

**FACULTATEA: Automatica si Calculatoare**

**SPECIALIZAREA: Calculatoare si tehnologia informatiei**

**DISCIPLINA: Tehnici de Programare**

**AN UNIVERSITAR: 2019/2020**

**PROIECT: Calculator Polinomial**

**Realizat de :**

Ghiura Darius

**Grupa:** 30227

**CUPRINS**

1. **Obiectivul temei**
2. **Analiza problemei, modelare, scenarii, cazuri de utilizare**
3. **Proiectare**
4. **Implementare**
5. **Rezultate**
6. **Concluzii**
7. **Bibliografie**
8. Obiectivul temei

Obiectivul temei este proiectarea si implementarea unui calculator care poate efectua operatiile de baza pe polinoame cu o variabila si coeficienti intregi.

Calculatorul are o interfata grafica dedicata unde utilizatorul poate introduce polinoamele si selecta operatia dorita. Operatiile care au fost implementate sunt: adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea.

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

In matematica, un polinom este o expresie formata din una sau mai multe variabile si coeficienti, folosind doar operații de adunare, scădere, înmulțire

si valori intregi, pozitive ale exponentilor variabilelor.

Un polinom este format din mai multi termeni numiti monoame. Un monom este o componenta alcatuita dintr-un coeficient inmultit cu una sau mai multe variabile si un exponent intreg pozitiv (De exemplu 3x^2 este un monom).

Gradul unui polinom este egal cu cel mai mare dintre exponentele monoamelor din care este alcatuit.

Aplicatia a fost proiectata pentru polinoame cu o variabila si coeficienti intregi.

Adunarea a doua polinoame este de fapt adunarea coeficientilor monoamelor care au acelasi exponent. Aceste monoame sunt introduse intr-un polinom nou care va fi polinomul rezultat dupa efectuarea operatiei de adunare.

Scaderea a doua polinoame se bazeaza pe acelasi principiu ca si adunarea polinoamelor, rezultatul operatiei fiind tot un nou polinom in care coeficientii monoamelor care au acelasi exponent se scad.

Inmultirea a doua polinoame este realizata prin inmultirea fiecarui monom dintr-un polinom cu fiecare monom din celalalt polinom. Regula pentru inmultire este : coeficientii se inmultesc, iar exponentii se aduna.

Impartirea a doua polinoame se face folosind algoritmul de „long division” care foloseste teorema impartirii cu rest.

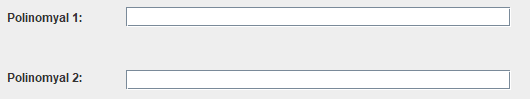
Derivarea unui polinom se face prin derivarea fiecarui monon in parte, monomul rezultat avand coeficientul - vechiul coeficient inmultit cu exponentul - iar exponentul va fi vechiul exponent – 1.

Integrarea unui polinom se face prin integrarea fiecarui monon in parte, monomul rezultat avand coeficientul – vechiul coeficient impartit cu (exponentul + 1) – iar exponentul va fi vechiul exponent + 1.

Polinoamele introduse in interfata grafica trebuie sa respecte urmatorul format : „ax^b” , unde a reprezinta coeficientul monomului si este un numar intreg (daca a = 1, acesta poate sa nu fie scris , iar daca a = -1 se poate scrie doar „-”), iar b este exponentul monomului care este un numar intreg pozitiv (daca b = 1 se poate scrie doar x, fara „^”, iar daca b = 0 se scrie doar coeficientul).

Scenariul aplicatiei este reprezentat de introducerea polinoamelor si selectarea operatiei dorite.

**Cazuri de utilizare :**

* **Adunare, scadere, inmultire, impartire**
* Se introduc polinoamele in zonele respective, respectand formatul specificat anterior;
* In caz de nerespectare a formatului pe ecran va aparea o fereastra de eroare cu mesajul „Incorrect input”. O eroare va aparea si pentru introducerea polinomului „0” in cazul impartitorului la operatia de impartire, cu mesajul „Cannot divide by 0”.
* Se apasa butonul corespunzator operatiei dorite;



* Se apasa pe butonul „=” pentru afisarea rezultatului;



* Rezultatul operatiei va fi afisat in zona din stanga, iar pentru impartire in zona din dreapta va fi afisat restul;
* Daca nu se introduce un polinom acesta va fi considerat ca fiind 0;
* Pentru stergerea polinoamelor de intrare, se apasa butonul de CE (Clear Entry).
* **Derivare, integrare**

In cazul derivarii si integrarii, procesul de introducerea datelor, selectarea operatiei si afisarea rezultatului este acelasi, doar ca va fi luat in considerare doar polinomul introdus in zona „Polynomial 1”.

Se pot efectua oricate operatii fara a mai fi nevoie de introducerea datelor de intrare.

1. Proiectare

Proiectul a fost realizat dupa modelul **MVC (Model View Controller)** . Acest model presupune impartirea programului in trei parti :

* Model - Gestioneaza datele si logica aplicatiei;
* View - Este partea de interfata grafica si reda continutul modelului;
* Controller - Gestioneaza datele de intrare primite de la utilizator si apeleaza metodele potrivite din model si transmite la view ce sa afiseze.

Proiectul este impartit in trei pachete :

* Model – Contine clasele Monomial, Polynomial si PolynomialCalculatorOperations
* Controller – Contine clasa PolynomialCalculatorController
* View - Contine clasa PolynomialCalculatorView

Pe langa astea mai exista si clasa PolynomialCalculatorMVC unde se afla metoda main a proiectului.

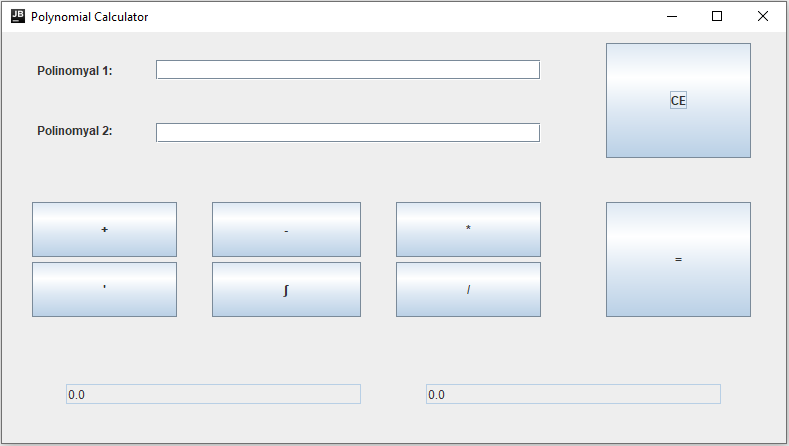
Din punct de vedere al structurilor de date, pentru implementarea unui polinom s-a folosit un ArrayList de monoame (ArrayList<Monomial>).

Clasele au fost proiectate respectand conceptul de programare orientata pe obiect de incapsulare. Variabilele instanta sunt declarate cu modificatorul de acces „private” ele fiind accesibile din exteriorul claselor doar prin gettere si settere.

Pentru extragerea si validarea monoamelor, respectiv a coeficientilor si exponentilor s-au folosit clasele **Pattern** si **Matcher** impreuna cu tehnica de REGEX (Regular Expression).

Pentru impartirea polinoamelor a fost folosit algoritmul de Polynomial long division.

Operatiile au fost implementate in clasa PolynomialCalculatorOperations din pachetul Model.

Interfata utilizator a fost implementata in clasa PolynomialCalculatorView, aceasta este alcatuita din trei paneluri grupate intr-un contentPanel. Exista 8 butoane (6 pentru operatii si 1 pentru egal si 1 pentru CE), 4 textField-uri pentru introducerea, respectiv afisarea polinoamelor si 2 label-uri.

**Diagrama UML**

1. Implementare

**Clasa : Polynomial**

Variabila instanta :

**private ArrayList<Monomoame> monomials = new ArrayList<Monomial>();**

Constructori :

**public Polynomial (String input)** – Acest constructor este folosit pentru creare polinomului dintr-un string. In el se folosesc clasele Pattern si Matcher pentru a desparti stringul in monoame. Regexul folosit este : „(-?[^-+]+)”; acesta cauta un caracter ‚-’ care poate exista sau nu, iar apoi cauta toate caracterele diferite de ‚-’ si ‚+’. Aceste caractere vor fi „coefx^exp”. Deci acest regex va returna un monom (de exemplu „2x^3”), iar acest string va fi folosit in constructorul din clasa monom pentru a separa coeficientul de exponent. Tot aici se adauga monoamele create in lista de monoame „monomials” si se sorteaza lista de monoame in ordine descrescatoare a exponentilor.

**public Polynomial (Monomial m)** – Adauga m listei de monoame;

**public Polynomial ()**;

Metode :

* **public ArrayList<Monomial> getMonomials() –** getter pentru lista de monoame;
* **public String toString() –** Folosit pentru afisarea polinomului;
* **public void sortPolynomial ()** – Pe langa sortarea polinomului, folosind metoda sort din clasa Collections, aceasta metoda verifica daca exista doua monoame care au acelasi exponent si le aduna coeficientii daca este cazul. Pentru a realiza acest lucru se parcurge lista folosind un Iterator si folosim un Monom m care reprezinta monomul dat de iterator si un Monom n care reprezinta monomul de la iteratia anterioara. Daca m si n au acelasi exponent, adunam coeficientul lui m in n si folosim metoda remove() din clasa Iterator pentru a sterge din lista monomul m. Daca m si n nu au acelasi exponent n primeste m si se trece la iteratia urmatoare. In plus, metoda scoate din lista monoamele care au coeficient 0, folosind metoda removeIf() din clasa ArrayList.

**Clasa : Monomial**

Aceasta clasa implementeaza interfata **Comparable** care este folosita pentru sortarea listelor de monoame.

Variabile instanta :

* **private double coef** – coeficientul monomului;
* **private int exp** – exponentul monomului;

Constructori :

* **public Monomial (double coef, int exp)** – Creeaza un monom prin setarea coeficentului si a exponentului;
* **public Monomial (String mon)** – Primeste un string din clasa Polynomial, care reprezinta un monom si ii extrage coeficientul si exponentul astfel : se creeaza un array de String-uri **String[] a** folosind **mon.split(„x\\^”);**
  + daca apelul acestei metode a produs un array de length == 2 asta inseamna ca monomul nostru are coeficient si exponent :
    - In continuare verificam daca primul element din array-ul de String-uri este egal cu „”. Daca da, asta inseamna ca monomul este de forma **„x^a”**, adica coeficientul = 1;
    - Verificam daca primul element din array-ul de String-uri este egal cu „-”, caz in care monomul are forma **„-x^a”** , adica coeficientul = -1;
    - Daca nu ne-am aflat in cazurile de mai sus inseamna ca avem un monom de forma „ax^b” si preluam coeficientul cu metoda **Integer.parseInt(a[0]);**
    - In final determinam exponentul folosind aceeasi metoda **Integer.parseInt(a[1]);**
  + daca apelul metodei nu a produs un array de length == 2 atunci verificam daca String-ul contine „x”, folosind **mon.contains(„x”)** :
    - Daca String-ul este egal cu **„x”** atunci coeficientul e 1
    - Daca String-ul este egal cu **„-x”** atunci coeficientul e -1
    - Altfel inseamna ca String-ul este de forma **„coefx\*x”** si pentru a gasi coeficientul se face **mon.split(„x”)** si se foloseste **Integer.parseInt(mon.split(„x”)[0]);**
    - In final, exponentul primeste valoarea 1
  + altfel inseamna ca monomul este doar un coeficient care este determinat folosind Integer.parseInt(mon), iar exponentul primeste valoarea 0

Metode :

* **public double getCoef();**
* **public int getExp();**
* **public void setCoef(double coef);**
* **public String toString() –** Folosit pentru afisarea monoamelor;
* **public int equals(Object o) –** Folosit pentru verificarea daca doua monoame sunt egale, adica au coeficient si exponent egal;
* **public int compareTo(Object o) –** Folosit pentru sortarea listelor de monoame;

**Clasa : PolynomialCalculatorOperations**

In aceasta clasa sunt implementate toate operatiile pe polinoame.

Variabile instanta :

* **private Polynomial result -**  Rezultatul operatiilor;
* **private Polynomial remainder –** Restul impartirii;

Metode :

* **public void reset()** – Reseteaza result si remainder;
* **public Polynomial getResult() ;**
* **public Polynomial getRemainder();**
* **public Polynomial addPolynomials(Polynomial p1, Polynomial p2) –** Adunarea polinoamelor se face prin inserarea tuturor elementelor din listele de monoame ale lui p1 si p2 in lista din polinomul de rezultat; apoi se apeleaza metoda sortPolynomial din clasa Polynomial care sorteaza, aduna coeficientii monoamelor cu acelasi exponent si scoate din lista monoamele cu coeficientul = 0. La final este returnat polinomul rezultat.
* **public Polynomial subPolynomials(Polynomial p1, Polynomial p2) –** Scaderea polinoamelor se face prin inserarea monoamelor a primului polinom in polinomul rezultat, inserarea monoamelor cu semnul coeficientului schimbat ale polinomului p2, iar apoi se apeleaza metoda sortPolynomial din clasa Polynomial care sorteaza, aduna coeficientii monoamelor cu acelasi exponent si scoate din lista monoamele cu coeficientul = 0. La final este returnat polinomul rezultat.
* **public Polynomial mulPolynomials(Polynomial p1, Polynomial p2) –** Inmultirea polinoamelor se face parcurgand cele două polinoame, iar în polinomul de rezultat se adauga un monom nou care este format prin înmulțirea fiecărui monom din p1 cu fiecare monom din p2. Coeficienții monoamelor se înmulțesc, iar exponentii acestora se adună. Este apelata apoi metoda sortPolynomial din clasa Polynomial care sorteaza, aduna coeficientii monoamelor cu acelasi exponent si scoate din lista monoamele cu coeficientul = 0. La final este returnat polinomul rezultat.
* **public void divPolynomials(Polynomial p1, Polynomial p2) -**
* **public Polynomial derivPolynomials(Polynomial p) -** se parcurge polinomul p si se adaugă în polinomul rezultat monoame noi care au ca si coeficient exponentul și coeficientul monomului din p inmultite, iar exponentul decrementat cu 1. La final este returnat polinomul rezultat.
* **public Polynomial integrPolynomials(Polynomial p) -** se parcurge polinomul p si se adaugă în polinomul rezultat monoame noi care au ca si coeficient coeficientul monomului din p impartit cu (exponentul + 1), iar exponentul incrementat cu 1. La final este returnat polinomul rezultat.

**Clasa : PolynomialCalculatorView**

Clasa in care este implementata interfata utilizator. Aceasta extinde clasa JFrame, iar in constructor se creeaza interfata grafica, care este realizata folosind layout-ul GridBag cu GridBagConstraints pentru toate elementele.

Variabile instanta :

* **private JTextField tfPol1 = new JTextField(); -**  TextField pentru polinomul 1;
* **private JTextField tfPol2 = new JTextField(); -** TextField pentru polinomul 2;
* **private JButton[] buttonsOp = new JButton[6];–** Array de dimensiune 6, pentru butoanele de operatii;
* **private JButton buttonEquals = new JButton("=");–** Butonul de egal;
* **private JButton buttonClear = new JButton("CE");** – Butonul de Clear Entry;
* **private JTextField tfRes = new JTextField(); –** TextField pentru rezultat;
* **private JTextField tfRes1 = new JTextField(); –** TextField pentru rest;

Metode :

* **public void resetEntry() –** Reseteaza TextField-urile polinoamelor de intrare
* **public String getPol1()**
* **public String getPol2()**
* **public void setRes(String res, String res1) –** Seteaza rezultatul si restul in TextField
* **public void addOpListener(ActionListener opAl) –** Seteaza listener pentru butoanele de operatii, parcurgand array-ul de JButton. Listener-ul este acelasi.
* **public void addEqualsListener(ActionListener eqAl) -**  Seteaza listener pentru butonul de egal
* **public void addClearListener(ActionListener ceAl) -** Setează listener pentru butonul de Clear Entry
* **public void showError(String errMessage) –** Afiseaza pe ecran o fereastra noua cu un mesaj de eroare