DOCUMENTATIE

TEMA 1

POLYNOMIAL CALCULATOR

NUME STUDENT: NEGRUT CIPRIAN

GRUPA: 302110

1. **Obiectivul temei**

Obiectivul principal al temei este realizarea unui Calculator Polinomail care sa aiba implementate urmatoarele operatii: adunare, scadere, inmultire, impartire, integrala si derivata.

Obiecvitele secundare sunt urmatoarele:

<> Implementarea clasei [ Main ]

- utilizata pentru rularea aplicatiei

<> Implementarea clasei [ GUI\_APP ]

- interfata utilizator, interactiunea utilizatorului cu aplicatia

<> Implementarea clasei [ Monomial ]

- utilizata pentru memorarea exponentului si coeficientului fiecarui monom

<> Implementarea clasei [ Polynomial ]

- utilizata pentru memorarea unui polinom

<> Implementarea clasei [ Operatii ]

- implementeaza operatiile [ adunare, scadere, inmultire, impartire, integrala, derivata ] pentru doua polinoame

<> Implementarea clasei [ Controller ]

- utilizata pentru a lega datele introduse de catre utilizator si logica din spatele aplicatiei, pentru a genera un raspuns dorit

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

Cadrul de cerinte functionale cuprinde: adunarea, scaderea, inmultirea, impartirea, integrala, derivata si resetul. Aceste cerinte ne ajuta sa executam diferite operatii intre doua polinoame sau operatii cu un singur polinom, cum este cazul integrala si derivata.

Use-case-ul: *adunare, scadere, inmultire, impartire*

<> Actorul principal: User-ul

<> Scenariul de succes principal:

1. Utilizatorul introduce in TextField-urile destinate pentru polinomul 1 si polinomul 2 doua polinoame dorite

2. Utilizatorul apasa pe butonul [adunare] [scadere] [inmultire] [impartire]

3. Aplicatia afiseaza pe ecran rezultatul in urma operatiei pe care utilizatorul a selectat-o celor doua polinoame

<> Secvente alternative: polinom incorect

- polinomul introdus de catre utilizator este incorect

- utilizatorului i se afiseaza pe ecran sub forma de eroare mesajul *“Polinomul introdus este incorect!”*

- la impartirea polinoamelor, daca cel de al doilea polinom este egal cu 0 apare eroarea: *“Eroare la impartirea polinoamelor!”*

Use-case-ul: *integrala, derivata*

<> Actorul principal: User-ul

<> Scenariul de succes principal:

1. Utilizatorul introduce in TextField-ul destinat pentru polinomul 1 polinomul dorit

2. Utilizatorul apasa pe butonul [integrala] [derivata]

3. Aplicatia afiseaza pe ecran rezultatul in urma operatiei pe care utilizatorul a selectat-o celor doua polinoame

<> Secvente alternative: polinom incorect

- polinomul introdus de catre utilizator este incorect

- utilizatorului i se afiseaza pe ecran sub forma de eroare mesajul *“Polinomul introdus este incorect!”*

Use-case-ul: *ordonarea polinomului*

<> Actorul principal: User-ul

<> Scenariul de succes principal:

1. Utilizatorul introduce in TexFild-urile destinate pentri polinomul 1 si polinomul 2 doua polinoame in ordine aleatoare a exponentilor

2. La apasarea butonului de catre utilizator, pentru efectuarea operatiei dorite asupra celor doua polinoame sau doar a unuia, aplicatia ordoneaza fiecare polinom in ordine descrescatoare in functie de exponenti

Use-case-ul: *reset*

<> Actorul pincipal: User-ul

<> Scenariul de succes principal:

1. Utilizatorul introduce in TexFild-urile destinate pentri polinomul 1 si polinomul 2 doua polinoame in ordine aleatoare a exponentilor

2. Utilizatorul apasa butonul [ RESET ]

3. Dupa apasarea butonului [ RESET ] de catre utilizator, listele de monoame din cei doi polinomi vor fi sterse

4. Textul din fiecare TextFild se va seta pe string-ul empty: “”;

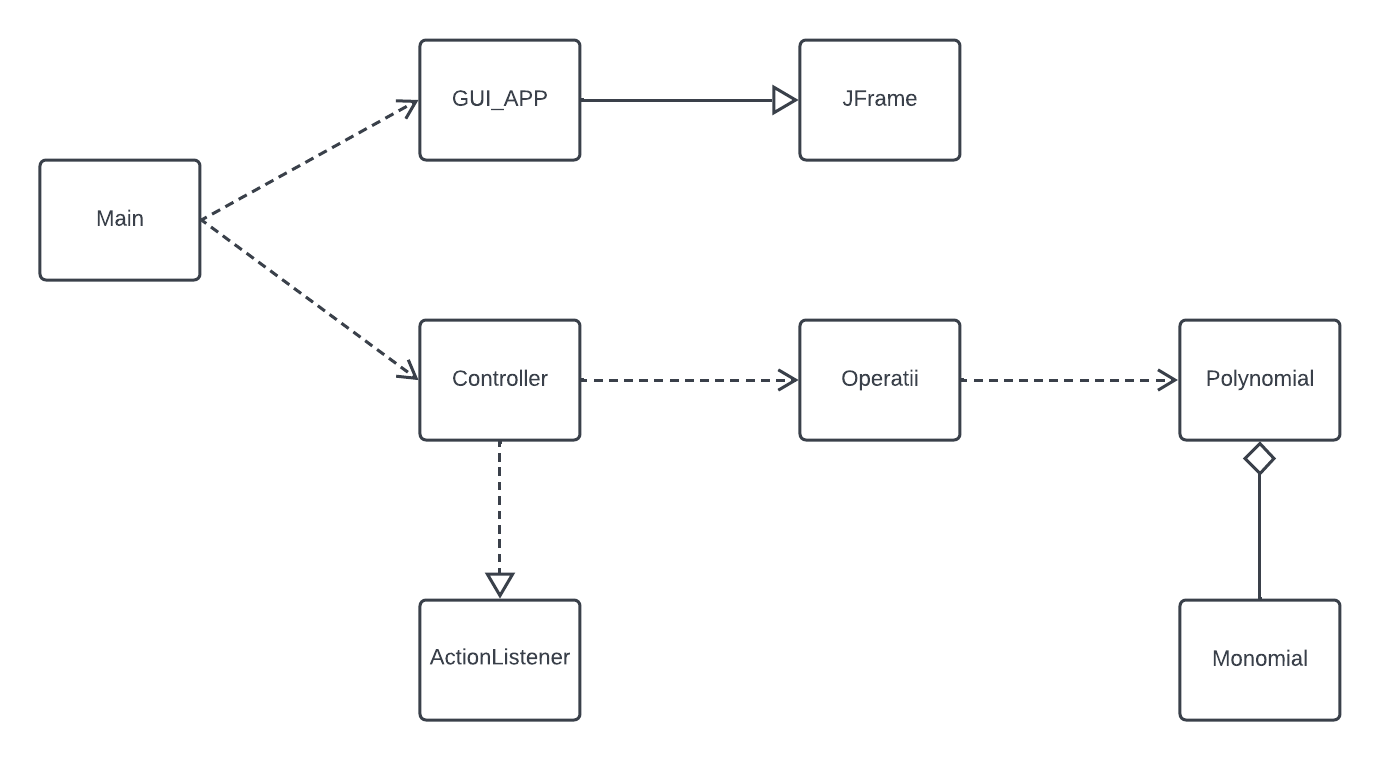
5. Pe ecran se va afisa mesajul *“Reset-ul s-a realizat cu succes!”*

<> Secvente alternative: reset-invalid

- daca utilizatorul incearca din prima sa reseteze aplicatia, pe ecran v-a aparea eroarea: “Reset invalid! Cel putin un polinom este null!”

- eroarea de reset-invalid apare doar atunci cand nu s-a efectuat cel putin o operatie cu polinomul 1, respectiv polinomul 2

1. **Proiectare**



Clasa Main:

<> Prin clasa Main se ruleaza aplicatia

Clasa GUI\_APP:

<> Interfata pentru utilizator, contine 2 TextFild-uri pentru polinomul 1, respectiv polinomul 2 si butoane pentru selectarea operatiei dorite, dar si pentru operatia de reset

<> Implementeaza ActionListener pentru fiecare buton asociat fiecarei operatii

Clasa Controller:

<> Realizeaza o legatura intre datele introduse de catre utilizator si logica aplicatiei

<> Contine clase care apeleaza fiecare operatie din clasa [ Operatii ]

<> Aici sunt tratate exceptiile din clasa [ Operatii ]

Clasa Operatii:

<> Implementeaza metode pentru fiecare operatie efectuata asupra celor doua polinoame

<> Arunca exceptii in cazul operatiei de impartire, respectiv reset

Clasa Polynomial:

<> Implementeaza metoda de conversie, dintr-un string intr-o lista de monoame

<> Arunca exceptii in cazul in care polinomul introdus este incorect

Clasa Monomial:

<> Contine variabile pentru exponent [ exp ] si pentru coeficient [ coef ]

1. **Implementare**

Clasa Controller:

<> Campuri:

- view: putem accesa metodele din clasa [ GUI\_APP ]

- operation: puteam accesa metodele din clasa [ Operatii ]

- poly1: reprezinta polinomul introdus in primul TextField

putem accesa metodele din clasa [ Polynomial ]

- poly2: reprezinta polinomul introdus in al doilea TextField

putem accesa metodele din clasa [ Polynomial ]

<> SubClase:

- AdunareListener: implementeaza ActionListener

~ in momentul apasarii butonului [ adunare ] de catre utilizator, lista de monoame din fiecare polinom, poly1 si poly2, este initializata cu monoamele introduse in TextField-ul asociat fiecarui polinom

~ apeleaza metoda [ adunare ] din clasa [ Operatii ] prin intermediul campului [ operation ]

~ seteaza, prin intermediul campului [ view ], textul din label-ul [ labelResult ] din clasa [ GUI\_APP ]

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de inroducerea unui polinom invalid in cele doua TextField-uri

- ScadereListener: implementeaza ActionListener

~ in momentul apasarii butonului [ scadere ] de catre utilizator, lista de monoame din fiecare polinom, poly1 si poly2, este initializata cu monoamele introduse in TextField-ul asociat fiecarui polinom

~ apeleaza metoda [ scadere ] din clasa [ Operatii ] prin intermediul campului [ operation ]

~ seteaza, prin intermediul campului [ view ], textul din label-ul [ labelResult ] din clasa [ GUI\_APP ]

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de inroducerea unui polinom invalid in cele doua TextField-uri

- InmultireListener: implementeaza ActionListener

~ in momentul apasarii butonului [ inmultire ] de catre utilizator, lista de monoame din fiecare polinom, poly1 si poly2, este initializata cu monoamele introduse in TextField-ul asociat fiecarui polinom

~ apeleaza metoda [ inmultire ] din clasa [ Operatii ] prin intermediul campului [ operation ]

~ seteaza, prin intermediul campului [ view ], textul din label-ul [ labelResult ] din clasa [ GUI\_APP ]

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de inroducerea unui polinom invalid in cele doua TextField-uri

- ImpartireListener: implementeaza ActionListener

~ in momentul apasarii butonului [ impartire ] de catre utilizator, lista de monoame din fiecare polinom, poly1 si poly2, este initializata cu monoamele introduse in TextField-ul asociat fiecarui polinom

~ apeleaza metoda [ impartire ] din clasa [ Operatii ] prin intermediul campului [ operation ]

~ seteaza, prin intermediul campului [ view ], textul din label-ul [ labelResult ] din clasa [ GUI\_APP ]

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de inroducerea unui polinom invalid in cele doua TextField-uri

- IntegralaListener: implementeaza ActionListener

~ in momentul apasarii butonului [ integrala ] de catre utilizator, lista de monoame din polinomul poly1 si este initializata cu monoamele introduse in TextField-ul asociat polinomului 1

~ apeleaza metoda [ integrala ] din clasa [ Operatii ] prin intermediul campului [ operation ]

~ seteaza, prin intermediul campului [ view ], textul din label-ul [ labelResult ] din clasa [ GUI\_APP ]

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de inroducerea unui polinom invalid in cele doua TextField-uri

- DerivataListener: implementeaza ActionListener

~ in momentul apasarii butonului [ integrala ] de catre utilizator, lista de monoame din polinomul poly1 si este initializata cu monoamele introduse in TextField-ul asociat polinomului 1

~ apeleaza metoda [ derivata ] din clasa [ Operatii ] prin intermediul campului [ operation ]

~ seteaza, prin intermediul campului [ view ], textul din label-ul [ labelResult ] din clasa [ GUI\_APP ]

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de inroducerea unui polinom invalid in cele doua TextField-uri

- ResetButtonListener: implementeaza ActionListener

~ in momentul apasarii butonului [ RESET ] de catre utilizator, se seteaza textul label-ului [ labelResult ] din clasa [ GUI\_APP] cu textul returnat de catre metoda [ resetButtonOp ] din clasa [ Operation ]

~ seteaza textul din TextField-urile, asociatea polinoamelor, cu valoarea empty “”

~ trateaza exceptia legata de numberFormat

~ trateaza exceptia legata de resetarea cand cel putin un polinom este null

Clasa Operatii:

<>Metode:

- *resetElementFundInEachMonomOf:* reseteaza campul [ found ] din fiecare monom, pentru polinomul cu care se apeleaza functia

- *adunare:* returneaza un polinom care contine valoarea adunarii a doua polinoame

- *scadere:* returneaza un polinom care contine valoarea scaderii a doua polinoame

- *inmultireMonomialAndPolynomial:* returneaza un polinom care contine valoarea inmultirii dintre un monom si un polinom

- *inmultire:* returneaza un polinom care contine valoarea inmultirii a doua polinoame

- *impartire:* returneaza un string care este compus dintr-un polinom care reprezinta catul impartirii [ polyReturnCat ] si un polinom care contine restul impartirii [ poly1 ]

- *integrala:* returneaza un polinom care contine valoarea integrarii unui polinom

- *derivata:* returneaza un polinom care contine valoarea derivarii unui polinom

Clasa Monomial:

<> Campuri:

- exp: contine exponentul unui monom

- coef: contine coeficientul unui monom

- found: folosit pentru a vedea daca un monom a fost folosit sau nu

<> Constructori:

- Monomial:

~ fara parametrii: construieste un obiect de tipul Monomial cu campurile [ exp, coef, found ] neinitializate

~ cu parametrii: construieste un obiect de tipul Monomial cu campurile [ exp, coef, found ] initializate cu cele primite ca si parametrii

<> Metode:

- *isFound:* returneaza valoarea lui found

- *setFound:* seteaza valoarea lui found

- *getExp:* returneaza valoarea lui exp

- *setExp:* seteaza valoarea lui exp

- *getCoef:* returneaza valoarea lui coef

- *setCoef:* seteaza valoarea lui coef

- *compareTo:* compara doua monoame intre ele dupa exp

Clasa Polynomial:

<> Campuri:

- listOfMonom: reprezinta lista de monoame pentru fiecare polinom

<> Constructori:

- Polynomial:

~ fara parametrii: initializeaza [ listOfMonom ] cu o lista goala

~ cu parametrii: initializeaza [ listOfMonom ] cu lista primita ca parametru

<> Metode:

- *addMonomToListOfMonom:* adauga monomul primit ca parametru in [ listOfMonom ]

- *checkStringForErrors:* verifica daca polinomul introdus de catre utilizator este unul valid, returneaza true / false

- *createMonomFromString:* creaza un polinom din string-ul scris de utilizator in TextField-urile destinate polinoamelor

- *sortPolynomial:* sorteaza lista de monoame dintr-un polinom in ordinea descrescatoarea a exponentilor

- *toString:* afisarea draguta a polinomului

Clasa Main:

<> Metode:

- *main:* ruleaza aplicatia

Clasa GUI\_APP [ interfata utilizator ]:

<> Campuri:

- adunareButton

- scadereButton

- inmultireButton

- impartireButton

- integralaButton

- derivataButton

- textPoly1

- textPoly2

- labelResult

- RESETButton

<> Metode:

- sunt generate getter si setter pentru fiecare buton, TextField, label

- implementeaza ActionListener pentru fiecare buton

- *displayErrorMessage:* afiseaza mesajul de eroare, primit ca si parametru, pe ecran

*Clasa OperatiiTest:*

<> Campuri:

- operation: se pot accesa toate metodele din clasa [ Operatii ]

<> Metode:

- implementeaza o metoda de testare pentru fiecare metoda din clasa [ Operatii ]

1. **Rezultate**

Rezultate in urma testarii cu Junit:

<> Scenariul de adunare:

- poly1 are valoarea [ x^2+2 ]

- poly2 are valoarea [ 2x ]

- rezultatul trebuie sa fie [ +x^2.0+2.0x+2.0 ]

<> Scenariul de scadere:

- poly1 are valoarea [ x^2+2 ]

- poly2 are valoarea [ 2x ]

- rezultatul trebuie sa fie [ +x^2.0-2.0x+2.0 ]

<> Scenariul de inmultire:

- poly1 are valoarea [ x^2 ]

- poly2 are valoarea [ 2x ]

- rezultatul trebuie sa fie [ +2.0x^3.0 ]

<> Scenariul de impartire:

- poly1 are valoarea [ x^2+2 ]

- poly2 are valoarea [ x+1 ]

- rezultatul trebuie sa fie [ +x-1] [ +3.0 ]

<> Scenariul de integrala:

- poly1 are valoarea [ 6x^2+2x+1 ]

- rezultatul trebuie sa fie [ +2.0x^3.0+x^2.0+x ]

<> Scenariul de derivata:

- poly1 are valoarea [ 6x^2+2x+1]

- rezultatul trebuie sa fie [ +12.0x+2.0 ]

1. **Concluzii**

In concluzie, de-a lungul implementarii acestei aplicatii, m-am lovit de unele lucruri pe care nu le stiam in java, dar cu putin research am reusit sa le rezolv si sa si invat in acelasi timp lucruri noi. Aceasta tema m-a ajutat sa inteleg mai bine cum se folosesc elementele de interfata-utilizator si sa stapanesc mai mult clasele si relatiile dintre clase si metode.

1. **Bibliografie**

Site-urile folosite: <https://www.geeksforgeeks.org/>