**TEHNICI DE PROGRAMARE FUNDAMENTALA**

**TEMA 1**

**CALCULATOR DE POLINOAME**

Nume: Vultur Ana-Maria

Grupa: 30223

**Cuprins:**

1. Obiectivul temei........................................................................................3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare.......................3
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)...................................................................................................4
4. Implementare.............................................................................................5
5. Rezultate....................................................................................................7
6. Concluzii....................................................................................................7
7. Bibliografie................................................................................................8
8. **Obiectivul temei**

**Obiectivul principal**

Obiectivul principal al temei este de a creea un calculator care sa implementeze un sistem de procesare a polinoamelor, acestea fiind construite din termeni numiti monoame, care sunt alcatuite dintr-o constanta (numita coeficient, inmultita cu o variabila, fiecare variabila poate avea un exponent constant intreg pozitiv.

**Obiective secundare**

|  |  |
| --- | --- |
| **Obiectiv Secundar** | **Descriere** |
| Dezvoltarea de use case-uri si scenarii | Într-un sistem software un ‘use-case’ este o listă de acțiuni sau pași de eveniment care definesc în mod obișnuit interacțiunile dintre un rol ( cunoscut ca un actor în limbajul unificat de modelare ( UML ) ) și un sistem in atingerea unui obiectiv.  In computere, un scenariu este o naratiune a interactiunilor dintre roluri ( actori in UML ) si sistem. In cazul nostru rolul - un utilizator al aplicatiei. |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capat obiectivul principal |
| Impartirea pe clase | Folosirea unui MVC ( Model – View – Controller) pentru a putea forma un GUI ( Graphic User Interface ), a clasei Polinom, Monom etc. |
| Dezvoltarea algoritmilor | Vor fi descrise structurile de date necesare pentru atingerea obiectivului principal, schema UML ( diagrama de clase ) precum si algoritmii folositi pentru realizarea operatiilor. |
| Implementarea solutiei | Vor fi descrise fiecare clasa cu campurile si metodele importante precum si descrierea interfetei utilizator; |
| Testare | Vor fi descrise cateva scenarii de testare a operatiilor pe polinoame, folosind ca instrument un Junit Test. |

1. **Analiza problemei**

Prin analiza problemei, ne referim la un prim set abstract de operații și proprietăți prin care încercăm să depistăm eventualele însușiri și comportamente ale proceselor necunoscute.

În cazul Calculatorului de Polinoame, se cunoaște că aplicația trebuie să implementeze operațiile algebrice elementare – or asta implică că trebuie să existe minim două polinoame și/sau două câmpuri de intrare. Modul de codificare al informației, dar și formatul acesteia ocupă un rol foarte important. In primul rand, acesta ar trebui sa permita utilizatorilor sa insereze polinoame, apoi acestia trebuie sa poata selecta operatia matematica, iar apoi sa vada rezultatul operatiei.

Alte cerinte non-functionale ale calculatorului de polinoame sunt faptul ca acesta ar trebui sa fie intuitiv si usor de utilizat de catre orice persoana. Programul va putea fi accesat de un număr ridicat de persoane, de aceea interfața cu utilizatorul devine punctul de pornire al proiectului. Ea trebuie să permită, într-o manieră cât mai convenabilă, comunicarea utilizatorului cu aplicația.

**Use Case-uri**

Utilizarea programului presupune introducerea de catre utilizator (in TextField-uri ) a celor doua polinoame ce urmeaza a fi folosite pentru efectuarea anumitor operatii ( adunare, scadere, inmultire, impartire, derivare, integrare ). Prin apasarea unor Butoane ce au semne specifice pentru operatii, intr-un alt TextField o sa apara rezultatul asteptat.

Deoarece e posibil ca unii utilizatori sa introduca polinoame gresite, de fiecare data cand un polinom introdus in TextField are un format gresit fata de cel normal al unui polinom, sau sunt introduse diferite

Stringuri care nu formeza un polinom ( ex. Ana are mere ), la apasarea unui buton ce semnifica operatia va aparea o fereastra ce va avertiza utilizatorul ca polinomul introdus de el nu este valid ( nu respecta formatul);

Use Case: inmultirea/adunarea/impartirea/scaderea polinoamelor

Actor principal: utilizatorul

Principalele scenarii de succes:

1. Utilizatorul introduce corect in interfata grafica 2 polinoame
2. Utilizatorul selecteaza operatia de inmultire
3. Rezulatatul va aparea in campul „Rezultat ” din interfata grafica

Scenarii de esec: - utilizatorul introduce gresit polinoamele => se va afisa un mesaj de eroare, iar utilizatorul se intoarce la primul pas de mai sus

- utilizatorul nu poate lansa aplicatia

Cerințele funcționale, după introducerea polinoamelor de intrare de către utilizator, sunt:

- realizarea adunării celor două polinoame introduse, dacă se apasă butonul + ;

- realizarea scăderii celor două polinoame introduse , dacă se apasă butonul - ;

- realizarea înmulțirii celor două polinoame introduse, dacă se apasă butonul \* ;

- realizarea împărțirii celor două polinoame introduse, dacă se apasă butonul / ;

- realizarea derivării polinomului P introdus de utilizator, dacă se apasă butonul P’ ;

- realizarea integrării polinomului P introdus de utilizator, dacă se apasă butonul ∫Pdx ;

**Scenarii**

Utilizarea normală a programului presupune introducerea primului polinom în formatul clasic LateX, cu coeficientii in ordine descrescatoare. Se repetă procesul și pentru polinomul numărul doi, iar apoi user-ul poate să efectueze operațiile elementare existente sub TextField-uri.

Totuși, cazul de mai sus este puțin probabil și adesea lucruri neașteptate pot să apară. Dintre acestea se remarcă :

• Introducerea de text, fără semnificație numerică, în câmpurile test

• Introducerea de necunoscute suplimentare(pe langă x)

• Folosirea altor cuvinte cheie decât cele specificate

• Scriere corectă a polinoamelor, dar moduri sintactice care diferă puțin de la user la user( de ex. 2x + 2 vs. 2\*x^3+5\*x)

1. **Proiectare**

Ca si structura de data principala ce am folosit-o este un ArrayList de Monoame (deoarece un polinom este format din mai multe monoame, un arraylist ar fi fost cel mai potrivit in rezolvarea acestei probleme ).

List<Monom> polinom = new ArrayList<Monom>();

Un monom este format dintr-un coeficient, un grad si o variabila ( ex. 2X^3 , unde 2 – coeficientul, x – variabila, iar 3 – gradul ( puterea ) ). Monom-ul va fi in java o clasa separata care va contine ca variabile instanta un intreg, reprezentat de putere ( grad ) si un obiect care reprezinta coeficientul ( acesta va fi de tip int sau double deoarece polinoamele de intrare au coeficienti intregi, dar in cazul impartirii sau integrarii se pot obtine polinoame cu coeficienti reali).

Calculatorul de polinoame utilizeaza pattern-ul arhitectural de MVC(Model-View-Controller), asa ca am folosit 3 pachete: Model, View si Controller. Pachetul Model contine clasele: Polynomial, Monomial, MonomialInt, MonomialDouble, si Model(unde am implementat metodele pentru operatiile de polinoame). Pachetul View contine o clasa View, care se ocupa de interfata grafica cu utilizatorul, realizata cu JavaSwing, iar pachetul Controller, contine o clasa principala Controller si clase interne care implenteaza actiunile de Listener pentru butonele din interfata. Asadar, acest pachet face legatura dintre Model si View, da comenzile pentru butoane, controleaza aplicatia.

Clasa Monomial este o clasa generica, deoarece coeficientul poate fi intreg sau real. Asadar, am creat clasele MonomialInt(are coeficientul de tip Integer) si MonomialDouble(are coeficientul de tip Double) care mostenesc clasa Monomial.

Atributele claselor sunt private, iar pentru a le accesa am folosit metode de getters si setters.

**Algoritmi** folositi:

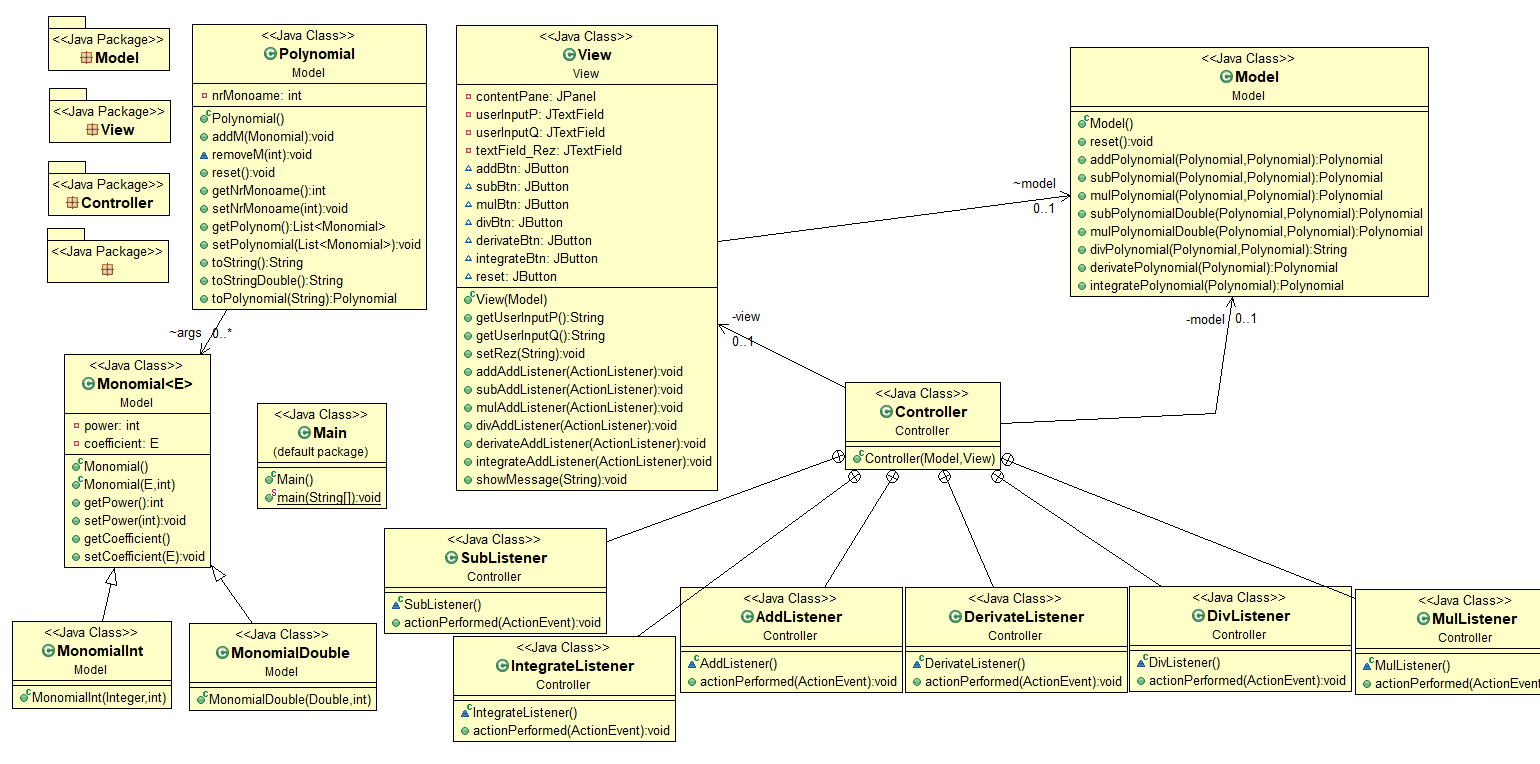
- Algoritmul de impartire a 2 polinoame: - monomul cu grad mai mare din deimpartit se imparte la cel cu grad mai mare din impartitor

- se adauga la cat

- monomul obtinut se inmulteste cu impartitoul, dupa care se scade din deimpartit

- noul deimpartit va fi rezultatul obtinut la pasul anterior

- se repeta pasii de mai sus pana cand gradul deimpartitului este mai mic decat gradul impartitorului, moment in care algoritmul se opreste si ce ramane in deimpartit va fi restul impartirii

**Diagram de clase:**

1. **Implementare**

Clasa **Monomial**:

- este o clasa generica

- variabila instanta: power de tip private int

- instanta a clasei: coefficient de tip private E

- pentru a accesa variabilele instanta si instantele am creat getters si setters

- am doi contructori, unul fara parametri si unul cu parametrii

Clasa **MonomialDouble**:

- este o subclasa a clasei Monomial, in care parametrul generic este Double

- are un constructor care are rolul de a initializa coeficientul si puterea monomului

Clasa **MonomialInt**:

- este o subclasa a clasei Monomial, in care parametrul generic este Integer

- are un constructor care are rolul de a initializa coeficientul si puterea monomului

Clasa **Polynomial**:

- ca instanta am un ArrayList de monoame de tip private

- variabila instanta: nrMonoame private int

- getters si setters pentru acestea

- metode de adaugare si stergere din lista de monoame

- metoda de transformare a unui polinom intr-un string(in 2 variante: cand are coeficienti intregi sau reali)

Clasa **View**:

- in constructor se creeaza frame-ul pentru GUI si se adauga componentele in frame.

- am 2 metode pentru a prelua textul introdus de utilizator in primul textField, respectiv in al doilea textField

- o metoda de a seta rezultatul in textField-ul de rezultat dupa apasarea pe butonul corespunzator operatiei careia dorim sa ii aflam rezultatul

- urmatoarele 6 metode sunt pentru a seta actionListener pentru fiecare buton din frame.

- metoda de afisare a unui mesaj de eroare in cazul in care am introdus un polinom in format gresit

Clasa **Controller**

**-** are 2 instante ale clasei: Model si View

- in constructor se initializeaza cele 6 instante si se adauga cele 7 actionListener care sunt definite ca si clase interne in aceasta clasa

- toate clasele interne au rolul de a realiza operatia corespunzatoare butonului apasat de utilizator

- operatia de derivare si integrare se face doar pentru primul polinom introdus

Clasa **Model**:

- nu are variabile instanta sau instante ale clasei

- contine metodele care realizeaza operatiile dintre monoame

- addPolynomial: se considera ca polinoamele sunt ordonate in ordinea descrescatoare a puterii, iar cat timp niciunul din polinoame nu s-a terminat, daca puterea este aceasi adun coeficientii si adaug in rezultat monomul obtinut, daca nu, adaug monomul care are puterea mai mare si trec la urmatorul monom din lista acelui polinom. Cand unul din polinoame a fost parcurs complet, se adauga la rezultat monoamele din celalalt polinom

- subPolynomial: la fel ca si la addPolynomial, doar ca se scad coeficientii monoamelor, nu se aduna

- mulPolynomial: se parcurge lista monoamelor celor 2 polinoame, utilizandu-se 2 for each-uri si se inmulteste fieacare monom cu fiecare: coeficientii se inmultesc, iar puteriile se aduna

- divPolynomial: se face algoritmul de impartirea a doua polinoame descris la punctul 3

- derivatePolynomial: returneaza un Polynomial de MonomialInt, cu coeficientii fiind rezultatul inmultirii dintre coeficientul monomului si puterea acestuia decrementata cu 1

- integratePolynomial: returneaza un Polynomial de MonomialDouble, deoarece la integrare se imparte coeficientul la puterea incrementata cu 1, fiind o impartire, rezultatul poate fi real.

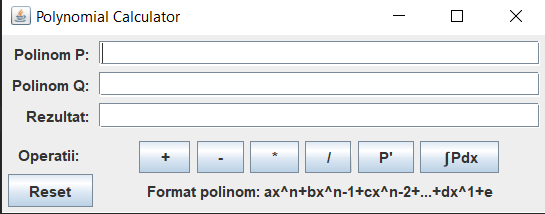
Clasa **Main:**

- contine metoda main de unde se poate porni executia programului

- se initializeaza cele 3 componente: Model, View, Controller

- se face fereastra de View vizibila si va aparea pe ecran calculatorul de polinoame

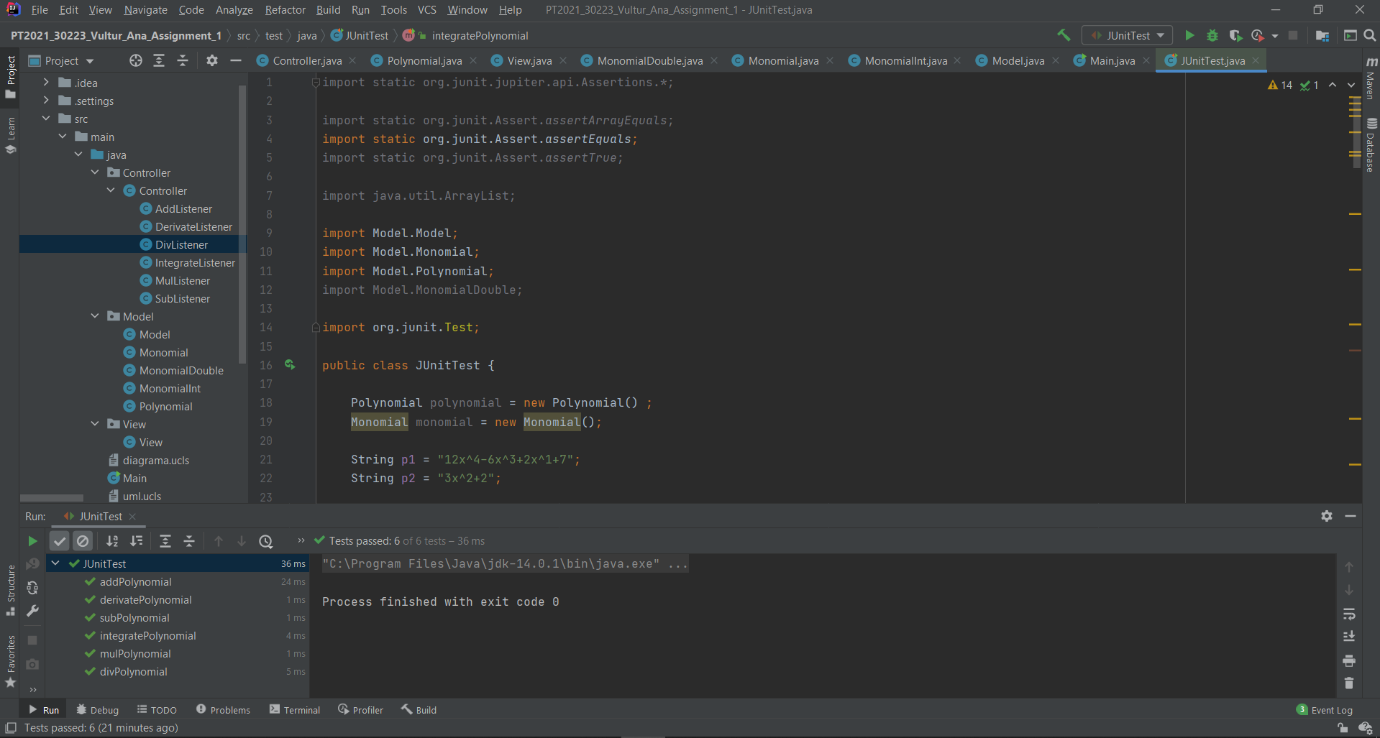
Asa va arata aplicatia la pornire:



1. **Rezultate**

Aplicația întoarce rezultatul așteptat în orice scenariu. Pentru a asigura acest lucru, la transformarea datelor introduse de utilizator, s-au folosit regular expressions pentru identificarea termenilor polinoamelor. Dacă un text field este lăsat gol și utilizatorul încearcă sa efectueze o operație, îi va fi afișat un mesaj, iar operația respectivă, care ar putea genera probleme, nu este efectuată. Totuși, chiar dacă s-ar efectua operația pe polinoame goale, aplicația ar returna rezultatul 0, sau, în cazul integrării, ar returna constanta care se adaugă la integrare.

Am testat operatiile folosind Junit. Am facut o noua clasa de test in proiectul meu unde am facut metode de test pentru fiecare operatie.



1. **Concluzii**

În concluzie, lucrul la această tema m-a ajutat să-mi reîmprospătez cunoștințele despre polinoame și operații cu acestea și, de asemenea, cum să le transpun în cod. Am învățat să lucrez cu modelul MVC, care mi se pare foarte util pentru proiectele viitoare. A fost primul proiect în care a trebuit să folosesc expresii regulate(Regex) pentru a verifica o intrare și a fost o provocare să încerc sa imi dau seama cum functioneaza.

Ceea ce privește evoluțiile ulterioare, am câteva idei pentru a ajusta interfața pentru a o face mai atractivă, deoarece în zilele noastre asta caută un utilizator, acesta nu are preocupări cu privire la modul în care funcționează aplicația sau de ce necesită atât de multă memorie . În ceea ce privește posibilele actualizări ale aplicației, am în vedere să implementez un algoritm care găsește rădăcinile polinomului și returnează o schemă grafică și, de asemenea, să mărească performanța, astfel încât aplicația să funcționeze mai repede.

1. **Bibliografie**

<https://www.tutorialspoint.com/java/java_regular_expressions.htm>

<https://www.geeksforgeeks.org/>

<https://www.jrebel.com/blog/java-regular-expressions-cheat-sheet> <https://www.edureka.co/blog/mvc-architecture-in-java/>

<https://docs.oracle.com>