

**Calculator de polinoame**

**Moldovan Vlad-Madalin**

**CUPRINS:**

**1.Obiectivul temei**

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

**3. Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii , packages, algoritmi, interfata utilizator)**

**3.1 Decizii de proiectare**

**3.2 Diagrama UML**

**3.3 Proiectare Clase**

**3.4 Interfata Utilizator**

**4.Implementare**

**4.1 Clasa Monom**

**4.2 Clasa Polinom**

**4.3 Clasa Operatii**

**4.4 Clasa PolinomView**

**4.5 Clasa PolinomContoller**

**4.6 Clasa PolinomMVC**

**5.Rezultate**

**6.Concluzii**

**7.Bibliografie**

**1.Obiectivul temei**

Obiectivul principal al temei este de a implementa un calculator de polinoame care sa efectueze urmatoarele operatii: adunare, scadere, inmultire , impartire, derivare, integrare. Obiectivul secundar al temei este de a implementa o interfata grafica pentru utilizator respectand formatul Model View Controller , de a lucra cu functii Regex pentru a modela polinoamele, utilizarea de parcurgeri foreach pentru obiecte in loc de for(int i=0…) , folosirea de Junit pentru a testa corectitudinea aplicatiei scrise.

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

Cerinta problemei este de a rezolva operatiile de adunare, scadere, inmultire, impartire, derivare, integrare a polinoamelor. Polinoamele care vor fi introduce sunt de forma: ax^n + bx^(n-1) + … + yx + z , unde a, b, … z reprezinta coeficienti monoamelor si vor fi de tip double , iar n, n-1 … reprezinta gradul monoamelor si vor fi de tip int , un exemplu de polinom care poate fi introdus este: 3x^2 + 2x + 3 sau x^3 - 3x – 2 , iar exemple care nu trebuie introduse sunt: x^2+3x^1+2 sau x^3-2x^0 .

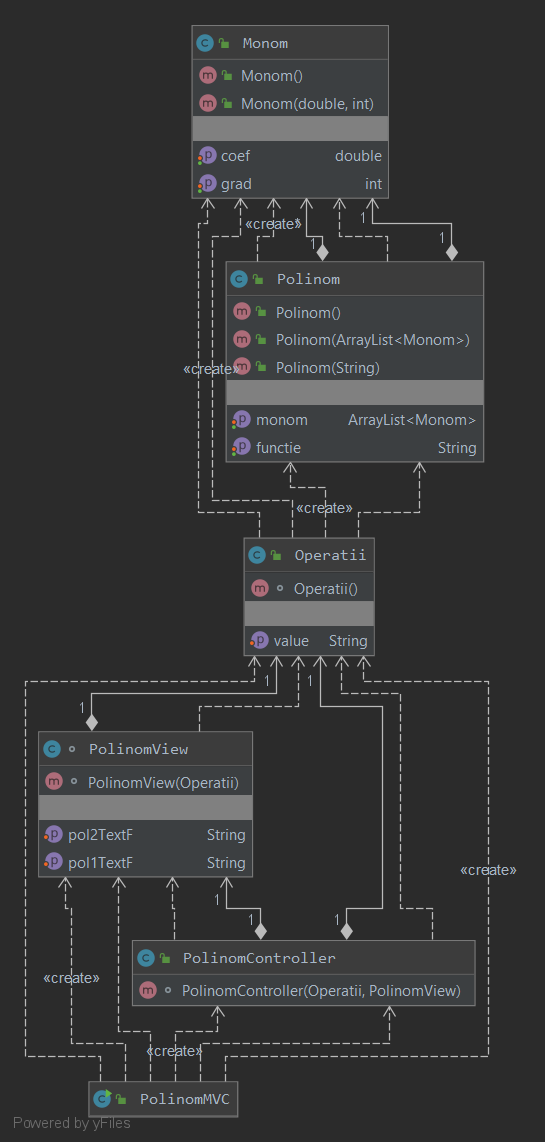
* Pentru adunarea polinoamelor se vor introduce doua polinoame care respecta formatul precizat mai sus in TextFieldurile : “Polinom 1”, respective “Polinom 2” , iar rezultatul va fi afisat in TextFieldul “Rezultat” dupa apasarea butonului “Adunare”.
* Pentru scaderea polinoamelor se vor introduce doua polinoame care respecta formatul precizat mai sus in TextFieldurile : “Polinom 1”, respective “Polinom 2” , iar rezultatul va fi afisat in TextFieldul “Rezultat” dupa apasarea butonului “Scadere”.
* Pentru inmultirea polinoamelor se vor introduce doua polinoame care respecta formatul precizat mai sus in TextFieldurile : “Polinom 1”, respective “Polinom 2” , iar rezultatul va fi afisat in TextFieldul “Rezultat” dupa apasarea butonului “Inmultire”.
* Pentru derivarea unui polinom se va introduce un polinom care respecta formatul precizat mai sus in TextFieldu “Polinom” , iar rezultatul va fi afisat in TextFieldul “Rezultat” dupa apasarea butonului “Derivare”.
* Pentru integrarea unui polinom se va introduce un polinom care respecta formatul precizat mai sus in TextFieldu “Polinom” , iar rezultatul va fi afisat in TextFieldul “Rezultat” dupa apasarea butonului “Integrare”.
* Pentru resetarea campurilor “Polinom 1”, “Polinom 2” si “Rezultat” se va apasa pe butonul “Clear”.

**3.** **Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii , packages, algoritmi, interfata utilizator)**

**3.1 Decizii de proiectare**

A fost implementat la ineceput clasa “Monom” (contine coefficient si grad) , dupa care am introdus clasa “Polinom” care contine o lista de monoame (si expresia functie de tipul string) .Am implementat clasa “Operatii” pentru adaugarea operatiilor de adunare,scadere,inmultire,derivare si integrare .Dupa care am implementat interfata grafica pentru utilizator de tipul “Model View Controller” , iar la final am implementat clasa de test unitar JUnitTest pentru testarea aplicatiei.

**3.2 Diagrama UML**

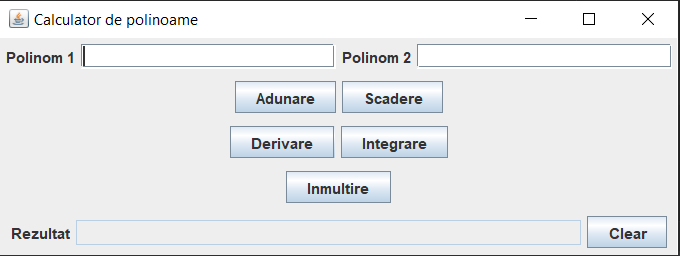
****

**3.3 Proiectare Clase**

* Clasa “Monom” : este utilizata pentru a stoca coeficientul ( de tip double ) si gradul ( de tip int ) pentru fiecare componenta a polinomului . Aceasta clasa implementeza interfata “Comparable” pentru a facilita sortarea polinomului .
* Clasa “Polinom” : este clasa utilizata pentru a stoca polinomul sub forma unei liste de monoame , dar si expresia sa sub forma unui String .
* Clasa “Operatii” : este clasa unde se gasesc metodele pentru operatiile care trebuie sa fie implementate pentru acest calculator de polinoame : adunare , scadere , inmultire , integrare , derivare . Aceasta reprezinta “Model” pentru designul ”Model View Controller” .
* Clasa “PolinomController” : este clasa care reprezinta “Controller” pentru designul “Model View Controller” . Aceasta contine ActionListener pentru fiecare buton al interfetei grafice pentru a controla afisarea rezultatului pentru fiecare operatie .
* Clasa “PolinomView” : este clasa care reprezinta “View” pentru designul “Model View Controller” . Aceasta contine interfata grafica a utilizatorului .
* Clasa “PolinomMVC” : este clasa main de unde se va rula designul “Model View Controller” .
* Clasa “JUnitTest” : este clasa unde se vor realiza teste unitare pentru a verifica corectitudinea aplicatie .

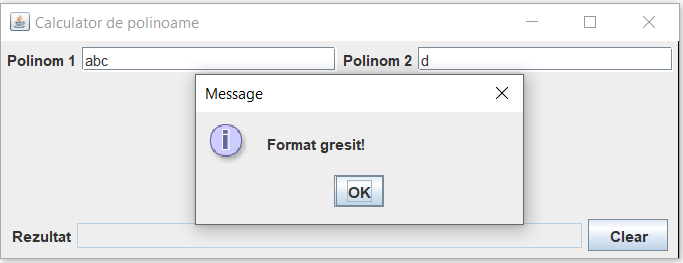
**3.4 Interfata utilizator**

Interfata permite interactionarea utilizatorului cu aplicatia intr-un mod mult mai usor . Aceasta usureaza modul de utilizare al programului .



Interfata acestei aplicatie permite utilizatorului sa efectueze urmatoarele operatii cu polinoame : Adunare , Scadere , Derivare , Integrare si Inmultire . Polinoamele se vor introduce in TextFieldurile “Polinom 1” si “Polinom 2” conform formatului specificat la punctul “2.Analiza problemei , modelare , scenarii , cazuri de utilizare”, iar rezultatul va fi furnizat in TextFieldu “Rezultat” dupa alegerea operatie care se doreste sa fie executata .

In cazul in care utilizatorul va introduce un Polinom cu un format gresit acest lucru va fi sesizat printr-o mesaj de eroare , iar utilizatorul va trebui sa introduca din nou polinomul sau polinoamele pentru a putea utiliza corect acest calculator de polinoame



**4.Implementare**

**4.1 Clasa Monom**

Clasa Monom este utilizata pentru formarea unui Polinom este alcatuit dintr-o lista de monoame . Aceasta clasa contine atributele “coef” care reprezinta coeficientul monomului ( de tip double ) si “grad” care reprezinta gradul polinomului ( de tip int ) .

Metode :

* public Monon()

- reprezinta constructorul default

* public Monom ( double coef , int grad )

- reprezinta constructorul clasei care atribuie coefficient si grad unui obiect nou instantiat

* public double getCoef()

- un getter prin care obinem coeficientul unui obiect de tip Monom

* public void setCoef(double coef)

- un setter prin care putem seta coeficientul unui obiect de tip Monom

* public int getGrad()

- un getter prin care obinem gradul unui obiect de tip Monom

* public void setGrad(int grad)

- un setter prin care putem seta gradul unui obiect de tip Monom

* public int compareTo(Monom c)

- aceasta metoda este scrisa cu @Override pentru a putea sorta o lista de tipul Monom , metoda este pentru ordonare descrescatoare in functie de gradul monoamelor

* public String toString()

- aceasta metoda este scrisa cu @Override pentru a putea afisa un obiect de tip Monom cu coeficientul si gradul sau

**4.2 Clasa Polinom**

Clasa Polinom contine un String “functie” care reprezinta polinomul pe care utilizatorul il va introduce de la tastatura si o lista ( ArrayList ) de monoame unde se va retine gradul si coeficientul fiecarui termen (monom) al polinomului pentru a facilita operatiile care trebuie sa se implementeze .

Metode :

* public Polinom(String functie)

- Un constructor care primeste ca parametru expresia polinomului sub forma de string in variabila “functie” . Pentru a usura delimitarea stringului se va folosi functia regex replace pentru a inlocui caracterel “-“ cu “+-”. Stringul va fi apoi separat intr-o lista de stringuri cu ajutorul functiei regex split , aceasta va fi delimitat de caracterul ‘+’ . Din elementele listei de stringuri se vor obtine gradul si coeficientul fiecarui monom si se va forma lista de monoame .

* public String adaugaFunctie(String s)

- Reprezinta o metoda pentru a usura determinarea coeficientului si gradul fiecarui monom . Aceasta metoda va adauga la fiecare element al listei de stringuri precizata mai sus caractere care ne vor ajuta sa obtinem gradul si coeficientul monomului . Metoda va transforma stringul primit intr-un string de tipul “ax^b” unde a va reprezenta coeficientul , iar b gradul . De exemplu : “x” -> “1x^1” , “2” -> “2x^0” , “x^2” -> “1x^2” .

* public Polinom(ArrayList<Monom> monom)

- Un constructor care primeste ca parametru o lista de monoame . Se va folosi o abordare asemanatoare metodei adaugaFunctie(String s) prezentata mai sus . Lista de monoame primita va fi utilizata pentru a obtine expresia polinomului sub forma de string . Se va parcurge fiecare element al listei de monoame in parte si va forma cate un string (“de tipul “ax^b” , unde a reprezinta coeficientul , iar b gradul cu exceptia cazurilor in care coeficientul este 1 caz in care acesta nu va fi afisat la fel si in cazul in care gradul polinoamelor este 1 sau 0 ) pentru fiecare element si vor fi adaugate la stringul care contine expresia polinomului

* public String getFunctie()

- un getter care va returna expresia polinomului sub forma de string

* public void setFunctie(String functie)

- un setter care va seta expresia polinomului de forma string

* public ArrayList<Monom> getMonom()

- un getter care va returna lista de monoame a polinomului

* public void setMonom(ArrayList<Monom> monom)

- un setter care va seta lista monoamelor

**4.3 Clasa Operatii**

Clasa Operatii implementeaza metodele necesare pentru efectuarea urmatoarelor operatii : adunare , scadere , inmultire , derivare , integrare .Aceasta reprezinta si “Model” pentru designul ModelViewController .

Metode:

* public Polinom adunare(Polinom p1, Polinom p2)

- Reprezinta o metoda pentru a aduna doua polinoame , iar rezultatul va fi un nou polinom .Abordarea folosita este de a cauta fiecare monom din primul polinom in cel de al doilea polinom , iar daca acesta se gaseste se vor aduna coeficienti lor si se va pune monomul in polinomul rezultat , iar daca acesta nu se gaseste se va pune in polinomul rezultat monomul din primul polinom .Dupa aceea se va parcurge cel de al doilea polinom si vom cauta elementele care apar in al doilea polinom si nu apar in primul polinom , iar monomul gasit se va adauga polinomului rezultat.

* public Monom find(Monom a, Polinom b)

- Metoda cauta un monom cu acelasi grad intr-un polinom si va returna un nou monom care va avea coeficientul egal cu suma celor doua monoame de acelasi grad sau va returna null in cazul in care nu se gaseste un monom cu grad egal in polinomul in care se cauta . Aceasta metoda este folosita pentru “adunare” care a fost prezentata mai sus.

* public Polinom subPolimon(Polinom p1, Polinom p2)

- Reprezinta o metoda pentru a scadea doua polinoame , iar rezultatul va fi un nou polinom .Abordarea folosita este de a cauta fiecare monom din primul polinom in cel de al doilea polinom , iar daca acesta se gaseste se vor scadea coeficienti lor( coef monomului din primu polinom – coef monomului din al doilea polinom) si se va pune monomul in polinomul rezultat , iar daca acesta nu se gaseste se va pune in polinomul rezultat monomul din primul polinom .Dupa aceea se va parcurge cel de al doilea polinom si vom cauta elementele care apar in al doilea polinom si nu apar in primul polinom , iar monomul gasit se va adauga polinomului rezultat cu semnul coeficientului schimbat.

* public Monom findSub(Monom a, Polinom b)

- Metoda cauta un monom cu acelasi grad intr-un polinom si va returna un nou monom care va avea coeficientul egal cu diferenta dintre coeficientii celor doua monoame de acelasi grad(coef a – coef monom din polinomul b) sau va returna null in cazul in care nu se gaseste un monom cu grad egal in polinomul in care se cauta . Aceasta metoda este folosita pentru “subPolinom” care a fost prezentata mai sus.

* public Polinom inmultire(Polinom p1,Polinom p2)

- Reprezinta o metoda pentru a inmulti doua polinoame . Se vor parcurge cele doua polinoame si se va inmulti fiecare termen din primul polinom cu fiecare termen din al doilea polinom , iar rezultatul va fi pus intr-o noua lista de monoame . Noua lista de monoame va fi parcursa din nou pentru a forma expresia noului polinom si se va restrange lista de monoame obtinuta anterior (daca lista de monoame contine x^2+x+x+1 aceasta va fi dupa restrangere x^2+2.0x+1). Restrangerea se va face cu ajutorul unei metode “restrange”.

* public ArrayList<Monom> restrange(Monom a,ArrayList<Monom> monom)

- Aceasta metoda primeste ca parametrii un monom si o lista de monoame si va cauta daca exista un monom in lista de monoame care are gradul egal cu cel a monomului transmis ca parametru , in caz ca se gaseste atunci la coeficientul monomului din lista de monoame i se va adauga coeficientul monomului primit ca parametru , in caz contrar monomul primit ca parametru se va adauga in lista de monoame .

* public Polinom derivare(Polinom p)

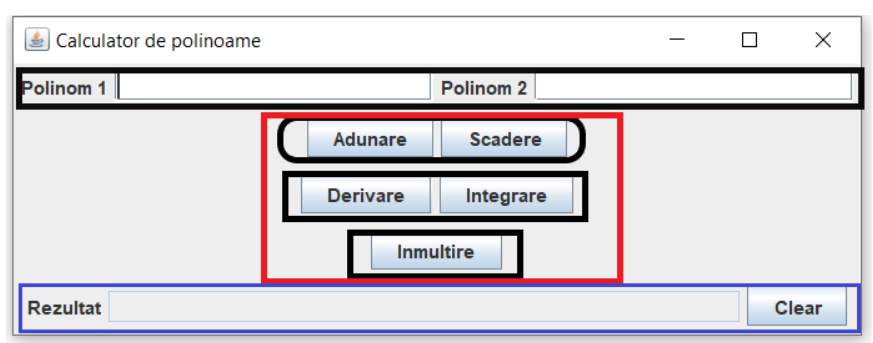
- Reprezinta metoda pentru a deriva un monom . Se va parcurge lista de monoame a polinomului , iar coeficientul fiecarui monom se va inmulti cu gradul actual , iar dupa aceea gradul va scadea cu 1.

* public Polinom integrare(Polinom p)

Reprezinta metoda pentru a integra un monom. Se va parcurge lista de monoame a polinomului , iar gradul fiecarui monom va creste cu 1 ,iar coeficientul monomului se va imparti la noul grad.

**4.4 Clasa PolinomView**

Clasa PolinomView implementeaza interfata grafica pentru utilizator.



Interfata este formata dintr-un JPanel care contine alte trei JPaneluri. Primul JPanel contine doua TextFielduri : “Polinom 1” si “Polinom 2” unde se vor introduce polinoamele pe care se vor efectua operatiile. Al doilea JPanel este alcatuit la randul sau din trei JPaneluri care vor contine butoanele pentru operatiile care pot fi executate : Adunare si Scadere (un JPanel) , Derivare si Integrare (alt JPanel) si Inmultire (alt JPanel) . Al treilea JPanel contine TextFieldu “Rezultat” pe care se va afisa rezultatul operatiei effectuate , contine si un buton de reset numit “Clear” care va reseta cele 3 TextFielduri “Polinom 1” , “Polinom 2” si “Rezultat”.

**4.5 Clasa PolinomController**

Clasa PolinomController contine ActionListenerurile pentru efectuarea operatiilor la apasarea butoanelor:

* class AdunareListener implements ActionListener

- implementeaza adunarea a doua polinoame la apasarea pe butonul “Adunare”

* class ScadereListener implements ActionListener

- implementeaza scaderea a doua polinoame la apasarea pe butonul “Scadere”

* class InmultireListener implements ActionListener

- implementeaza inmultirea a doua polinoame la apasarea pe butonul “Inmultire”

* class DerivareListener implements ActionListener

- implementeaza dereviarea unui polinom la apasarea pe butonul “Derivare”

* class IntegrareListener implements ActionListener

- implementeaza integrarea unui polinom la apasarea pe butonul “Integrare”

* class ResetListener implements ActionListener

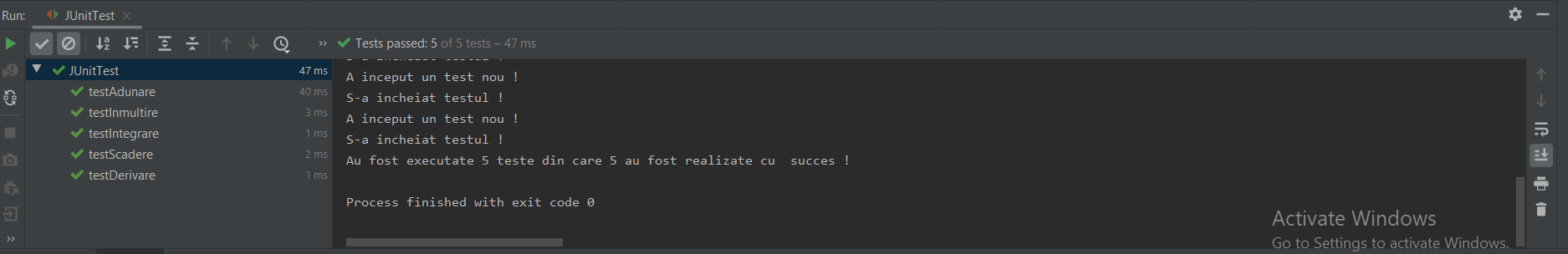
- reseteaza TextFieldurile la apasarea pe butonul “Clear”

**4.6 Clasa PolinomMVC**

Clasa PolinomMVC reprezinta clasa Main a proiectului , iar aici sunt facute conexiunile pentru designul ModelViewController.

**5.Rezultate**

Toate testele au fost efectuate cu success.



**6.Concluzii**

Au fost implementate operatiile de adunare,scadere,inmultire,integrare,derivare pentru polinoame impreuna cu o interfata grafica pentru utilizator pentru a fi mai accesibila utilizarea acestei aplicatii. Din aceasta tema am invatat sa lucrez cu interfata grafica de tipul “ModelViewController” si utilitatea testelor unitare Junit.O dezvoltare ulterioara care se poate aduce este implementarea operatiei de impartire.

**7.Bibliografie**

1.Laboratorul din semestrul I de la Programare Orientata Pe Obiect ( Codul pentru ModelViewController si JUnitTest).

2. <https://mkyong.com/tutorials/junit-tutorials/>

3. <http://ptgmedia.pearsoncmg.com/images/9780321927767/samplepages/0321927761.pdf>

4. <https://www.geeksforgeeks.org/regular-expressions-in-java/>