Calculator Polinomial

*Logo, company name

Description automatically generated*

**Profesor: Florin Oniga**

**Asistent profesor: Alexandru Rancea**

**Student: Stratulat Ildiko Vivien**

**Grupa : 30221**

**1.Obiectivul temei**

Implementarea unui sistem de procesare a polinoamelor, considerând că acestea au coeficienți reali și să efectueze operații cu aceștia.

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

* **Analiza problemei:**

În matematică un polinom este o expresie construită dintr-una sau mai multe variabile și constante, folosind doar operații de adunare, scădere, înmulțire și ridicare la putere.

Polinoamele sunt construite din termeni numiți monoame, care sunt alcătuite dintr-o constantă, adică coeficientul, înmulțită cu una sau mai multe variabile, fiecare variabilă poate avea un exponent.

Astfel, suma a mai multor monoame reprezintă un polinom, de exemplu: x2+2x-1, acesta are trei

termeni cu trei exponenți diferiți: primul are gradul doi, al doilea are gradul unul, iar ultimul are gradul 0.

Totodată aceste polinoame se pot reprezenta și astfel:

Fiecare pereche (ai,i) corespunde termenului aixi , de exemplu polinomul -2x^3+3x+5, poate fi reprezentat prin secvența (-2,3),(3,1),(5,0).

Secvența conține perechea (coeficient,grad) de la fiecare monom de la polinomul dat. Folosind această reprezentare se pot executa cele mai comune operații asupra polinoamelor cum ar fi: adunare, scădere, înmulțire, împărțire, derivare și integrare.

* **Modelarea problemei:**

Utilizatorul va avea posibilitatea de a folosi funcțiile calculatorului polinomial prin intermediul interfeței grafice, unde va introduce polinoamele și va putea alege ce operații să efectueze cu acestea:

* Adunarea a două polinoame
* Diferența a două polinoame
* Înmulțirea a două polinoame
* Derivarea unui polinom
* Integrarea unui polinom

Rezultatul obținut în urma efectuării operației alese va fi afișat în interfața grafica

**3.Scenarii și cazuri de utilizare:**

Un caz de utilizare reprezintă o colecție de scenarii posibile, referitoare la comunicarea între aplicație și utilizatori, caracterizate cu anumite scopuri. Cazurile de utilizare arată ce trebuie să facă sistemul și nu cum.

Graphical user interface

Description automatically generatedDeoarece cazurile de utilizare sunt puternic conectate cu pașii utilizatorului am încercat să fac interfața grafică cât mai user-friendly:

Utilizatorul va introduce cele două polinoame în căsuțele text, dacă va dori să execute o anumită operație cu aceste polinoame tot ce trebuie să facă este să apese butonul corespunzător, iar în urma calculului rezultatul va apărea în căsuța text „Result:”

În caz că utilizatorul apasă pe butonul de adunare/scădere/înmulțire fără să introducă ambele polinoame va apărea o ferestră noua care îi va solicita acestuia să introducă toate datele:

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedIar în cazul în care apasă pe butonul de integrare sau derivare fără să introducă un polinom în prima căsuță text de asemenea va apărea o fereastră care îi solicită să introducă polinomul:

**3.Proiectare:**

* *Diagrame*
* Diagrame Use Case

Diagram

Description automatically generated

Diagram

Description automatically generated

Use Case-ul prezintă un actor, care în cazul nostru este utilizatorul care interacționează cu aplicația. Acesta poate realiza numaroase operații cu cei doi polinomi, cum ar fi: adunare, scădere, înmulțire, derivare,integrare.

* Diagrama claselor

A picture containing graphical user interface

Description automatically generated

-Diagramele de clase sunt folosite în modelarea orientată pe obiect pentru a a descrie structura statică a sistemului, modului în care este el structurat

-Oferă o notație grafică pentru reprezentarea: claselor, entități ce au caracteristici comune și a relațiilor dintr două sau mai multe clase.

* **Structuri de date**

Structurile de date care au fost folosite în cadrul acestui proiect sunt ori tipuri de date primitive, double și int, ori un tip mai complex: ArrayList și obiectele create de mine, Monom și Polinom, cel din urmă reprezentând chiar o listă de monoame si atributele acesteia

Am ales să folosesc ArrayList în locul obișnuitelor Array-uri din cauza funcționalității și perfomanței acestuia, fiind mai eficient în momentul în care dorim să accesăm elementele lui. Totodată, operația de adăugare de termeni este mai ușoară la ArrayList-uri față de cele clasice la care trebuie să predefinim o anumită lungime.

* **Pachete**

Pachetele sunt folositoare în menținerea proiectului sub o formă mai organizată, fapt ce facilitează înțelegerea acestuia. Prin pachete se înțelege gruparea a multiple clase și interfețe.

Astfel, în programarea orientată pe obiecte se folosește în zilele de astăzi metodologia MVC(model-view-controller) pentru a gestiona eficient interfețele grafice cu modele de date.

MVC este considerat de mulți specialiști în domeniu drept un pattern folositor în cazul reutilizării codului și datorită căruia dezvoltarea aplicațiile cu interfețe grafice pentru utilizatori durează mai puțin.

Acest MVC este alcătuit din trei mari componente care vor fi utilizate în dezvoltare software:

* *MODELUL*: -reprezintă structura logică a datelor într-o aplicație software, acesta nu deține nicio informație despre interfața grafică.
* *VIEW-UL*: -alcătuit dintr-o mulțime de clase care reprezintă diferite elemente în interfața grafică, elementele vizible pentru utilizator și cu care poate interacționa, acestea fiind: butoane, textfield-uri, combo box-uri ș.a.m.d
* *CONTROLLER-UL*: reprezintă legătura din model și view, are drept scop permiterea comunicării dintre clasele din model și din view

**4.Implementare:**

Datorită beneficiilor utilizării metodologiei MVC, am decis să o folosesc în proiectul meu, astfel acesta conține următoarele pachete:

* App – conține o singură clasă, în interiorul căreia se afla metoda principala main()
* Model – conține clasele care rezolvă problema propusă (operațiile pe polinoame)
* Monom
* Polinom
* CalcModel
* View - conține o singură clasă care reprezintă interfața grafică pentru utilizator
* Controller -creează legătura dintre model și view

**Descrierea claselor:**

**-**în cadrul fiecărei clase am aplicat una din paradigmele principale ale programării orientate pe obiecte: încapsularea, am declarat fiecare variabilă instanță din clase drept private, astfel aceasta fiind vizibilă doar în cadrul acestei clase.

* **Clasa Monom**

-un polinom este alcătuit din mai mulți termeni, care în domeniul matematicii sunt numiți „monoame”

-clasa are două variabile instanță: degree - de tip int

coefficient- de tip double

Constructor:

Public Monom(int degree,double coefficient) -acesta inițializează câmpurile monomului cu valorile transmise

Metode:

-public int getDegree( ) – returnează exponentul monomului

-public double getCoefficient( ) – returnează coeficientul monomului

-public setDegree(int degree) – setează valoarea câmpului degree cu parametrul transmis

-public setCoefficient(double coefficient)- setează valoarea câmpului coefficient cu parametrul transmis

-public void printMonom( )- afișează monomul în consolă (va fi folosit ulterior în afișarea polinoamelor)

-public String toString( )-transformă monomul într-un string

* **Clasa Polinom**

-aceasta are o singură variabilă instanță care este de fapt o listă, mai exact un ArrayList

alcătuit din obiecte de tip Monom, astfel stochez polinomul

Constructor:

Public Polinom( ): constructorul standard

Metode:

-public ArrayList<Monom> getPoly( )-returnează polinomul drept un ArrayList alcătuit din monoame

-public void printPoly( )-afișează în consolă polinomul, folosindu-se de metoda de afișare a monoamelor

-public Monom getDegree(int degree )-returnează monomul care are gradul transmis ca parametru

-public void sortPoly( )-sortează termenii polinomului în ordine descrescătoare după grad

* **Clasa CalcModel**

-are drept variabile instanță rezultatele obținute în urma efectuării operațiilor asupra polinoamelor: suma, diferența, produsul, derivata, integrala.

Metode:

-public Polinom replicatePolinom(Polinom poly)- are rolul de a face o copia a unui polinom

-public void add(Polinom poly1,Polinom poly2) -face adunarea a două polinoame

-public void substract(Polinom poly1,Polinom poly2) – calculează diferența a două polinoame

-public void integrate(Polinom poly1) -calculează integrala polinomului transmis

-public void derivation(Polinom poly1) – calculează derivata polinomului transmis

-public void multiplication(Polinom poly1,Polinom poly2) – calculează produsul a două polinoame

-public Polinom getSum( )-returnează suma calculată, care va fi apoi folosită pentru afișarea rezultatului în interfața grafică

-public Polinom getDifference( )-returnează diferența calculată, care va fi apoi folosită pentru afișarea rezultatului în interfața grafică

-public Polinom getProduct( )-returnează produsul calculat, care va fi apoi folosită pentru afișarea rezultatului în interfața grafică

-public Polinom getIntegrated( )-returnează integrala calculată, care va fi apoi folosită pentru afișarea rezultatului în interfața grafică

-public Polinom getDerivated( )-returnează derivata calculată, care va fi apoi folosită pentru afișarea rezultatului în interfața grafică

* **Clasa CalcView**

-această clasă extinde Jframe, datorită acestui lucru este mai ușor să setez anumite proprietăți a interfeței grafice, cum ar fi mărimea, vizibilitatea, DefaultCloseOperation, layout-ul și să adaug elemente grafice (butoane, textfield-uri,label-uri)

-conține mai multe variabile instanță ce reprezintă elementele grafice ale interfeței grafice, cum ar fi butoane, căsuțe text.

Constructor:

-public CalcView( ) – aici setez toate proprietățile menționate mai sus și adaug interfeței grafice elementele grafice

Metode:

-public void setResult(String result) – setează rezultatul de pe interfață

-public String getFirstPoly( )-returnează polinomul introdus în prima casetă text, care este de tip String

-public String getSecondPoly( )-returnează polinomul introdus în a doua casetă text, care este de tip String

-următoarele metode sunt metode de ActionListener-uri pentru butoanele de pe interfața grafică

-public void addAddButtonActionListener(ActionListener actionListener) -setează butonul de adunare drept ActionListener

-public void addSubstractButtonActionListener(ActionListener actionListener) -setează butonul de scădere drept ActionListener

-public void addMultyButtonActionListener(ActionListener actionListener)- setează butonul de înmulțire drept ActionListener

-public void addIntegrateButtonActionListener(ActionListener actionListener)- setează butonul de integrare drept ActionListener

-public void addDerivateButtonActionListener(ActionListener actionListener)- setează butonul de derivare drept ActionListener

* **Clasa CalcController**

-conține legătura dintre model și view

-aceasta gestionează datele în model și actualizează view-ul când are loc vreo schimbare

-are următoarele variabile instanță :

* CalcView theView
* CalcModel theModel
* Polinom polynom1
* Polinom polynom2

Constructor:

-public CalcController(CalcView theView,CalcModel theModel)

Metode:

-public Polinom getInput(String input) – preia polinomul scris în căsuța text

-public Polinom createPoly( String[] splitString) – creează un polinom din stringul de la input

* **Clasa Calc**

-conține metoda principală main care pornește aplicația

**Algoritmi:**

* **Adunare**

-suma a două polinoame se obține prin adunarea coeficienților termenilor care au același grad, de exemplu:

 (a_2x^2+a_1x+a_0)+(b_1x+b_0)=a_2x^2+(a_1+b_1)x+(a_0+b_0) 

* **Scădere**

-scăderea a două polinoame se obține prin scăderea coeficienților termenilor care au același grad, de exemplu:



* **Înmulțirea**

-înmulțirea a două polinoame se obține prin înmulțirea termen cu termen



* **Integrarea**

-integrarea unui polinom presupune mărirea gradului termenilor acestuia cu o unitatea și împărțirea lor cu acest grad

* **Derivarea**

-derivarea unui polinom presupune înmulțirea fiecărui termen cu gradul său, apoi scăzând o unitate din grad

**Interfața grafică**

-are rolul de a conecta utilizatorul cu aplicația

-acest utilizator poate să introducă date, în cazul de față unul sau două polinoame și să execute operații cu acestea

-fiecare polinom introdus trebuie să respecte o anumită structură: să fie alcătuit din unul sau mai mult termeni, iar termenii la rândul lor trebuie să aibă o structură specifică

-un termen trebuie să aibă forma: semn+coeficient+x^+grad

-nu contează ordinea acestor termeni (după grad) aceasta se va face automat de către o metodă care sortează polinomul rezultat

-

Graphical user interface

Description automatically generated

**5.Rezultate:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OPERAȚIE** | **INPUT** | **REZULTAT AȘTEPTAT** | **REZULTATUL EFECTIV** | **PASS/FAIL** |
| Adunare | -2x^3+3x+5; x+2 | -2.0^3+4.0x+7.0 | -2.0^3+4.0x+7.0 | PASS |
| Scădere | -2x^3+3x+5; x+2 | -2.0x^3+2.0x+3.0 | -2.0x^3+2.0x+3.0 | PASS |
| Înmulțire | -2x^3+3x+5; x+2 | -2.0x^4-4.0x^3+3.0x^2+11.0x+10.0 | -2.0x^4-4.0x^3+3.0x^2+11.0x+10.0 | PASS |
| Derivare | -2x^3+3x+5 | -6.0x^2+3.0 | -6.0x^2+3.0 | PASS |
| Integrare | -2x^3+3x+5 | -0.5x^4+1.5x^2+5.0x | -0.5x^4+1.5x^2+5.0x | PASS |

-pentru testarea metodelor pentru operațiile asupra polinoamelor am folosit Junit

**Junit:**

Reprezintă un framework de Unit Testing pentru Java și folosirea acestuia are următoarele avantaje:

* Îmbunătățirea vitezei de scriere a codului și creșterea calității acestuia
* Clase test sunt ușor de scris și de modificat
* Clasele de test pot fi rulate automat, rezultatele fiind vizibile imediat
* Clasele de test măresc încrederea programatorului în codul sursă scris și îi permit să urmăarească mai ușor cerințele de implementare ale proiectului.

Graphical user interface

Description automatically generated**-Adunare:**

Graphical user interface

Description automatically generated**-Scădere:**

Graphical user interface

Description automatically generated-**Înmulțire:**

Graphical user interface, application

Description automatically generated-**Derivare:**

Graphical user interface

Description automatically generated-**Integrare:**

**6.Concluzii:**

Acest proiect a reprezentat un exercițiu absolut necesar pentru aprofundarea noțiunilor studiate în primul semestru despre Programarea orientată pe obiecte, pe parcursul realizării acestuia am întâmpinat anumite dificultăți, dar datorită acestora am fost nevoită să mă documentez mai mult, astfel înțelegând mult mai bine aceste noțiuni.

O astfel de noțiune a fost MVC-ul despre care doar am auzit în primul semestru și nu înțelegeam neapărat care este rostul în a împărți în atâtea clase lucruri care se puteau la fel de bine scrie într-una, dar am înțeles acum că astfel se obține o organizare mult a bună a proiectului.

Iar noțiunea despre care doar am auzit, dar și mai vag este Regex-ul, dar din câte am observat este vital în cadrul aplicațiilor cu interfață grafică.

Iar cel din urmă, testarea unitară, adică JUnitul, la fel, am auzit câte ceva în primul semestru, dar nu am folosit niciodată până acum, iar acum am văzut că datorită acestuia realizarea unor teste este mult mai rapidă.

**Dezvoltări ulterioare:**

Din câte se poate observa, am implementat majoritatea operațiilor fundamentale a polinoamelor în afară de împărțire, care din punctul meu de vedere necesită un algoritm mai complex decât celelalte operații.

Astfel pentru dezvoltări ulterioare, aș sugera implementarea și acestei operații pe lângă altele cum ar fi: calcularea rădăcinilor unui polinom, scrierea sub formă canonică a acestuia.

Totodată, aș stiliza puțin mai mult interfața grafică pentru a arăta mai estetic dar să fie și mai user-friendly.

**7.Bibliografie:**

1. [www.wikipedia.org](http://www.wikipedia.org)
2. [www.geeksforgeeks.com](http://www.geeksforgeeks.com)
3. [www.w3school.com](http://www.w3school.com)
4. [www.stackoverflow.com](http://www.stackoverflow.com)
5. Cursurile de la POO