

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**~SPECIALIZAREA CALCULATOARE SI TEHNOLOGIA INFORMATIEI~**

**Polynomial Calculator**

**Grupa 30226**

**An 2, Semestrul 2**

**Arseniuc Anamaria**

**-Documentatie-**

**Cuprins**

1. Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . pg 3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . pg 3
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . pg 3-4
4. Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .pg 4-6

4.1. Clase si metode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . .pg 4-5

4.2. Interfata grafica . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . pg 5-6

1. Rezultate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .pg 6
2. Concluzii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .pg 6
3. Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . pg 6
4. Obiectivul temei

Obiectivul acestei temei de laborator este de a proiecta si a implementa un calculator de polinoame cu interfata grafica, in care utilizatorul poate introduce 2 polinoame (cu o singura variabila si coeficienti intregi) si a face anumite operatii asupra acestora (adunare, scadere, inmultire, impartire, derivare si integrare) si , desigur, de a se afisa rezultatul.

1. Analiza problemei

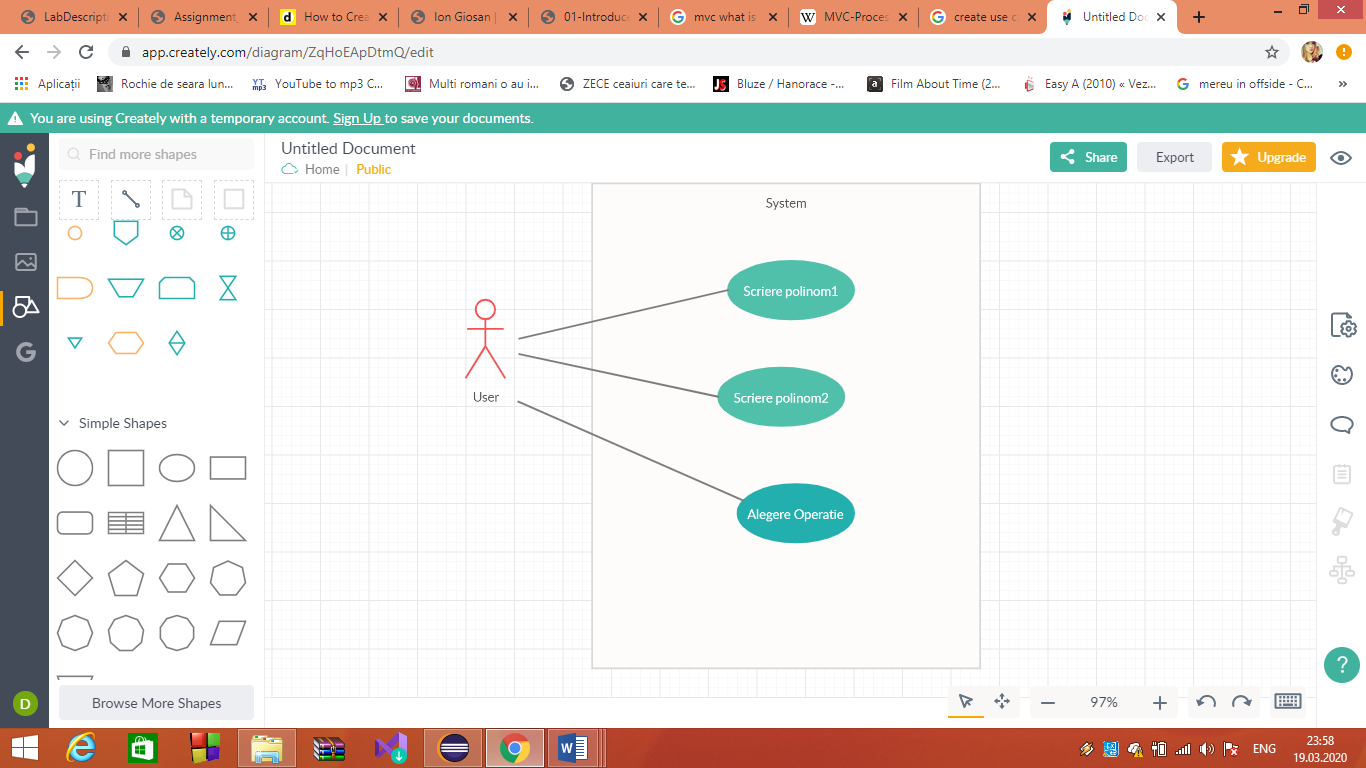
Un polinom este o expresie, formata din mai multe monoame, care, la randul lor, sunt construite din coeficienti si exponenti. Exista o multitudine de metode prin care se poate realiza un calculator de polinoame. E la atitudinea programatorului modul prin care vrea sa se introduca informatiile (doar coeficientii; polinomul intreg, scris ca un sir (ex. “3x^3-2x^2+5x^1-3)), modul in care sunt ulterior asezati, sortati , aspectul de final, si aspectul interfetei grafice.

Figura1. Use case pentru utilizator

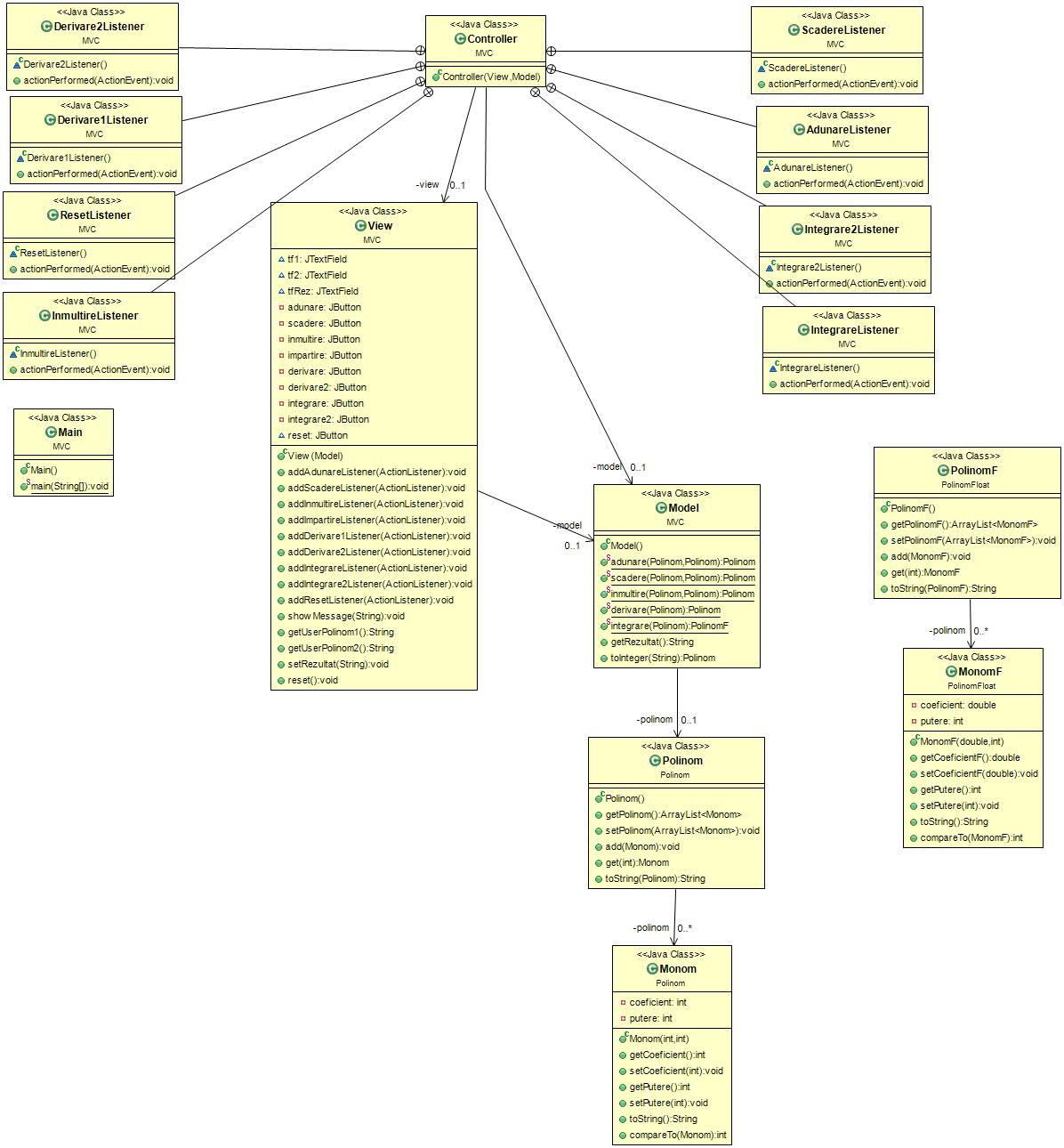
Utilizatorul poate face 3 actiuni: sa scrie cele 2 polinoame (in casete diferite) si sa aleaga operatia pe care o doreste.

1. Proiectare

Proiectul am ales sa il implementez in 3 pachete:

1. Polinom : cuprinde clasele Monom si Polinom.
2. PolinomF: cuprinde aceleasi clase, diferenta fiind in reprezentarea reala a coeficientilor monomului fata de cea intreaga de mai sus; aceste clase au fost necesare pentru a putea fi valida operatia de integrare (de ex., daca aveam ca si coeficient “2/3”, rezultatul ar fi fost “0” cu clasa Monom care detine coeficienti intregi).
3. MVC: cuprinde clasele sale specifice (Model-View-Controller), dar si Clasa principala, Main.

Diagrama UML a proiectului este cea din *figura2*. UML (Unified Modeling Language) este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost 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.

 Figura2. Diagrama UML a proiectului

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . .. . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . .. . . . . . . .. . . . . .. . . . . . . . . . . . . . . . . . .. . . . . . . . .. . . . . .. . . . . . .. . . . . . . .. . .. . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . .

1. Implementare
   1. Clase si metode

***Clasa Monom***

Clasa cea mai low-level, de unde se incepe implementarea efectiva a programului; e cea in care tinem monoamele polinomului (coeficientii si puterile). Metodele folosite sunt cele cunoscute :settere si gettere asupra atributelor clasei; metoda *compareTo()* implementata de interfata *Comparable* pentru aputea sorta monoamele unui polinom dupa puteri si metoda *toString()* pentru a afisa “frumos” monoamele.

***Clasa Polinom***

Cuprinde o lista (si nu tablou) de monoame. Ca si metode principale, pe langa settere si gettere, ar fi *add(Monom m)* –adauga un monom la polinom, *get(int i)*-returneaza monomul de la pozitia “i” si cea din urma, metoda *toString()* care ajuta la formarea polinomului ca si un sir (“ax^3+bx^2+cx^1+d”).

***Clasa MonomF si PolinomF***

La fel ca si cele de mai sus, diferenta e ca coeficientii sunt de tip real (double).

***Clasa Main***

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

***Clasa Controller***

Aceasta clasa reprezinta granita dintre interfata grafica si programul propriu-zis, unde se petrece interactiunea utilizator-sistem de calcul. Ea imi preia datele din View, le prelucreaza in clasa Model si le transmite inapoi la View.

***Clasa View***

Aceasta ne ajuta in crearea efectiva a interfetei (adaugarea butoanelor, a TextField-urilor, Panel-urilor etc). De asemenea, aici imi creez listenere pentru butoane, metode pentru preluarea datelor scrise de utilizator in interfata si setarea rezultatului, cat si o metoda de *reset* care imi elibereazaspatiile din aceasta.

***Clasa Model***

In aceasta clasa am implementate toate operatiile cerute (adunarea, scaderea, inmultirea etc) pentru a fi prelucrate mai apoi in clasa Model.

* 1. Interfata grafica

MVC este un model de proiectare software utilizat în mod obișnuit pentru dezvoltarea interfețelor utilizatorului, care împarte logica programului aferent în trei elemente interconectate: model, view, controller (de unde si numele). Acest lucru se face pentru a separa reprezentările interne ale informațiilor de modul în care informațiile sunt prezentate și acceptate de utilizator.

In figura 4 (cea de mai jos), avem un exemplu de calculare a sumei a doua polinoame si afisarea rezultatului in casuta corespunzatoare acestuia.

Se observa, totodata, butoanele pentru diferitele operatii ce se pot face asupra polinoamelor (in numar de 9).

Atentie!! Termenul liber al polinomul se poate introduce atat liber (2x^2 +3), cat si scriindu-l cu x^0 (5x^3-3).

Figura3. Diagrama interacțiunilor în cadrul modelului MVC.

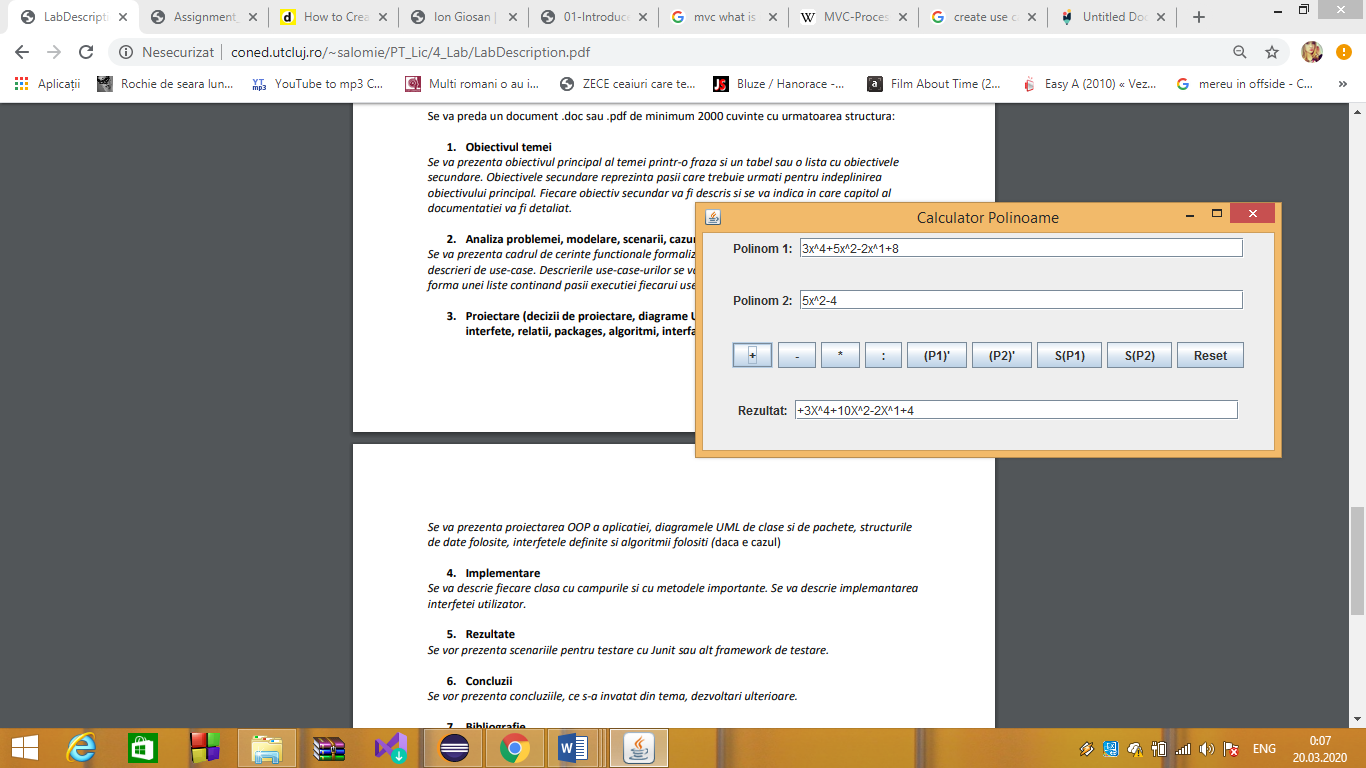


Figura4

Butoane:

1. “ + ” – la apasarea acestuia, se afiseaza in locul aferent rezultatul *adunarii* celor 2 polinoame
2. “ - ” - la apasarea acestuia, se afiseaza in locul aferent rezultatul *scaderii* celor 2 polinoame
3. “ \* ” - la apasarea acestuia, se afiseaza in locul aferent rezultatul *inmultirii* celor 2 polinoame
4. “ : ” - la apasarea acestuia, se afiseaza in locul aferent rezultatul *impartirii* celor 2 polinoame
5. “ (P1)’ ” - la apasarea acestuia, se afiseaza in locul aferent rezultatul *derivarii primului polinom*
6. “ (P2)’ ” - la apasarea acestuia, se afiseaza in locul aferent rezultatul *derivarii celui de-al doilea polinom*
7. “ S(P1) ”- la apasarea acestuia, se afiseaza in locul aferent rezultatul *integrarii primului polinom*
8. “ S(P2) ” - la apasarea acestuia, se afiseaza in locul aferent rezultatul *integrarii polinomului al doilea*
9. “ Reset ”- la apasarea acestuia, *se reseteaza* (curate) interfata, stergandu-se datele din momentul respectiv
10. **Rezultate**

In mare, rezultatele sunt cele asteptate. Utilizatorul scrie polinoamele din punct de vedere matematic (ax^p1+bx^p2+cx^p3 +…), programul reuseste sa imi extraga monoamele din string-ul primit de la interfata si mi da inapoi spre interfata cu acelasi format.

1. **Concluzii**

Reprezentarea polinoamelor cu ajutorul monoamelor este una utila, permitand acces rapid la fiecare termen (cu puterea respectiva) alaturi de coeficientul sau. Operatiile si cautarea sunt astfel mult mai eficiente, permitandu-se lucrul pe o structura cu dimensiune modificabila.

In concluzie, consider ca acest proiect mi-a aprofundat cunostiintele in tot ceea ce inseamna limbajul Java, implementarea paradigmelor Programarii Orientate pe Obiect, creearea unui program cu o interfata grafica. Sunt de parere, totodata, ca sunt chestii care pot fi imbunatatite din privinta introducerii polinoamelor pentru o utilitatea mai buna, un aspect mai placut si depunerea unui efort mai mic din partea utilizatorului pentru a le scrie.

**7. Bibliografie**

* 1. StackOverflow
  2. Wikipedia
  3. Youtube
  4. <https://app.creately.com/diagram/ZqHoEApDtmQ/edit>