**Documentatie Calculator**

**Polinoame**

Pop Mihali Marco Silviu

1. Obiectivul temei

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

3. Proiectare

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

**1. Obiectivul temei**

Obiectivul principal al temei este de a crea o aplicatie java ce poate efectua operatii de adunare, scadere, inmultire, impartire, derivare si integrare pe polinoame introduse de utilizator prin intermediul unei interfete grafice.

Obiective secundare:

* Crearea interfetei grafice (cap 4 )
* Preluarea inputului de la utilizator(cap 4)
* Prelucrarea inputului si stocarea acestuia in memorie (cap 4)
* Implementarea operatiilor pe polinoame (cap 4)

Crearea interfetei grafice:

Crearea unei interfete grafice ce va lasa utilizatorul sa introduca polinoame si sa selecteze operatiile prin intermediul unor butoane. De asemenea afisarea rezultatea se va face tot in interiorul interfetei.

Preluarea inputului de la utilizator:

Textul introdus este preluat din campurile de tip text si este stocat.

Prelucrarea inputului si stocarea in memorie

Pe Stringul ce contine polinomul se efectueaza operatii de replace si match folosind regex pentru al transforma intr-un polinom pe care se efectueaza operatiile ulterioare.

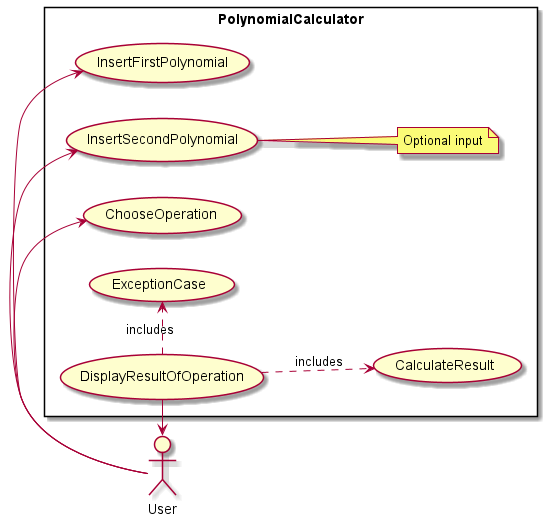
Implementarea operatiilor pe polinoame

Crearea funtionalitatii de a efectua operatii de adunare, scadere, inmultire, impartire, derivare sau integrare pe polinomul/polinoamele introduse.

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

Cerinte functionale:

* Posibilitatea de introducere a unul sau doua polinoame
* Posibilitatea alegerii operatiei
* Efectuarea operatiilor in cazul in sunt posibile
* Returnarea unei exceptii in cazul in care operatiile nu se pot efectua
* Afisarea unui rezultat sau a unui mesaj care descrie eroarea in functionare



Use-case diagram

Scenariul principal:

* Utilizatorul introduce 1 sau 2 polinoame
* Se verifica daca polinoamele sunt corecte
* Se verifica daca operatia poate fi efectuata

Scenariul succes:

* Operatia este efectuata
* Polinomul sau polinoamele rezultate sunt afisate in interfata grafica

Scenarii alternative:

1. Unul dintre polinoame nu este introdus ( cand operatia necesita 2 operanzi )
2. Este aruncata o exceptie
3. Mesajul este afisat in locul rezultatului
4. Operatia aleasa este impartire si numaratorul este 0
5. Este aruncata o exceptie
6. Mesajul este afisat in locul rezultatului

**3. Proiectare**

Structura, organizarea in pachete: constants, exceptions, model, service, validator, view

* constants : contine toate clasele in care sunt definite doar constante
* exceptions: contine toate clasele care extind Exceptions din java
* model : in model se alfa toate clasele care retin date sau stari
* service: aici sunt clasele ce au doar metode statice folosite pentru diferite calcule
* validator: contine doar clase cu metode ce valideaza anumite date
* view: clasele in care sunt declarate atribute din Java Swing, adica toate clasele ce au elemente ce vor face parte din interfata grafica

Structuri de date folosite:

Am folosit o clasa Monomial care sa retina puterea si coeficientu pentru un singur monom dintr-un folonomial. Pentru a retine un polinom am folosit o clasa Polynomial ce are ca atribute o lista de monoame. Am folosit 2 interfate numite TestInterface, ForMessage generica cu 2 tipuri pentru a abstractiza operatiile pe monoame din clasa MonomialCalculator.

Algoritmi folositi:

Adunarea polinoamelor = Am adunat coeficientii monoamelor cu puteri egale

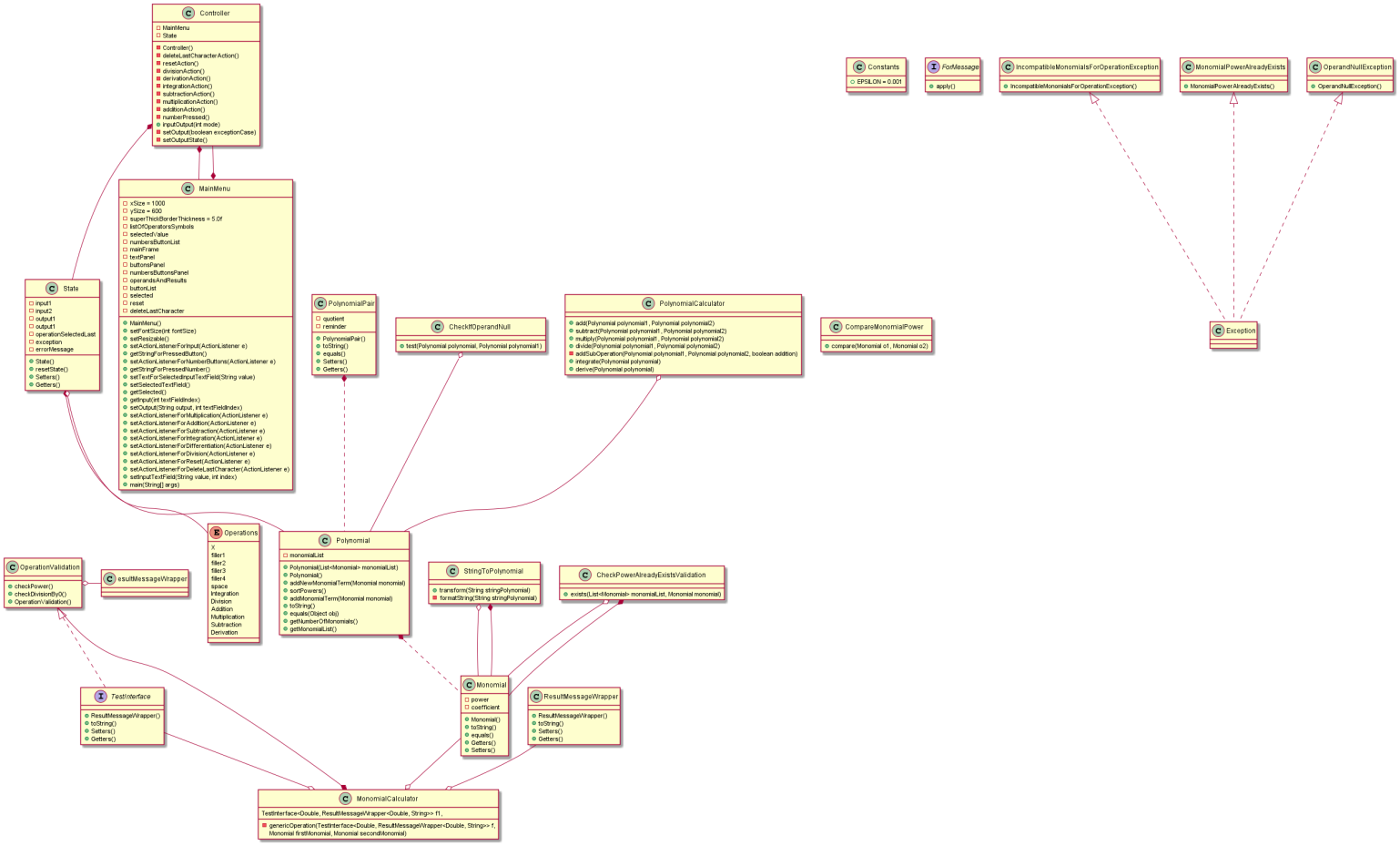
Scaderea polinoamelor = Am scazut coeficientii monoamelor cu puteri egale

Inmultirea polinoamelor = Am calculat produsul de convolutie al listelor de monoame si am sortat polinomul rezultat in ordinea descrescatoare a puterilor

Derivarea polinoamelor = Pentru fiecare monom am inmultit coeficientul initial cu puterea si am scazut puterea cu o unitate

Integrarea polinoamelor = Pentru fiecare monom am impartit coeficientul initial cu puterea initiala + 1 si am crescutputerea cu o unitate

Impartirea polinoamelor = Cat timp gradul deimpartitului este mai mare sau egal cu gradul impartitorului urmatoarele operatii au loc. Am impartit monoamele dominante si punem monomul la polinomul de cat, dupa care inmultesc monomul rezultat din impartire cu impartitorul intr-un polinom temporar. Dupa scad din deimpartit temporar. In final deimpartitul ramas devine restul operatiei.



UML Class Diagram

**4. Implementare**

Pachetul constants:

* Constants = Aici sunt retinute toate constantele ca atribute static final, in particular exista o singura constanta EPSILON folosita pentru precizia compararii variabileleor de tip double

Pachetul controller:

* Controller = Clasa face legatura dintre view si model.
  + MainMenu = atribut pentru interfata grafica
  + State = obiect ce retine starea applicatiei, polinoamele introduse si operatia aleasa
  + Controller() = contructor ce activeaza optiunea de resizable a interfetei grafice si care seteaza action listeners pentru butoane
  + Mai multe clase de tip \*Action pentru a seta state-ul si output-ul pentru fiecare caz
  + inputOutput(int more) = seteaza textul din interfata grafica l in functie de datele din state
  + setOutputState = seteaza valorile de output din state, valori ce urmeaza a fi afisate

Pachetul exceptions:

* IncompatibleMonomialForOperationException
* MonomialPowerAlreadyExists
* OperandNullException

Pachetul model:

* Monomial = Retine un monom
  + power = double ce retine puterea monomului
  + coefficient = double ce retine coeficientul monomului
  + equals() = verifica daca 2 monoame au valori egale
  + toString() = transforma un monom intr-un text ce poate fi afisat
* Polynomial = Retine un polinom ca lista de monoame
  + monomialList = lista ce retine monoamele din care e format polinomul
  + addNewMonomialTerm(Monomial) = primeste ca input un monom si il adauga in lista doar daca un monom cu aceeasi putere nu exista deja, altfel returneaza o exceptie
  + addMonomialTerm(Monomial) = primeste ca input un monom pe care il adauga, in cazul in care exista un monom cu aceeasi putere aduna coeficientii
  + toString()
  + equals()
  + getNumberOfMonomials = returneaza numarul de monoame cu coeficient nenul din polinom
  + sortPower() = sorteaza puterile monoamelor in ordine descrescatoare
* PolynomialPair = Retine o pereche de polinoame, rezultatul unei impartiri
  + quotient = retine catul unei impartiri a doua polinoame
  + reminder = retine restul unei impartiri a doua polinoame
  + toString()
  + equals()
* State = Retine starea curenta a aplicatiei
  + input1, input2 = polinoame ce retin informatiile din campurile text de input
  + output1, outpu2 = polinoame ce retin ce trebuie afisat ca rezultat al operatiei selectate
  + operationSelectedLast = ultima operatie selectata de utilizator
  + exception = boolean ce are valoarea true daca datele introduse pentru operatie sunt incorecte
  + errorMessage = mesajul ce trebuie afisat pentru utilizator
  + resetState() = aduce starea aplicatiei la cea initiala unde output-rile sunt polinoame goale si operatia aleasa este filler1, valoare ce reprezinta starea dinainte de a fi aleasa o operatie de utilizator

Pachetul service:

* CheckIfOperandNull
  + test(Polynomial, Polynomial) = verifica daca cel putin unul dintre polinoamele introduse este gol, in caz pozitiv arunca o eroare
* CompareMonomialPower
  + compare(Monomial, Monomial) = returneaza diferenta dintre puterile monoamelor
* MonomialCalculator
  + genericOperation() = este o metoda generica ce efectueaza o operatie sau arunca o exceptie in cazul in care operatia nu poate fi efectuata
  + add()
  + subtract()
  + multiply()
  + divide()
  + derive()
  + integrate()
* PolynomialCalculator
  + add() = apeleaza addSubOperation() cu ultimul parametru true
  + subtract() = apeleaza addSubOperation() cu ultimul parametru false
  + multiply()
  + divide() = returneaza un obiect de tip PolynomialPair ca rezultat al impartirii
  + addSubOperation() = efectueaza adunare sau scaderea polinoamelor in functie de valoarea unui parametru numit addition
  + integrate()
  + derive()
  + changeNothingToZero(PolynomiarPair) = primeste un PolynomiarPair, iar daca vreunul din polinoamele sale are 0 monoame ii este adaugat un monom cu puterea si coeficientul egale cu 0
* StringToPolynomial = transforma un string intr-un polinom
  + transform() = textul primit este transformat in numere si semne '^', dupa care un scanner citeste valorile si le stocheaza
    - partialResult = o lista de monoame ce retine monoamele ce vor fi face parte din polinomul rezultat
    - pattern = sablonul pentru regex care separa textul in monoame
  + formatString() = metoda ce aduce textul sub forma unor numere cu semne de puteri

Pachetul validator:

* CheckPowerAlreadyExists
  + exists() = verifica daca puterea unui monom se afla intr-o lista specificata de monoame
* OperationValidation
  + checkPower() = verifica puterea monoamelor pentru adunare,
  + checkDivisionBy0 = verifica daca impartitorul sa nu fie 0

Clasa Main creaza un obiect de tip controller.

Interfata grafica:

Interfata grafica are 4 componente tip textfield si 24 de butoane. Dimensiunea butoanelor si a scrisului se modifica in functie de dimensiunea ferestrei. Operatiile sunt efectuate dinamic, daca dupa apasarea unei operatii input-ul se schimba rezultatul se va schimba automat considerand ultima operatie efectuata.

* ferestre de input

In prima coloana dintre ferestrele de tip text se afla 2 ce au rol de input. Cea de pe prima linie este pentru primul polinom, iar cea de pe randul 2 pentru al doilea polinom. In cazul in care operatia este de integrare sau derivare informatia din cea dea doua fereastra este ignoarata. In ferestre se poate introduce text in 2 moduri folosind tastatura sau folosind butoanele. Se poate selecta fereastra in care va aparea textul prin apasarea unui click stanga pe fereastra dorita.

* ferestre de output

A doua coloana este formata din 2 ferestre text in care va fi afisat output-ul operatiei selectate. Pentru operatia de impartire ambele ferestre voi face parte din raspuns, dar pentru restul cea dea doua va fi mereu goala. In cazul impartirii prima fereastra va fi goala daca catul rezultat este 0. Prima fereastra va afisa si mesajele de eroare daca este este cazul.

* butoanele pentru cifre

Interfata are 10 butoane pentru cele 10 cifre. Pe fiecare buton apare ca text cifra corespunzatoare. Cifrele, respectiv numerele formate cu acestea, reprezista coeficienti sau puteri ale monoamelor in functie de pozitia acestora.

* butoanele pentru simboluri polinom

Exista 6 butoane speciale pentru introducerea polinoamelor:

* + Butonul 'x' introduce in fereastra de text variabila x care este folosita implicit pentru polinoame
  + Butonul '+'
  + Butonul '-'
  + Butonul '\*'
  + Butonul '^' reprezinta puterea unui singur monom, acesta trebuie pus dupa x
  + Butonul 'space' pune un spatiu un spatiu in fereastra in care se introduce textul
* butoanele pentru operatii

Exista 6 butoane numite 'Integraiton', 'Division', 'Addition', 'Multiplication', 'Subtraction', 'Derivation', care efectueaza operatiile de integrare, impartire, adunare, inmultire, scadere, respectiv derivare. Dupa ce a fost apasat unul dintre butoane operatia ramane salvata in state-ul aplicatiei, iar orice schimbari ulterioare ale input-ul vor modifica output-ul fara a fi nevoie de reapasarea butonul pentru ultima operatie selectata.

* butonul de reset

Butonul 'reset' reseteaza starea aplicatiei, cand se intampla asta toate ferestrele de text devin goale, input-urile si operatia selectata nu mai sunt retinute.

* butonul de stargere

Butonul de '<--' starge ultimul caracter scris in fereastra de text selectata daca acesta exista.

**5. Rezultate**

PolynomialTest:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | initial input | expected output | actual output | passed? |
| sort powers | -3x^2+4x^5+9x^3 | 4x^5+9x^3-3x^2 | 4x^5+9x^3-3x^2 | passed |
| equals | -3x^2+4x^5+9x^3  ==  3x^2+4x^5+9x^3 | true | true | passed |
| add monomial term | -3x^2 add  x^2 | -2x^2 | -2x^2 | passed |

CompareMonomialPowerTest:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | initial input | expected output | actual output | passed? |
| compare monomials powers | -3.7x^2 ? -3x^2 | 0 | 0 | passed |
| -3.7x^3 ? -3x^2 | -1 | -1 | passed |
| -3.7x^2 ? -3x^4 | 1 | 1 | passed |

MonomialCalculator:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | initial input | expected output | actual output | passed? |
| add  monomials | 2x + 3x | 5x | 5x | passed |
| subtract monomials | 2x – 4x | -2x | -2x | passed |
| multiply monomials | 2x^-4 \* 4x^3 | 8x^-1 | 8x^-1 | passed |
| divide  monomials  (exception) | 2x^2 / 0x^3 | DividedBy0Exception | DividedBy0Exception | passed |
| divide  monomials | 4x^-4 / 3x^3 | 1.3333 x^-7 | 1.3333 x^-7 | passed |
| derive  monomial | -2x^2 | -4x | -4x | passed |
| integrate  monomial | 2x^-5 | -0.5x^-4 | -0.5x^-4 | passed |

PolynomialCalculator:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | initial input | expected output | actual output | passed? |
| multiply  polynomials | 4x^4 +4 x^3 -3x^ 2+4 x \*  2x^3-2x ^ 2 -3 x +1 | 8x^7-26x ^ 5+6x ^ 4+5x ^ 3 -15 x ^ 2 +4 x | 8x^7-26x ^ 5+6x ^ 4+5x ^ 3 -15 x ^ 2 +4 x | passed |
| add  polynomials | x^2+3 +  x^2 + x +4 | 2x^2 + x +7 | 2x^2 + x +7 | passed |
| subtract  polynoamials | x^3 + x^2 –  x^2 +3 | x^3-3 | x^3-3 | passed |
| derivate  polynomial | 2x^3 + 3x^2 | 6x^2 + 6x | 6x^2 + 6x | passed |
| integrate  polynomial | 2x^3 + 3x^2 | 0.5x^4 + x^3 | 0.5x^4 + x^3 | passed |
| division  polynomial | x^3+ 2x^2 +5 /  x^2 +1 | x +2  -x+3 | x +2  -x+3 | passed |

StringToPolynomial:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | initial input | expected output | actual output | passed? |
| transform string  to polynomial | "5x ^7-103x ^3 - x^30 +666" | 5x ^7-103x ^3 - x^30 +666 | 5x ^7-103x ^3 - x^30 +666 | passed |

CheckPowerAlreadyExists:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | initial input | expected output | actual output | passed? |
| check if power  exists | -3x^2 + 4x^5 + 9x^3  check 4x | false | false | passed |

**6. Concluzii**

Am realizat o aplicatie java care efectueaza operatii cu polinoame, acestea fiind introduse de utilizator dintr-o interfata grafica. In aceasta tema am invatat sa folosesc regular expressions. Ca dezvoltari ulterioare se pot crea functionalitati cum ar fi gasirea radacinii polinoamelor sau ridicarea la putere.

**7. Bibliografie**

* stackoverflow.com
* www.baeldung.com
* youtube - Web Dev Simplified
* https://regexr.com
* https://www.rexegg.com/