UNIVERSITATEA DE AUTOMATICA SI CALCULATOARE

DOCUMENTATIE TEMA 1
CALCULATOR DE POLINOAME

Nume si prenume: Timis Iulia Georgeana

Grupa: 30226

Profesor laborator: Dorin Vasile Moldovan

CUPRINS

1.Probleme si solutia problemei

2. Obiecive

3. Studiul problemei, modelare, scenarii, cazuri de utilizare

4. Proiectare

5. Implementare

6. Rezultat

7. Concluzii

8. Bibliografie

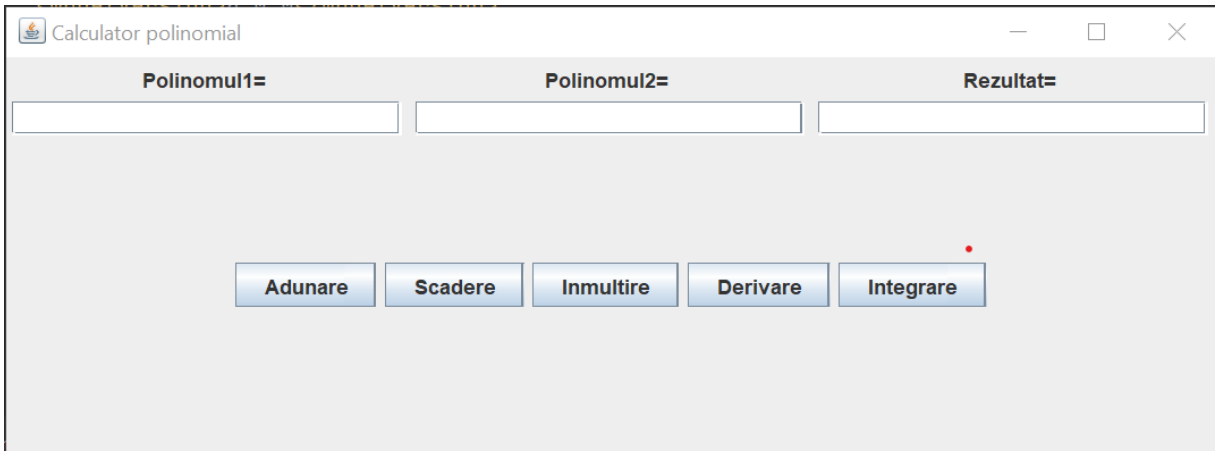
1. Probleme si solutia problemei:

Problema: Rezolvarea diverselor operatii pentru un polinom este dificila si necesita mult timp.

Solutia: Pentru a ne usura munca este recomandata folosirea unui “Calculator de polinoame”

Putem calcula foarte usor o operatie : (Ex: 3x^4+x^3+2x+5x^0) fara a avea probleme cu gresirea calculelor, si dupa reluarea exercitiului.

Un exemplu de interfata pentru un calculator de polinoame:



2. Obiective:

Obiectivul acestui laborator este de a implementa si a proiecta un sistem de calcul al polinoamelor.

Calculatorul de polinoame poate efectua diferite operatii: aundare, scadere, inmultire, derivare si integrare. Datele sunt introduse de utilizator, iar rezultatul va fi generat , dupa efectuarea calculelor, pe ecran in casuta atribuita acestuia.

Pentru interfata, am folosit tehnica “Model-View-Controller”, pentru a crea un aspect cat mai placut si usor de utilizat pentru orcine ar vrea sa isi rezolve corect si in scurt timp exercitiile cu polinoame.

3. Studiul problemei, modelare, scenarii, cazuri de utilizare

Programul „Calculator polinoame” este un program de consolă pur, care este folosit pentru a adăuga, scădea, înmulți și împărți polinoame până la n grade.

Acesta permite utilizatorului sa incarce polinoame de la tastatura, urmand ca sa se rezolve diverse operatii matematice dorite de utilizator.

Ce reprezinta un polinom ?

In matematica, un polinom este definit ca fiiind o expresie alcatuita din mai multe variabile si constante, folosind doar operatii de adunare, scadere, inmultire, impartire si ridicare la putere.

Ce reprezinta un monom ?

Monoamele sunt termenii din care este alcatuit un polinom si sunt alcatuite dintr-o constanta si un exponent. Fiecare variabila poate avea sau nu un exponent intreg pozitiv, acesta da gradul variabilei din monom .

Operatiile definite in acest program sunt: adunare, scadere, inmultire, derivare si integrare. Pentru a aduna, scadea, sau inmulti avem nevoie de doua polinoame. Noi va trebui sa introducem un polinom in prima casuta , iar in cea de a doua casuta, alt polinom. A treia casuta este pentru afisarea rezultatului.

Tot aici trebuie specificat faptul ca pentru derivarea si integrarea unui polinom avem nevoie doar de prima casuta unde vom introduce polinomul si rezultatul va aparea de asemenea in cea de a treia casuta.

Pentru implementarea „Calculatorului polinomial”, am folosit sase clase pentru implementare si o clasa de test.

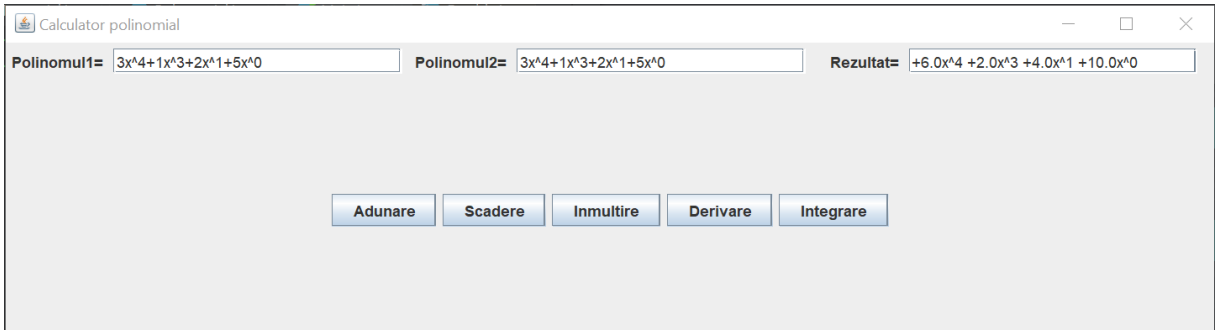
Programul cuprinde clasele:Main, View, Controller, Rejex, Monomial si Polynomial.

In clasa Monomial definim conceptul de monom si functiile de adunare, scadere, inmultire si derivare, urmand ca in clasa Polynomial, sa ne generalizam operatiile pentru un polinom. Clasa Rejex ma ajuta sa separ listele si sa extrag doar informatia pe care vreau sa o folosesc. In clasele Main, View si Controller imi definesc interfata.

De remarcat ca am folosit pachetul java.SWING pentru implementarea interfetei.

4. Proiectare

O sa incep prin descierea interfetei. Este un program foarte usor de utilizat, interfata lui fiind una „User friendly”. Se introduce polinomul de la tastaura, si dupa se alege operatia dorita.

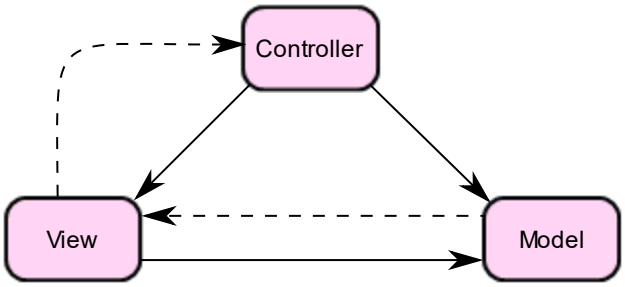


Am folosit MVC care este un model arhitectural utilizat în ingineria software. Succesul modelului se datorează izolării logicii de business față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual sau/și nivelele inferioare ale regulilor de business sunt mai ușor de modificat, fără a afecta alte nivele.

Clasa View este folosita pentru reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controlor.

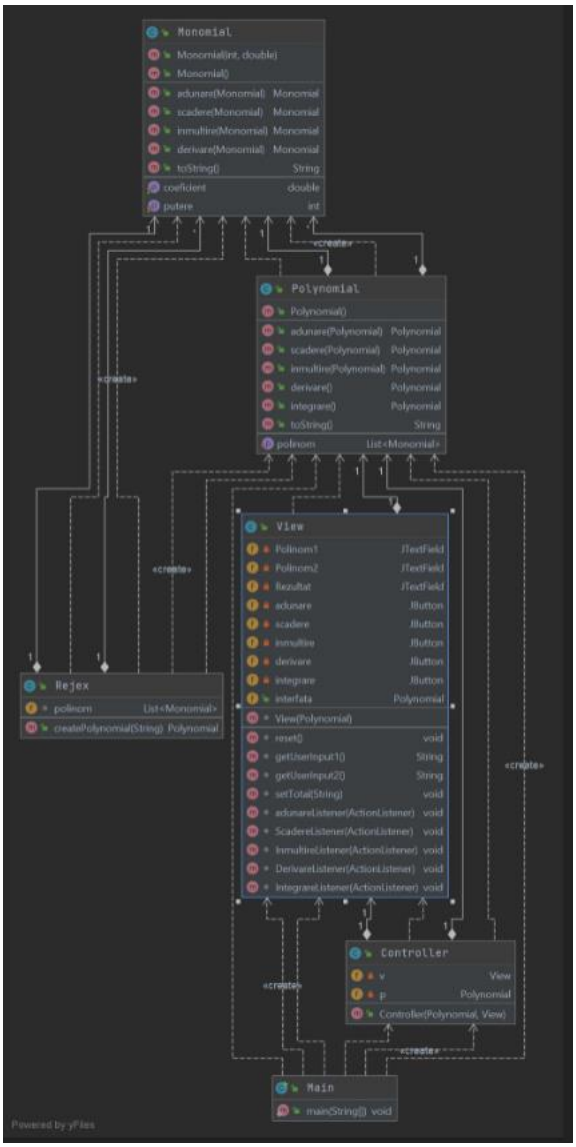
Cu ajutorul Controlorului, modelului sau a parții de vizualizare putem manipula următoarele elemente: datele. Depinde de noi cum manipulăm și interpretăm aceste "date". Acum cunoaștem că unicele date ale unei adrese web statice sunt: obținerea unui fișier de pe disc(hard disk) sau din Internet, etc. și, interpretat (recunoscut/decodificat) sau nu, serverul răspunde.

Modelul, precum controlorul și vizualizarea (interfața grafică) manipulează toate datele ce se relaționeză cu el. Și numai partea de Vizualizare poate demonstra această informație. În acest fel am demonstrat ierarhia programului nostru: Controlor-Model-Vizualizare.



Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

Diagrama UML a proiectului e:



5. Implementare

Clasa Monomial:

In prima parte a programului, am initializat doua variabile cu nume sugestive: putere, aceasta este de tip intreg si reprezinta oridinul la care variabila se ridica, si coeficient, care este de tipul double si reprezinta numarul cu care este inmultita variabila.

```
public Monomial(int putere, double coeficient){
    this.putere=putere;
    this.coeficient=coeficient;
}
```

Am folosit o metoda, unde i-am atribuit unei variabile un coeficient si o putere, urmand dupa niste de functii de get si set atat pentru putere cat si pentru coeficient.

Exemplu pentru putere:

```java
public int getPutere() {
    return putere;
}
```

```java
public void setPutere(int putere) {
    this.putere = putere;
}
```

In aceasta clasa se alfa de asemenea metodele pentru operatiile cerute. Metoda de adunare denumita”adunare” realizeaza suma dintre doua monoame, se aduna coeficientul din fata variabilei cu acceasi putere. Metoda scadere, denumita „scadere” realizeaza diferenta dintre doua monoame, se scade coeficientul din fata variabilei cu acceasi putere. Metoda inmultire denumita „inmultire ” realizeaza inmultire dintre doua monoame, se inmulteste coeficientul din fata variabilei si se aduna puterile lor (se aplica pentru fiecare termen al unui viitor polinom, primul inmultindu-se pe rand cu fiecare termen si asa mai departe). Metoda de derivare denumita „derivare”, realizeaza derivarea unui monom astfel: puterea se inmulteste cu coeficientul variabilei, dupa scade cu o unitate puterea.

Exemplu pentru adunare

```java
public Monomial adunare(Monomial mem){
    Monomial rezultat=new Monomial();
    rezultat.setPutere(this.getPutere());
    rezultat.setCoeficient(this.getCoeficient()+mem.getCoeficient());
    return rezultat;
}
```

De asemenea am folosit @Override pentru a suprascrie metoda toString, care ma ajuta in afisarea mai cosmetica a monomului.

Clasa Polynomial:

Are un singur atribut: o lista de monoame, care va fi data de catre utilizator. Am folosit urmatoarele metode:

metoda de adunare pe care o voi exemplifica mai bine, restul se fac asemanator. Aici se dau trei monoame, dimensiunile a doua polinoame si variabilele de pas. In cazul adunarii polinoamelor se verifica 3 cazuri:daca gradele sunt egale ne vom folosi de momul aux pentru a le aduna, daca gradul primului momom e mai mic decat gradul celui de al doilea si cazul in care gradul primului momom e mai mare decat gradul celui de al doilea. In ambele situatii vom creste contorul polinomului in cauza si il vom adauga pe acessta in rezultat. Se mai verifica doua cazuri: daca polinomul nu a fost parcurs in intregime.

Metoda de scadere ,inmultire, derivare fac parte de asemenea parte din aceasta clasa doar ca de aceasta data operatiile se realizeaza la o scala mai mare, pentru un intreg polinom. In plus fata de clasa precedenta, am mai implementat si metoda de integrare a unui plolinom denumita „integrare” se realizeaza prin inmultirea coeficientului cu variabila la puterea ei +1 impartita la putere+1, luandu-se pe rand fiecare element al polinomului.

```java
//integrare
public Polynomial integrare() {
    Polynomial rezultat = new Polynomial();
    for(Monomial mem : this.polinom) {
        Monomial aux = new Monomial();

        int puterei;
        double coeficienti;

        puterei = mem.getPutere();
        coeficienti = mem.getCoeficient();

        if(puterei >= 0) {
            puterei++;
            coeficienti = coeficienti / (double)puterei;
            aux.setCoeficient(coeficienti);
            aux.setPutere(puterei);
            rezultat.polinom.add(aux);
        }
    }
    return rezultat;
}
```

Tot din motive de cosmetizare, am suprascris metoda toString.

```java
@Override
public String toString() {
    StringBuilder s = new StringBuilder();
    for (Monomial monomial : polinom)
        s.append(monomial.toString());
    return s.toString();
}
```

Clasa Rejex:

O expresie rejex este alcatuita dintr-o succesiune de litere si simboluri care definesc un model logic. String-urile pot fi comparate cu modelul ales de utilizator pentru a identifica daca acestea se potrivesc sau nu cu modelul logic definit de rejex

La inceputul clasei, initializam lista de monoame.

Rejex cauta diferie caractere intr-un string, astfel el reuseste sa „sparga” secventa data de utilizator, dupa felul caracterelor rezultand crearea usoara a unui polinom.

Clasa View:

Clasa View reprezinta implementarea interfetei grafice facute cu ajutorul pachetului java.swing.

Clasa incepe cu 3 casute de text cu nume sugestive pentru cele doua polinoame: Polinom 1 respectiv Polinom 2, aceste casute sunt precedate de casuta pentru rezultatul lor: Rezultat.

Polinoamele vor fi date de la tastatura de catre utilizator, field-urile lor fiind editabile in schimb, field-ul pentru rezultat nu poate fi editat deoarece acolo sa va genera rezultatul operatiei alese.

Urmeaza instantierea butoanelor pentru operatiile pe care le va efectua calculatorul polinomial. Pentru operatia de adunare avem butonul cu nume sugestiv „Adunare”. Pentru operatia de scadere avem butonul cu nume sugestiv „Scadere”. Pentru operatia de inmultire avem butonul cu nume sugestiv „Inmultire”. Pentru dervivare avem butonul cu nume sugestiv „Derivare”. Pentru integrare avem butonul cu nume sugestiv „Integrare”.

```
public class View extends JFrame {

    //casuta text pentru polinoame/rezultat
    private JTextField Polinom1 = new JTextField( columns: 25);
    private JTextField Polinom2 = new JTextField( columns: 25);
    private JTextField Rezultat     = new JTextField( columns: 25);

    //butoane de calcul
    private JButton    adunare = new JButton( text: "Adunare");
    private JButton    scadere    = new JButton( text: "Scadere");
    private JButton    inmultire = new JButton( text: "Inmultire");
    private JButton derivare = new JButton( text: "Derivare");
    private JButton    integrare = new JButton( text: "Integrare");
```

De asememenea avem doua metode cu ajutorul carora preluam datele intoduse de catre utilizator in campurile alocate pentru scrierea polinoamelor si o metoda pentru afisarea rezultatului in campul destinat acestuia.

```
String getUserInput1() { return Polinom1.getText(); }

String getUserInput2() { return Polinom2.getText(); }

void setTotal(String newTotal) { Rezultat.setText(newTotal); }
```

In aceasta clasa se afla si ascultatorii atribuiti fiecarui buton, urmand ca implementarea lor sa fie realizata in clasa Controller.

```
void setTotal(String newTotal) { Rezultat.setText(newTotal); }

void adunareListener(ActionListener adunare) { this.adunare.addActionListener(adunare); }

void ScadereListener(ActionListener scadere) { this.scadere.addActionListener(scadere); }

void InmultireListener(ActionListener inmultire) { this.inmultire.addActionListener(inmultire); }

void DerivareListener(ActionListener derivare) { this.derivare.addActionListener(derivare); }

void IntegrareListener(ActionListener integrare) { this.integrare.addActionListener(integrare); }
```

Clasa Main:

In clasa Main vom apela restul claselor pentru a putea sa vedem interfata.

```
public class Main {                                          ⚠ 1
    public static void main(String[] args) {
        Polynomial Model = new Polynomial();
        View view = new View(Model);
        Controller controller = new Controller(Model, view);
        view.setVisible(true);
    }
    //3x^4+1x^3+2x^1+5x^0
    //7x^4+2x^3+3x^0
```

Cele doua polinoame exemplificate mai sus le-am folosit pentru testarea operatiilor.

Este clasa in care spunem programului ce sa execute. In aceasta clasa se realizeaza deschiderea interfetei grafice.

Clasa Controller:

Cu acest element putem controla accesul la aplicația noastră. Pot fi fișiere, scripturi (eng. scripts) sau programe, in general orice tip de informație permisă de interfață. În acest fel putem diversifica conținutul nostru de o formă dinamică și statică, în același timp.

Clasa Controller este alcatuita dintr-o subclasa pentru fiecare buton. Trebuie sa facem legatura dintre clasa View si clasa Polynomial, asa ca am instantat doua atribute: v si p. Vom folosi ascultatorii creati in clasa View pentru fiecare operatie a calculatorului.

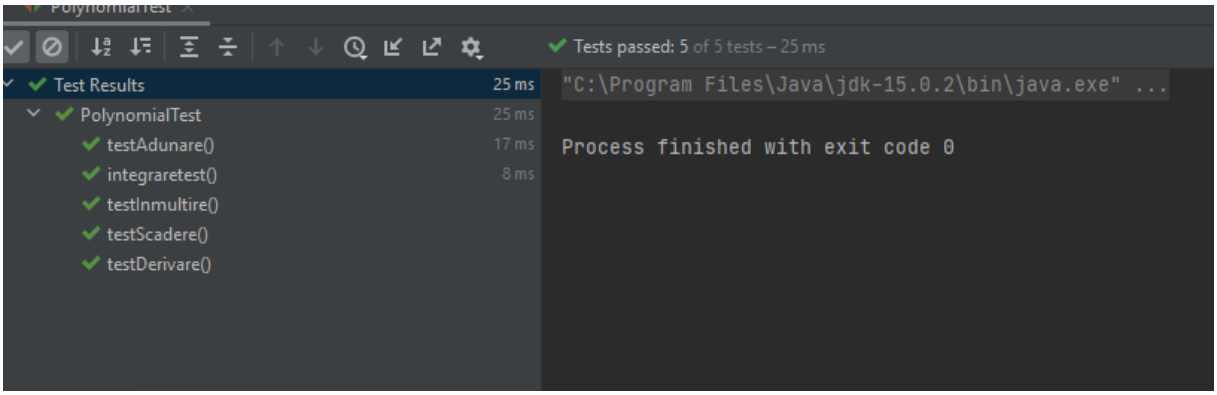
Fiecare buton are implementata o clasa, pentru fiecare ascultator al sau. Fiecare subclasa are instructiunea de reset pentru a nu exista viitoare probleme in cazul in care butonul respectiv sa va apasa de mai multe ori de catre utilizator.

Depinzand de operatia dorita, vom folosi unul sau doua siruri pentru datele de intrare. In cazul operatiei de adunare, scadere si inmultire vom avea nevoie de doua string-uri deoarece aceste operatii necesita doua polinoame, in schimb pentru operatia de derivare si integrare este necesar doar unul. Pentru operatiile de derivare si integrare, trebuie sa introducem polinomul in prima casuta de text, pentru ca doar asa poate sa fie rezolvat si afisat in casuta de rezultat.

Ne folosim de rejex pentru a putea polinomul dorit, si ca in cazul de mai sus, pentru operatia de adunare, scadere si inmultire avem nevoie de doua variabile de tip rejex, una pentru fiecare polinom. In cazul operatiei de derivare si integrare este suficienta doar o variabila de tip rejex pentru ca se va introduce un singur polinom.

Cu ajutorul instructiunii setTotal, polinomul va fi convertit la tipul string, interactionand cu view-ul.

6. Rezultate



Pentru a verifica corectitudiea operatiilor, am implementat o clasa de test numita PolynomialTest.
Pentru fiecare operatie, am creat o metoda de testare unde am oferit rezultatul final calculat de mine si am utilizat operatia dorita. Testarea a fost de tipul unitar implementata cu ajutorul Junit care reprezinta un cadru de testare unitara pentru Java, adica prin metoda Assertions , asertEquals .
Am folosit @BeforeEach pentu ca inaintea fiecarei operatii de testare sa se initializeze polinomul caruia urmeaza sa ii se faca testele.

7. Concluzii

Pentru acest proiect, cea mai importanta concluzie ar fi ca: am realizat inca o data rolul si importanta informaticii in orice domeniu.

De asemenea, sunt de parere ca acest proiect m-a ajutat sa imi aprofundez mai bine cunostiintele in tot ce inseamna limbajul Java, implementarea paradigmelor OOP, creearea unui program cu o interfata grafica, folosirea interfetei model view controller. De asemenea m-au ajutat sa imi reamintesc tehnicile de programare invatate semestrul trecut.

O viitoare impunatatire a acestui proiect ar fi calculul operatiei de impartire, calculul radacinilor polinomului, reducerea acestuia la forma cea mai simpla.

Posibili viitori utilizatori: elevii de liceu, in special cei care sunt in clasa a 12-a, studentii, mai pe scurt, fiecare utilizator care intampina greutati in rezolvarea corecta a operatiilor cu polinoame.

8. Bibliogarfie

Wikipedia: https://ro.wikipedia.org/wiki/Model-view-controller

M-am folosit de wikipedia pentru diferite definitii si informatii pentru interfata model view controller.

Wikipedia: https://ro.wikipedia.org/wiki/Polinom

De aici m-am inspirat pentru definirea polinomului si a monomului.

Pentru certitudinea realizarii operatiilor corecte cu polinoame m-am ajutat de:

https://pdfslide.tips/documents/operatii-cu-polinoame.html

Pe langa acestea, m-am uitat pe diverse site-uri de programare (geeksforgeeks sau W3schools) pentru a incerca sa implementez cat mai bine programul meu.