**DOCUMENTATIE TEMA NR.1**

CALCULATOR DE ………………………….POLINOAME

Student: Bumbu Iulia-Diana

An: II Semestrul: 2

Grupa: 30228

CUPRINS:

1. Obiectivul temei
2. Analiza problemei
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie
8. Obiectivul temei:

1.1.Obiectiv principal:

Principalul scop al proiectului este de a realize o aplicatie care sa implementeze operatii specifice calculului cu polinoame, precum: adunare, scadere, inmultire, impartire, derivare sau integrare. De asemenea, pentru realizarea corecta a operatiilor polinoamele trebuie procesate, fiind alcatuite dintr-o serie de monoame. Fiecare monom are cate un coefficient intreg, precum si un exponent intreg pozitiv care reprezinta rangul/ gradul monomului.

1.2.Obiective secundare:

Pentru o dezvoltare optima a unei aplicatii de acest tip trebuie realizati anumiti pasi pentru indeplinirea obiectivului principal.

Principalele **obiective secundare** sunt:

* Realizarea USE CASE-urilor si a scenariilor:

O diagrama a cazurilor de utilizare ofera o descriere generala a modului in care va fi utilizat sistemul, furnizand o privire de ansamblu asupra functionalitatilor care doresc a fi oferite de sistem. (capitolul 2)

* Identificarea structurilor de date:

Structura de date este o metoda sistematica de stocare a informatiilor intr-un calculator astfel incat le sa poata fi folosite in mod eficient. (capitolul 3)

* Impartirea claselor:

Procesarea polinoamelor precum si a operatiilor se va realiza utilizand clasele Monom si Polinom. Partea de GUI (Graphic User Interface) sa va diviza folosind un MVC (Model- View- Controller). (capitolul 3)

* Dezvoltarea algoritmilor:

Se va gasi solutia necesara realizarii fiecarei operatii folosind metodologia de calcul cunoscuta de la matematica. (capitolul 3)

* Implementarea solutiei:

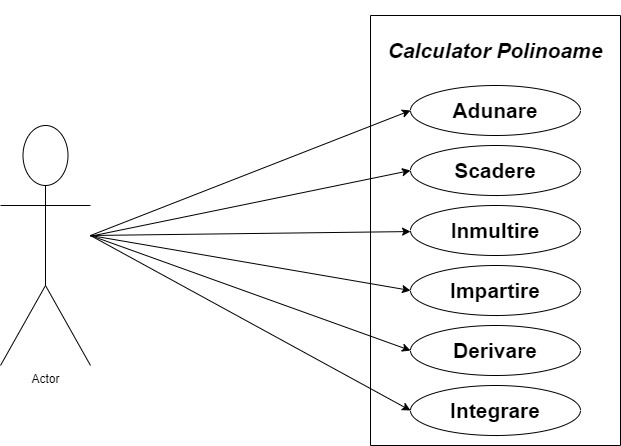
Este reprezentata de descrierea in cod a tuturor claselor anterior explicate, a metodelor necesare efectuarii operatiilor, precum si a interfetei utilizator. (capitolul 4)

* Realizarea testarii:

Se vor scrie scenarii de testare pentru fiecare operatie, validarea realizandu-se folosind JUnit Test. (capitolul 5)

2.Analiza problemei:

2.1.USE CASE Diagram:



2.2. Scenarii:

O utilizare corecta a programului presupune introducerea a doua polinoame (pentru operatiile cu 2 intrari) respectiv a unui polinom (pentru operatiile cu o singura intrare) in text field-uri intr-un format valid. Formatul valid presupune scrierea polinomului sub forma . Termenii nu necesita neaparat sa contina toate gradele sau sa fie in ordine strict descrescatoare, insa este imperativa folosirea literei “x” pentru variabila functiei si specificarea explicita a tuturor coeficientiilor care se considera intregi. In cazul introducerii polinomului in format invalid se va afisa un mesaj de eroare. Daca formatul introdus este valid, dupa apasarea butonului corespunzator operatiei dorite rezultatul va aparea in text field-ul urmator. In cazul operatiilor de derivare si integrare exista posibilitatea alegerii care polinom doreste sa fie calculat prin bifarea butonului P1/ P2.

1. Proiectare:

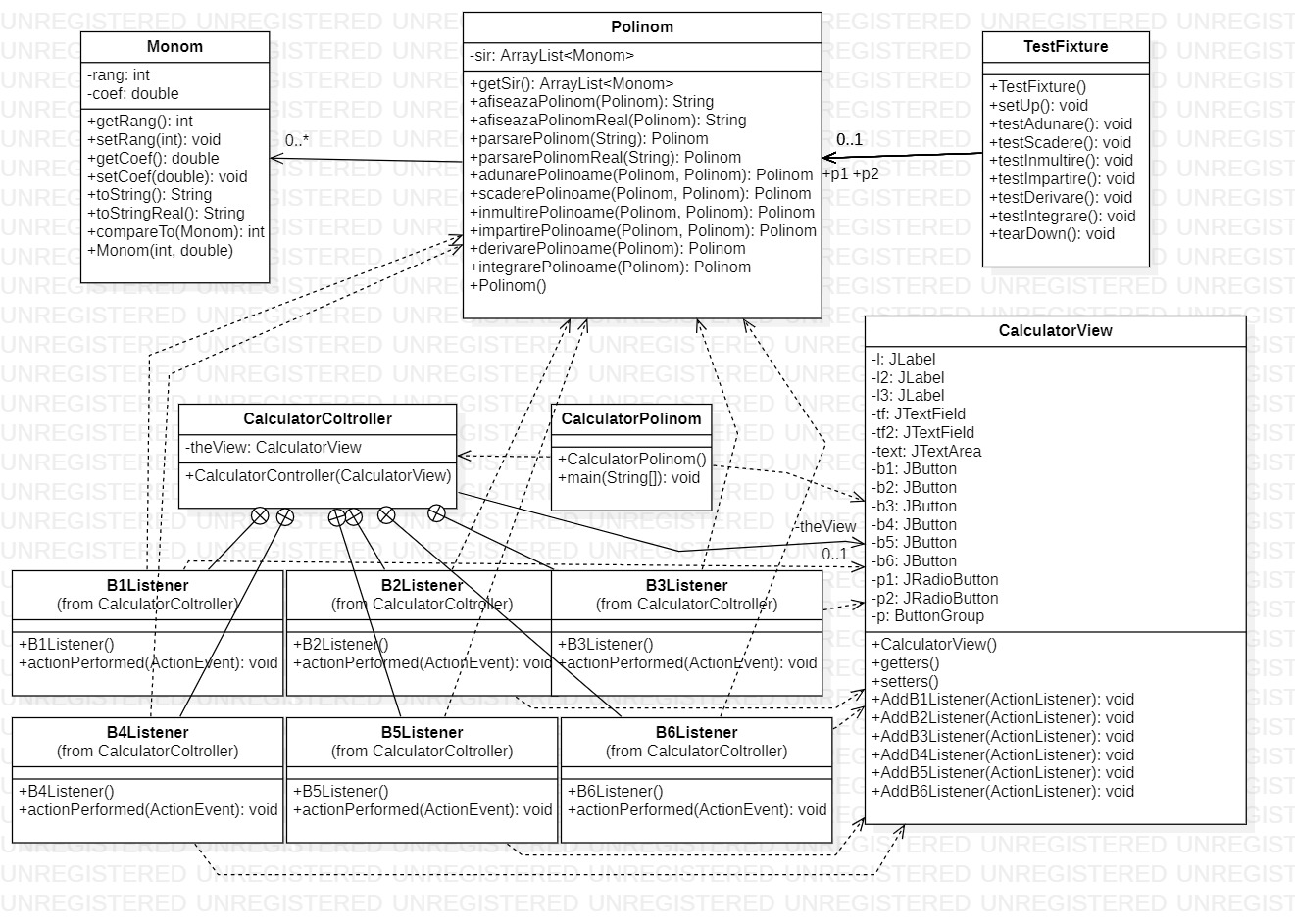
3.1.Structuri de date:

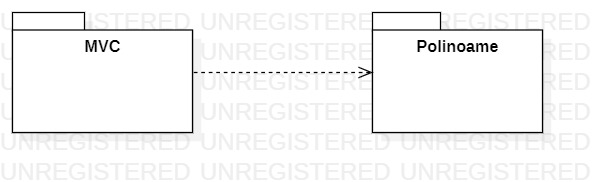
Tinand cont ca datele pe care dorim sa le reprezentam sunt polinoame cu coeficienti intregi si ca un polinom este alcatuit din mai multe monoame, structura de date folosita pentru reprezentarea polinoamelor este un ArrayList de monoame.

ArrayList<Monom> sir= **new** ArrayList<Monom>();

3.2.Diagrama de clase

Diagrama de clase este folosita in modelarea orientata obiect pentru a descrie structura statica a sistemului si a modului in care este el structurat. De asemenea, aceasta ofera o notatie greafica pentru reprezentarea claselor (entitati care au caracteristici comune) si a relatiilor (legaturi dintre doua sau mai multe clase) dintre ele.

3.3.Diagrama de pachete:



UML furnizeaza mijloace de grupare a elementelor din cadrul diagramelor numite pachete. Pachetele constituie baza pentru controlul configuratiei si stocare, contribuind la reutilizarea codului.

3.4.Algoritmi:

Pentru realizarea operatiei de impartire a doua polinoame s-a folosit un algoritm corespunzator modelului matematic modelat in pseudocodul de mai jos:

**function** n / d **is**

require d ≠ 0

q ← 0

r ← n // At each step n = d × q + r

**while** r ≠ 0 **and** degree(r) ≥ degree(d) **do**

t ← lead(r) / lead(d) // Divide the leading terms

q ← q + t

r ← r − t × d

**return** (q, r)

1. Implementare:

4.1.Descrierea claselor din pachetul Polinoame:

* Clasa Monom:

Are ca principala functionalitate ajutarea la formarea polinomului deoarece contine doua atribute

**private** **int** rang;

**private** **double** coef;

folosite la identificarea fiecarui termen component al polinomului. Desi calculatorul este realizat pentru polinoame cu coeficienti intregi, atributul coef este declarat **double** deoarece in cazul operatiilor de integrare sau impartire exista posibilitatea ajungerii in situatii in care tipul coeficientului rezultatului se schimba.

Clasa contine trei metode

**public** String toString() -folosita pentru afisarea monomului in cazul in care se doreste afisare cu coeficient intreg

**public** String toStringReal() -folosita pentru afisarea monomului in cazul in care se doreste afisare cu coeficient real, cu doua zecimale dupa virgula

**public** **int** compareTo(Monom o) -implementata pentru sortarea ulterioara a unui sir de monoame utilizand metoda:

void [java](eclipse-javadoc:%E2%98%82=TPPolinomCalc/C:%5C/Program%20Files%5C/Java%5C/jdk-13.0.1%5C/lib%5C/jrt-fs.jar%60java.base%3Cjava).[util](eclipse-javadoc:%E2%98%82=TPPolinomCalc/C:%5C/Program%20Files%5C/Java%5C/jdk-13.0.1%5C/lib%5C/jrt-fs.jar%60java.base%3Cjava.util).[ArrayList](eclipse-javadoc:%E2%98%82=TPPolinomCalc/C:%5C/Program%20Files%5C/Java%5C/jdk-13.0.1%5C/lib%5C/jrt-fs.jar%60java.base%3Cjava.util(ArrayList.class%E2%98%83ArrayList).sort([Comparator](eclipse-javadoc:%E2%98%82=TPPolinomCalc/C:%5C/Program%20Files%5C/Java%5C/jdk-13.0.1%5C/lib%5C/jrt-fs.jar%60java.base%3Cjava.util(ArrayList.class%E2%98%83ArrayList~sort~Ljava.util.Comparator%5C%3C-TE;%3E;%E2%98%82java.util.Comparator)<? super [Monom](eclipse-javadoc:%E2%98%82=TPPolinomCalc/C:%5C/Program%20Files%5C/Java%5C/jdk-13.0.1%5C/lib%5C/jrt-fs.jar%60java.base%3Cjava.util(ArrayList.class%E2%98%83ArrayList~sort~Ljava.util.Comparator%5C%3C-TE;%3E;%E2%98%82Polinoame.Monom)> c)

* Clasa Polinom:

Are ca unic atribut

**private** ArrayList<Monom> sir= **new** ArrayList<Monom>();

utilizat pentru retinerea tuturor monoamelor din care este constituit polinomul.

Principalele metode utilizate sunt:

**public** String afiseazaPolinom(Polinom p) -folosita la afisarea polinomului in cazul in care rezultatul are coeficienti intregi si indiferent de caz nu exista posibilitatea aparitiei coeficientilor reali

**public** String afiseazaPolinomReal(Polinom p) -folosita la afisarea polinomului in cazul in care rezultatul are coeficienti reali

**public** Polinom parsarePolinom(String s) -deoarece utilizatorul introduce de la tastatura polinomul sub forma de text, acesta trebuie parsat pentru identificarea coeficientilor si rangului/ gradului fiecarui monom din care este alcatuit polinomul

In prima faza se imparte String-ul introdus in portiuni de text corespondente monoamelor de forma cx^r sau c\*x^r utilizand RegEx cu pattern-ul:

Pattern pattern = Pattern.*compile*("([+-]?[^-+]+)");

Pentru fiecare astfel de match gasit se cauta pozitia lui x in sir si relativ la aceasta se salveaza intr-un sir de intregi (deoarece polinoamele citite de la tastatura sunt cu coeficienti intregi) valorile corespondente numarului dinaintea lui x (coeficientul) si dupa x(rangul). Parcurgand acest nou sir format se instantiaza monoamele si fiecare monom se adauga in sirul aferent polinomului.

**public** Polinom adunarePolinoame(Polinom a, Polinom b)

-metoda primeste ca parametri doua polinoame a, b si returneaza ca rezultat polinomul c. Pentru fiecare monom din polinomul a se verifica daca are grad egal cu monoamele din b si in caz afirmativ se instantiaza un nou monom cu grad identic cu al lui a si b si coeficient egal cu suma coeficientilor lui a si lui b, apoi se adauga in sirul de monoame al polinomului c. De asemenea, intr-o lista de monoame

ArrayList<Monom> lista= **new** ArrayList<Monom>();

se adauga monoamele din a si b ale caror coeficienti au contribuit la crearea unui monom din c pentru a tine evidenta a termenilor din a care nu au rang corespondent in b sau invers. Pentru a adauga acest tip de termeni la rezultatul final din lista de monoame din a si din b se sterg monoamele din lista deoarece ele au fost deja luate in calzul, iar cele ramase se adauga la rezultatul final c. Inainte de a returna rezultatul c sortam sirul de monoame pentru a putea fi afisat in ordine normala, de la termenul cu rang cel mai mare pana la cel cu rang cel mai mic

**public** Polinom scaderePolinoame(Polinom a, Polinom b)

-metoda primeste ca parametri doua polinoame a, b si returneaza ca rezultat polinomul c. Pentru fiecare monom din polinomul a se verifica daca are grad egal cu monoamele din b si in caz afirmativ se instantiaza un nou monom cu grad identic cu al lui a si b si coeficient egal cu diferenta coeficientilor lui a si lui b, apoi se adauga in sirul de monoame al polinomului c. Procesarea restul valorilor care nu au corespondent din punct de vedere al egalitatii gradelor se proceseaza analog procedeului descris la operatia de adunare, dupa care se returneaza polinomul rezultat c.

**public** Polinom inmultirePolinoame(Polinom a, Polinom b)

-metoda primeste ca parametri doua polinoame a, b si returneaza ca rezultat polinomul c. Deoarece aceasta metoda va fi folosita si la impartire unde coeficientii pot fi si reali vom declara un sir de numere reale pe care ulterior le vom folosi la pastrarea rezultatului adunarii coeficientilor :

ArrayList<Double> sum= **new** ArrayList<Double>();

Pentru fiecare monom din a, respectiv pentru fiecare monom din b se salveaza in ArrayList-ul declarat sum la indexul x.getRang()+y.getRang() valoarea

sum.get(x.getRang()+y.getRang())+x.getCoef()\*y.getCoef())

La finalul realizarii tuturor iteratiilor din cele doua foruri intercalate in sum o sa avem pe pozitia sum.size()-i-1 a i-a suma a coeficientilor calculata, astfel putand sa se.proceseze polinomul final c.

.**public** Polinom derivarePolinoame(Polinom a)

-metoda primeste ca parametru un polinom a si returneaza un nou polinom c. Conform principiului matematic, pentru fiecare monom din sirul polinomului a se creeaza un nou monom cu rang egal cu randul monomului din a minus 1 si coeficient egal cu coeficientul celui din a inmultit cu rangul monomului din a. Noul monom instantiat este adaugat in lista de monoame a rezultatului c. Cazul in care monomul din a este o constanta este tratat special deoarece o constanta derivata este zero.

**public** Polinom integrarePolinoame(Polinom a)

-metoda primeste ca parametru un polinom a si returneaza un nou polinom c, insa in acest caz tratam polinomul rezultat ca avand coeficienti reali deoarece exista posibilitatea aparitiei a astfel de numere in cazul impartirii numerelor care nu sunt divizibile. Pentru fiecare monom din sirul polinomului a se creeaza un nou monom cu rang egal cu rang+1 si coeficient egal cu coeficientul din a impartit cu rangul+1

**public** Polinom impartirePolinoame(Polinom a, Polinom b) **throws** Exception

-metoda primeste ca parametri doua polinoame a, b si furnizeaza ca rezultat polinomul c. Implementarea urmareste pseudocodul descris in sectiunea 3.4. In algoritmul implementat apar urmatoarele semnificatii: Polinom q-retine catul, Polinom r-retine restul, Polinom t- retine un rezultat partial, Polinom b1-retine impartitorul. Dupa ce copiem in r deimpartitul si in b1 impatitorul verificam daca gradul deimpartitului este mai mic decat cel al impartitorului. In acest caz impartirea nu se poate efectua asa ca aruncam o exceptie. In caz contrar, daca si restul nu este nul se executa bucla while in care calculam cu cat ar trebui inmultit impartitorul pentru ca termenul cel mai semnificativ sa fie egal cu cel termenul 0 al deimpartitului. Acest rezultat il salvam in polinomul t precum si in stringul ts, dupa care il inmultim cu impartitorul din b1 si noul rezultat il scadem din restul r. Dupa se efectueaza scaderea dintre q si polinomul t. Acest set de instructiuni se realizeaza cat timp se respecta conditiile din while si la iesirea din bucla in stringul ts avem toate rezultatele partiale, iar dupa parsarea acestuia in polinomul t1 vom avea catul final. In r a ramas ultimul rest asa ca generam rezultatul final ca fiind cat+rest, rezultat pastrat in c.

4.1.Descrierea claselor din pachetul MVC:

Clasele incluse in acest pachet sunt folosite pentru implementarea interfetei grafice care respecta conventia Model-View-Controller. Clasele pentru vedere si control sunt create explicit, pe cand cea pentru model este considerata ca este constituita din cele din pachetul Polinoame.

* Clasa CalculatorPolinom:

Contine metoda statica main in care sunt instantiate partile de View si Controller

* Clasa CalculatorView:

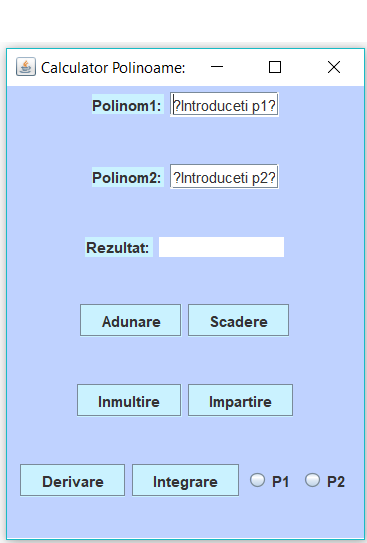
Contine ca atribute elemente din Java Swing folosite pentru a customiza o interfata grafica cat mai user-friendly cu scop de a face acest calculator de polinoame mult mai usor de folosit. Astfel, s-au folosit urmatoarele elemente:

* JFrame: -rama in care adaugam toate elementele si care se inchide la comanda Exit\_on\_close
* JPanel: -s-au instantiat sase panouri care au servit la gruparea mult mai usoara a componentelor, precum si a unui panou principal care le include pe celelalte folosind modul de organizare BoxLayout
* JLabel: -exista trei etichete pentru a indica cele doua polinoame introduse de la tastatura precum si rezultatul
* JTextField: -cele doua spatii in care se introduc polinoamele de la tastatura
* JTextArea: -folosita pentru a scrie rezultatul operatiei in spatiul text aparut
* JButton: -exista sase butoane pentru fiecare operatie specifica polinomului, la apasarea acestora rezultatul fiind scris in spatiul test creat de JTextArea. Procesarea evenimentelor aparute la apasara acestora se face in clasa CalculatorController
* JRadioButton: -utilizat la creea a doua butoane care ne permit sa alegem ce polinom dorim sa folosim la operatiile de derivare/integrare. Deoarece acestea au fost adaugate in acelasi ButtonGroup bifarea unui buton implica debifarea celuilalt
* Clasa CalculatorController:

Contine un atribut **private** CalculatorView theView; care permite adaugarii vederii ascultatorii creati in cele sase subclase. Pentru fiecare buton a fost creata cate o subclasa in CalculatorController care permite procesara evenimentului generat de apasarea butonului prin metoda: **public** **void** actionPerformed(ActionEvent e) Fiecare ascultator preia textul din TextField-uri, il transforma in polinom/polinoame si realizeaza operatia specifica lui: adunare/scadere/inmultire/impartire/derivare/integrare.

Valoarea text a polinomului rezultat calculat este scris in TextArea declarata iar in cazul in care polinomul introdus nu a fost valid si nu se poate realiza operatia, exceptia este interceptata intr-un bloc try-catch, afisandu-se un mesaj de eroare.

Dupa procesarea tuturor claselor constituente pachetului MVC interfata grafica arata in felul urmator:



1. Rezultate:

Pentru realizarea verificarii rezultatelor s-a folosit framework-ul JUnit si s-a creat clasa:

**public** **class** TestFixture **extends** TestCase

in care au fost scrise scenarii de test pentru fiecare operatie. Metoda setUp() initializeaza polinoamele considerate ca date de intrare, iar functia tearDown() indica finalul fiecarui test. Pentru fiecare caz ce test efectuat assertTrue verifica daca rezultatul generat de operatiile cu polinoame scrise e acelasi cu rezultatul asteptat, confirmand corectitudinea metodelor scrise pentru efectuarea operatiilor.

1. Concluzii:

Conclusiv, aceasta tema a fost propice pentru aprofundarea cunostintelor legate de OOP dobandite pana acum, precum si pentru descoperirea a noi functionalitati ale GUI. Proiectul poate suporta dezvoltari ulterioare, precum adaugarea de noi operatii (ex. Valoarea unui polinom, radacinile unui polinom) sau de largire a contextului de validare a polinomului prin introducerea de coeficienti reali sau de utilizarea a altor variabile ( nu doar ‘x’).

7. Bibliografie:

1.wikipedia.com

2.GeeksForGeeks.com

3. StackOverflow.com