța în care s-a dat click să fie disponibilă pentru inserarea unui polinom.

Clasa Monom implementează interfața Comparable, ce primește ca și tip de argument obiecte de tip Monom. Această interfață implementează metoda compareTo, metodă care este folosită în aplicație pentru a ordona monoamele unui polinom în ordine descrescătoare după grad.

Algoritmi folosiți:

Adunarea:

Pentru a calcula suma a două polinoame, se adună coeficienții monoamelor de același grad.

Scăderea:

Pentru a calcula diferența a două polinoame, se scad coeficienții monoamelor de același grad.

Înmulțirea:

Pentru a calcula produsul a două polinoame, se înmulțesc pe rând monoamele polinomului 1 cu monoamele polinomului 2, monoamele obținute memorându-se. La final se adună monoamele de același grad.

Împărțirea:

Pentru a calcula câtul împărțirii a două polinoame, se urmează pașii:

1. Se ordonează monoamele polinomului 1 și 2 în ordine descrescătoare după grad.

2. Se va împărți polinomul care are cel mai mare grad la polinomul cu grad mai mic (considerăm polinomul 1 ca fiind cel care are gradul cel mai mare pentru exemplificarea următorilor pași).

3. Se împarte primul monom al polinomului 1 la primul monom al polinomului 2 și se obține primul termen al câtului.

4. Se înmulțește valoarea câtului obținut până acum cu polinomul 2.

5. Rezultatul obținut la pasul anterior îl scădem din polinomul 1, obținând restul împărțirii.

6. Se repetă algoritmul începând de la pasul 2, considerând restul ca fiind noul polinom 1, până când gradul restului va fi mai mic decât gradul polinomului 2.

Derivarea:

Pentru a calcula derivata unui polinom, pentru fiecare monom al său:

1. Coeficientul se înmulțește cu puterea.

2. Valoarea puterii va scădea cu o unitate.

Integrarea:

Pentru a calcula integrala unui polinom, pentru fiecare monom al său:

1. Valoarea puterii va crește cu o unitate.

2. Coeficientul se împarte la noua valoare a puterii.

4. Implementare

Se va descrie fiecare clasa cu câmpuri si metodele importante. Se va descrie implementarea interfeței utilizator.

Aplicația este structurată pe 4 mari clase: CalcPolinomController, CalcPolinomModel, CalcPolinomView și Monom.

Clasa CalcPolinomView:

Această clasă asigură implementarea interfeței grafice cu utilizatorul a aplicației.

Câmpurile care sunt declarate în clasă sunt elemente ale interfeței

Avem astfel o etichetă pentru titlul aplicației (care va fi pusă sus, centrat), etichete și casete text pentru cele două polinoame ce urmează a fi introduse și pentru rezultatul ce urmează a fi afișat, și butoane pentru operațiile între cele două polinoame, respectiv butoane ce permit scrierea polinoamelor caracter cu caracter.

Butoanele numerice, precum și butoanele simbol (“+”, “-“, “.” Etc.) permit adăugarea respectivului caracter (reprezentat de către acestea) în cadrul polinomului în care se face introducerea, prin apăsarea lor de către utilizator.

Pentru a introduce un polinom, utilizatorul trebuie doar să dea click în căsuța text a polinomului unde dorește să facă adăugarea, apoi să apese pe butoanele numerice sau cu simboluri pentru a le scrie.

Butonul de “del” (delete) are rolul de a șterge ultimul caracter introdus întru-n polinom, dacă se dorește acest lucru, sau ștergerea întregului polinom, prin apăsarea repetată a butonului de către utilizator.

Căsuța text a rezultatului va afișa rezultatul operației selectate a se efectua asupra polinoamelor, prin apăsarea butonului specific operației.

În cadrul acestei clase avem o singură metodă pe lângă constructor, și anume WritePolinoms, care primește două numere întregi (folosite pe post de indicatori) și un string. Metoda are rolul de a verifica care din cele două indicatoare are valoarea 1 (indicatorul 1 va fi 1 dacă căsuța polinomului 1 a fost selectată, indicatorul 2 va fi 1 dacă căsuța polinomului 2 a fost selectată) și să adauge conținutul stringului în polinomul al cărui indicator e 1. Stringul va avea valoarea unui caracter specific butoanelor numerice sau butoanelor cu simboluri.

Clasa CalcPolinomController:

Clasa Controller comandă ce anume se va executa în Model și ce anume se va afișa în View, prin receptarea diverselor evenimente / acțiuni produse de utilizator (apăsare butoane, apăsare într-o casetă text). Această clasă asigură legătura între View și Model.

Clasa are ca și variabile o componentă de tipul View, o componentă de tipul Model și doi indicatori care în decursul execuției, prin schimbarea alternativă a lor, vor indica în ce polinom / casetă text se va face scrierea.

Metoda actionPerformed detectează ce buton s-a apăsat și comandă să se execute următoarele:

- Dacă s-a apăsat un buton numeric sau simbol, se va scrie în polinom.
- Dacă s-a apăsat butonul unei operații, se va trece la execuția acesteia în metoda CalcResult

Metoda CalcResult detectează ce operație s-a selectat, trimite execuția în Model pentru aflarea rezultatului, iar în final trimite rezultatul obținut în View pentru a fi afișat.

Clasa CalcPolinomModel:

Această clasă asigură reprezentarea polinoamelor ca și ArrayList – uri ale clasei Monom și efectuarea operațiilor asupra lor.

Variabilele clasei sunt 4 ArrayList – uri: Polinom1, Polinom2, RPolin unde va fi memorat rezultatul și QPolin (cât), polinom care să ajute la efectuarea operației de împărțire.

Metoda ConstructPolin primește un string și un indicator i care va specifica în ArrayList – ul cărui polinom se va face adăugarea. Metoda reinițializează rezultatul anterior RPolin (util în special la împărțire) și folosind pattern matching, separă stringul polinomului în substringuri, reprezentând monoamele polinomului, în forma "(-)coefficient*X^putere". Se construiește apoi un obiect Monom cu substringul obținut. Dacă i este 1, se adaugă monomul în polinomul 1, iar dacă este 2, se adaugă în polinomul 2, iar la final se sortează listele în ordine descrescătoare după puterea monoamelor.

Metoda Add primește două polinoame reprezentate ca și string și un indicator i care, dacă e 1, precizează că se va executa adunarea normală, iar dacă e -1, se va face adunarea primului polinom cu -1*(al doilea polinom). Monoamele rezultatului vor fi memorate în RPolin.

Metoda Multiply primește două polinoame reprezentate ca stringuri și le înmulțește monoamele fiecăruia cu monoamele din celălalt, rezultatele fiind memorate în RPolin. Dacă unul din monoamele rezultate în urma înmulțirii are aceeași putere cu un monom din RPolin, îi vom modifica coeficientul (executăm adunarea monoamelor de același grad).

Metoda Divide primește două polinoame reprezentate ca stringuri și le compara gradele maxime (după creare în ConstructPolin, monomul de grad maxim al fiecărui polinom este primul din listă). În funcție de care are gradul mai mare, se transmit listele ca argumente spre metoda DividePolinomials, în ordinea deîmpărțit, împărțitor. La revenirea din metoda, polinomul QPolin va avea memorat rezultatul, care va fi copiat în RPolin.

Metoda DividePolinomials primește două liste, deîmpărțitul și împărțitorul. Împărțirea va fi realizată conform algoritmului de împărțire descris în Capitolul 3. Polinomul cu ArrayList – ul QPolin va fi folosit pentru reținerea câtului în cadrul împărțirii repetate.

Metoda toString primește un ArrayList de monoame și creează polinomul descris de listă, convertind-o într-un string cu coeficienți, operații, parametrii și puteri.

Metoda Derivative primește un string reprezentând primul polinom și îi calculează derivata, monoamele rezultatului fiind memorate în RPolin.

Metoda Integration primește un string reprezentând primul polinom și îi calculează integrala, monoamele rezultatului fiind memorate în RPolin.

Metoda WriteResult caseta text a rezultatului JTextField și, folosind ArrayList – ul RPolin deja calculat în cadrul celorlalte metode, îl transformă într-un string pe care să îl afișeze pe ecran.

Clasa Monom:

Clasa asigură reprezentarea fizică a unui monom al unui polinom.

Variabilele clasei sunt elemente specifice ale monomului. term reține reprezentarea acestuia ca string, așa cum este extras substringul. Stringul coeficient va memora substringul din monom corespunzător valorii coeficientului. Coefval și power vor reține valoarea reală a coeficientului și puterea.

Constructorul cu un parametru primește stringul extras din polinom și folosind pattern matching, extrage substringul coeficientului și puterea monomului. Se apelează metoda FormCoef pentru a calcula valoarea coeficientului.

Metoda FormCoef primește un string reprezentând expresia coeficientului și calculează valoarea acestuia. Coeficienții care ajung în această funcție au stringul fie de forma “+”, “-“; fie de forma “x/y”. În primul caz, coeficientul ca avea valoarea 1, respectiv -1. În cel de-al doilea caz, folosind un alt pattern matching, extragem x și y, le parsăm valorile și punem câtul împărțirii în coefval.

Constructorul cu doi parametrii primește valorile coeficientului și puterii unui monom și construiește reprezentarea acestuia ca string în term. Se va apela setter-ul setTerm.

Metoda compareTo, implementată prin intermediul interfeței Comparable pe Monom (Comparable<Monom>) primește un obiect de tip monom și prin comparație a valorii puterii cu puterea obiectului curent, returnează un număr negativ ce va dicta ordonare descrescătoare.

5. Rezultate

Se vor prezenta scenariile pentru testare cu Junit sau alt framework de testare.

Vom testa aplicația cu Junit, verificând efectuarea corectă a operațiilor Add, Subtract, Multiply, Divide, Derivative și Integration din cadrul modelului, prin apelul lor din metode de testare.

Pentru fiecare operație, definim polinomul 1 ca fiind “3*X^2-X+1” și polinomul 2 ca fiind “X-2”. Vom ține cont și de modul de scriere / formare a rezultatului ca string, având în vedere scrierea coeficienților cu formatul "%.2f".

Pentru testul adunării, definim metoda de test testAdd. Rezultatul așteptat este "3.00*X^2-1.00". Verificăm cu assertEquals.

Pentru testul scăderii, definim metoda de test testSubtract. Rezultatul așteptat este "3.00*X^2-2.00*X+3.00". Verificăm cu assertEquals.

Pentru testul înmulțirii, definim metoda de test testMultiply. Rezultatul așteptat este "3.00*X^3-7.00*X^2+3.00*X-2.00". Verificăm cu assertEquals.

Pentru testul împărțirii, definim metoda de test testDivide. Rezultatul așteptat este "3.00*X+5.00". Verificăm cu assertEquals.

Pentru testul derivării, definim metoda de test testDerivative. Rezultatul așteptat este "6.00*X-1.00". Verificăm cu assertEquals.

Pentru testul integrării, definim metoda de test testIntegration. Rezultatul așteptat este "X^3-0.50*X^2+X". Verificăm cu assertEquals.

La final constatăm că toate cele 6 teste au fost executate cu succes.

6. Concluzii

Se vor prezenta concluziile, ce s-a învățat din tema, dezvoltări ulterioare.

În urma executării testelor cu succes se constată că aplicația are comportamentul dorit și ajunge la rezultatele așteptate.

Se constată că polinoamele pot fi introduse ca și stringuri și apoi prelucrate pentru a obține monoamele, iar apoi coeficienții și puterile lor.

Tema exemplifică cât de important este ca o aplicație să fie dezvoltată cu o organizare bună pe obiecte și arată cât de importantă este crearea unei arhitecturi a aplicației de forma Model, View, Controller.

În urma acestui proiect am învățat mai multe despre depanarea unei aplicații și organizarea procedurală a funcțiilor și metodelor, precum și lucrul între clase.

Ca dezvoltare ulterioară aș dori extinderea posibilităților de reprezentare a polinoamelor, astfel încât operațiile să se poată efectua și pentru polinoame cu coeficienți complecși și cu puteri negative.

7. Bibliografie

https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf

https://regex101.com/

https://www.baeldung.com/regular-expressions-java

https://courses.lumenlearning.com/wmopen-collegealgebra/chapter/introduction-dividing-polynomials/