**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

FUNDAMENTAL PROGRAMMING TECHNIQUES

ASSIGNMENT 1

**POLYNOMIAL CALCULATOR**

**Viman Andrei-Liviu**

**Grupa: 30226**

**Cuprins**

1.Obiectivul

2.Analiza problemei si scenarii de utilizare

3.Proiectare

4.Implementare

5.Rezultat

6.Concluzii

7.Bibliografie

1.Obiectivul

Obiectivul acestei teme este de a proiecta si de a implementa, un program care proceseaza operatiile cu polinoame. Polinoamele sunt de o singura variabila, noi fiind nevoiti efectiv sa implementam adunarea, scaderea,inmultirea, impartirea, derivarea si integrarea polinoamelor. Altfel spus un calculator de polinoame cu o interfata grafica “User Friendly”, care sa poata sa fie utilizat cu usurinta de orice utilizator.

Subobiectivele problemei actuale sunt: analiza problemei si identificarea nevoilor, proiectarae calculatorului polinomial, implementarea calculatorului polinomial, si testarea calculatorului.

2.Analiza problemei si scenarii de utilizare

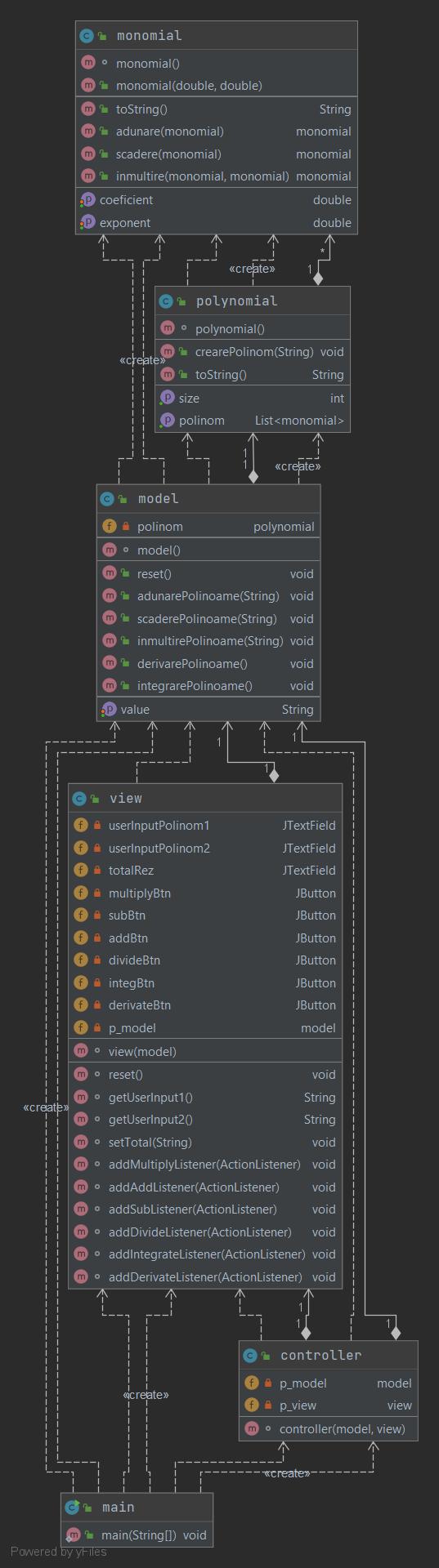
Plecand de la premiza ca un polinom este alcatuit din mai multe monoame, din care fiecare monom este alcatuit la randul lui dintr-un coeficient si dintr-un exponent, am ales sa implementez calculatorul in felul urmator: fiecare monom trebuie sa aiba si un coeficient si un exponent, nu vom putea scrie „x^2+x^1+1x^0” ci doar „1x^2+1x^1+1x^0”, deoarece am ales sa folosesc REGEX pentru a putea face ca polinoamele sa aiba orice grad.

Modul prin care se introduc informatiile, modul in care sunt salvate in memorie polinoamele, modul in care sunt sortate in ordinea gradelor, aspectul interfetei grafice poate sa difere de la programator la programator, modul de implementare stand la latitudinea fiecaruia.

Modul ales de mine, are o singura problema, aceea fiind ca fiecare monom trebuie sa aiba si coeficient si x si exponent, in rest nu ar putea aparea erori ulterioare, dar sun convins caci cu putin mai multa munca ar putea fi remediata si aceasta problema.

Pentru scenariile de utilizare, nu am implementa impartirea in rest, calculatorul functioneaza perfect, atata timp cat utilizatorul introduce atat un coeficient cat si un exponent la fiecare putere a factorului x. Nu am un buton de submit asa ca introduci primul polinom, il introduci pe al doilea si poti sa dai pe operatie si el garantat o sa functioneze.

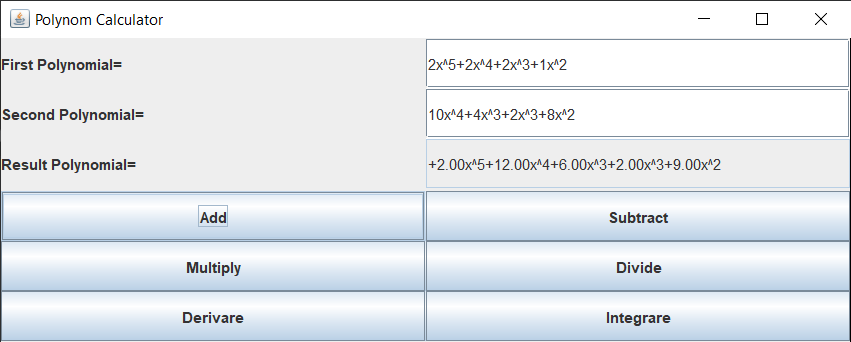
3.Proiectare

Diagrama UML a proiectului meu este:

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de [modele](https://ro.wikipedia.org/w/index.php?title=Modelare_orientat%C4%83_pe_obiect&action=edit&redlink=1) și [specificații](https://ro.wikipedia.org/w/index.php?title=Specifica%C8%9Bie&action=edit&redlink=1) pentru [software](https://ro.wikipedia.org/wiki/Software). Limbajul a fost creat de către consorțiul [Object Management Group](https://ro.wikipedia.org/w/index.php?title=Object_Management_Group&action=edit&redlink=1) (OMG) care a mai produs printre altele și standardul de schimb de mesaje intre sisteme [CORBA](https://ro.wikipedia.org/w/index.php?title=CORBA&action=edit&redlink=1). 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.

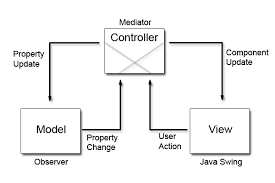
La ce revine analiza problemei? Este foarte simplu, avem nevoie de un calculator rapid de polinoame deoarece operatiile cu polinoame daca vei sta sa le faci pe hartie iti vor lua foarte mult timp si vedea ca sunt foarte dificile. Cum ar fi ca tu sa fi nevoit doar sa introduci polinoamele si sa ocupe calculatorul de restul??? Ei bine acum totul este posibil.

Pentru inceput vreau sa incep prin a descrie interfata grafica aleasa de mine, ea fiind una „USER FRIENDLY” adica usor de folosit de toata lumea, voi pune mai jos cum arata aceasta:



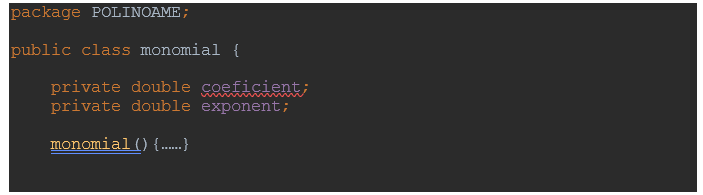
Pentru a aduna, scadea, sau inmulti avem nevoie de doua polinoame asfel noi va trebui sa introducem un polinom in prima casuta si in a doua iar in a treia va aparea rezultatul. Tot aici trebuie specificat faptul ca pentru integrarea si derivarea unui polinom avem nevoie doar de prima casuta unde vom introduce polinomul si rezultatul va aparea de asemenea in a treia casuta. De remarcat ca am folosit pachetul java.SWING pentru implementarea interfetei.

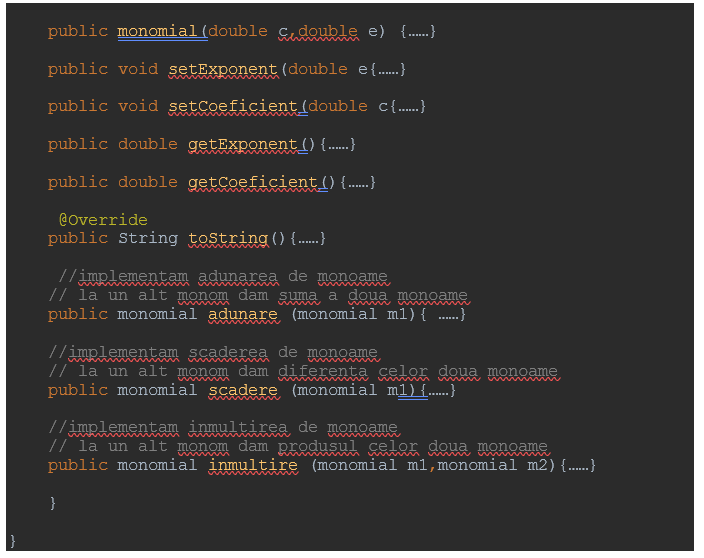
Pentru a gasi o solutie cat mai buna, am folosit un singur pachet cu numele de POLINOAME, in care am 6 clase si am ales sa folosesc patternul model-view-controller. In model am incapsulat date si am implementat functionalitati, in controller am implementat cate o subclasa care care implementiaza action listener, mai exact ascultatorii pentru butoare care asteapta ca noi sa apasam pe unul dintre ele. In view am implementat asa cum s-ar spune, asa zisa legatura dintre utilizator si calculator. Acest pattern sta la baza majoritara a aplicatiilor java cu interfata grafica.

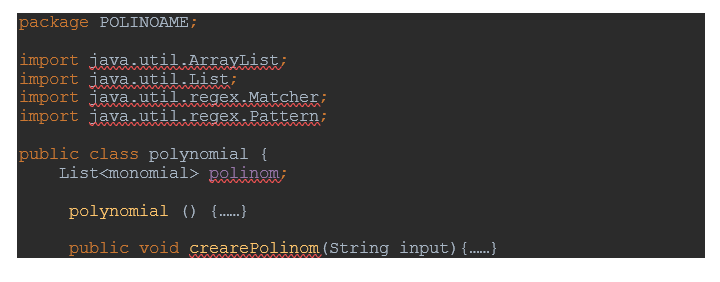


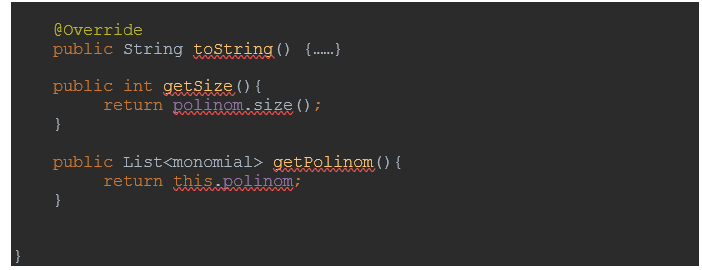
4.Implementare

Am decis sa implementez clasele cat mai intuitiv posibil, cu un cod cat mai lizibil si cat mai usor de inteles deoarece in viitor va fi nevoie ca o ehipa intreaga sa inteleaga codul pentru a putea lucra toti la acel proiect. O sa descriu fiecare clasa pe rand:

**Clasa monomial** care are 2 atribute de tip real cu nume sugestiv, coeficient si exponent care vor fi exponentul si coeficientul fiecarui monom. Mai multe monoame impreuna vor forma un polinom. In ea se mai pot gasi gettere si settere atat pentru coeficient cat si pentru exponent. Am suprascris metoda toString() ca sa pot afisa frumos in gui fiecare monom. Tot in aceasta clasa mai exista trei metode cu nume sugestiv: adunare, scadere si inmultire. Metoda adunare realizeaza adunarea dintre doua monoame ea facandu-se foarte usor, deoarece la adunarea polinoamelor adunam coeficientul care apare in fata lui x unde este acelasi exponent. Metoda scadere functioneaza similar cu adunarea, atat doar ca al doileqa coeficient se va scadea din primul. Inmultirea dintre 2 monoame este puti mai complicata deoarece ea nu se realizeaza doar cu puteri egale, eu am implementa-o asfel: am imultilt coeficientii si am adunat exponentii. Un mic preview poate fi vazut mai jos.



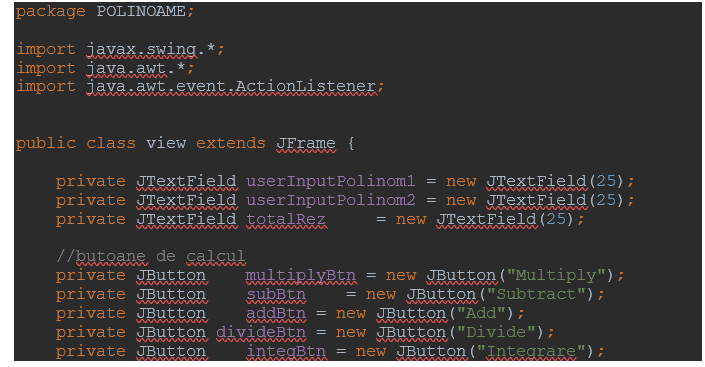
**Clasa polynomial** are un singur atribut, o lista de tipul array list de monoame dearece asa cum am spus si mai sus un polinom este alcatuit din mai multe monoame. Ca si metode am ales sa implementez doar suprascrierea toString() si crearea polinomului din stringul de inputul utilizatorului. Am facut acest lucru cu ajutorul lui regex care cauta anumite patterne intr-un string dat asfel reusint foarte usor sa implementez un polinom de orice grad , deoarece plec de la o premiza care nu este oarecum adevarata, ca orice monom va avea cu siguranta un coeficient si un exponent. 



**Clasa view** reprezinta implementarea interfetei grafice facute cu ajutorul pachetului java.swing, ea fiind foarte usor de implementat, de aceea am si ales-o dar si pentru ca deja o stiam folosi de semestrul trecut de la cursul de poo.

In ea am trei text fieldeuri cu nume sugestiv, userInputPolinom1 care reprezinta locul unde utilizatorul poate introduce primul polinom, userInputPolinom2 care reprezinta locul unde utilizatorul poate introduce al doilea polinom iar ultimul text field are numele totalRez unde se va afisa rezultatele operatiilor. Primele 2 fielduri pot fi editate, in schimb ultimul nu poate fi editat deoarece acolo va fi doar rezultatul. In continuare voi discuta despre cele sase butoane implementate care si ele la randul lor au nume sugestiv fiecare buton facand o operatie anume, de adunare, de scadere, de inmultire, de impartire, de derivare si de integrare.

Tot in aceasta clasa am adaugat ascultatori pentru fiecare buton, dar implementarea propriu-zisa fiind in clasa controller. In aceasta clasa mai exista trei metode cu care luam stringurile introdus in primul si al dolea filde dar si cu care vom afisa rezultatul in fieldul de rezultat.





**Clasa controller** are mai multe subclase, cate o subclasa pentru fiecare buton, mai exact cate o clasa pentru pecare ascultator al fiecarui buton. Am ca si atribute un model si un view deoarece controllerul trebuie sa faca legatura intre model si view.

In fiecare subclasa, deoarece nu am implementat fizic un buton de reset, am pe prima linie resetarea modelului, in caz ca se apasa mai multe butoane consecutiv sa poata exista o reinitializare a aplicatiei, mai apoi depinzand de ce operatie se selecteaza pot exista unul sau doua inputuri, unul singur doar pentru derivare si integrare,sin pentru restul operatiilor cate doua. Acolo unde este nevoie de un singur input trebuie obligatoriu sa introducem polinomul in primul text field pentru ca calculatorul sa poata indeplinii functia. Cu metodele getValue si setValue interactionez cu modelul pentru a instantia un nou polinom si pentru a transforma polinomul intr un sir de caractere, cu functiile getUserInput si setTotal interactionez cu view-ul pentru a lua valoarea introdusa de utilizator in text field dar si pentru a pune totalul in field-ul de rezultat. Despre operatiile propriu-zise o sa vorbesc in momentul in care o sa descriu clasa model. Un avantaj zic eu destul de mare este ca aici nu a fost nevoie sa creez instante de polinom doarece am implementat in asa fel incat toata magia sa se intample in clasa model. Si functiile de adunare, inmultire, scadere le-am apelat cu un string reusind asfel sa scurtez mult lungimea fiecarei subclase.



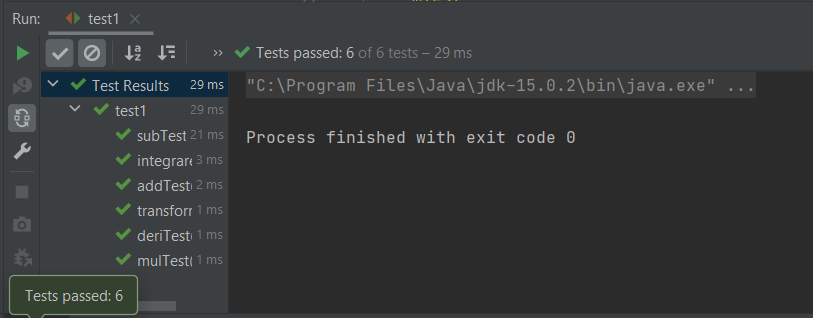
**Clasa model** descrie modul de functionare a calculatorului, mai exact implementeaza toate operatiiile pe care calculatorul le poate face, de a aduna doua polinoame, de a le sacadea, de al le inmulti, de a deriva un polinomm sau de a integra unul, toate acestea fiind scrise in locul potrivit, dupa cum consider eu, deoarece acesta este scopul modelului intr-o arhitectura model – view – controller.

Singurul atribut al acestei clase este o instanta de tipul polinom pentru a putea apela creearea de polinom si toStringul din el deoarece avem nevoie de ele in metodele getValue si setValue. In metoda setValue se creeaza initializeaza instanta cu un nou polinom, mai exact cu ajutorul metodei din polinom transforma din string intr-un polinom asa cum l-am conceput eu pentru functionare. Pentru metoda setValue se apeleaza metoda suprascrisa toString din polinom, ca sa se poata transforma din polinomul conceput de mine in string.

Toate celelate metode ramase sunt doar operatii pe polinoame, in urmatoarele randuri o sa descriu modul in care functioneaza acestea. Metoda de inmultire are un mic defect, adica presupune ca utilizatorul are gata sortate polinoamele in ordine descrescatoare a gradelor, dar daca utilizatorul este atent si indeplineste acest principiu adunarea se realizeaza cu succes. Principiul ales de mine este un principiu de interclasare, aduna doar monoame de grade egale, daca nu la rezultat adauga monomul cu grad mai mare urmand ca mai apoi sa verifice care dintre cele doua polinoame are grad mai mare si tot asa. Dupa ce iese din prima structura repetitiva, programul se asigura ca amandoua polinoame pe care le adunam au ajus la finl iar daca nu adauga ce a ramas la rezultat, urmand ca mai apoi rezultatul sa fie inscris inapoi in primul polinom. Principiul scaderii este asemanator, stiu ca la scadere puteam sa o implementez mult mai simplu, inmultind cu minus al doilea polinom si adunandu-l la primul si rezultand asfel o scadere, pe moment nu mi-a venit aceasta idee, asa ca am implementat to printr-o interclasare.

Metoda folosita la inmultire este una destul de usor de implementat dar si usor de inteles, am facut o copie primul polinom si urmand mai apoi sa il eliberez pentru a putea sa il folosesc in adunare viitoare. Acum, voi inmulti primul monom al celui de-al doilea polinom cu fiecare monom din primul polinom, adunand ce rezulta la rezultatul final si asfel reusind sa implementez usor inmultirea.

Pendru derivarea polinoamelor, am ales o implementare destul de scurta si destul de usor de inteles. Am parcurs fiecare monom al polinomului curent, am inmultit coeficientul cu exponentul, si am initializat cu aceasta valoare noul coeficient, iar noua putere noului monom am initializat cu puterea monomului curent scazand o unitate din ea. Pentru integrare modul de implementare este asemanator, tot ce difera este faptul ca puterea va creste cu o unitate iar coeficientul se va impartii cu valoarea noului exponent.

5.Rezultat

Pentru rezultate, am implementat o clasa de test cu numele test1 unde am facut teste pentru toate functionalitatile calculatorului cu ajutorul unei testari unitare implementate cu Junit, mai exact cu ajutorul metodei din Assertions, assertEquals unde am dat polinomul rezultat in urma operatiilori si rezultatul calculat de mine pe hartie. Am facut cate o metoda de testare pentru urmatoarele operatii: transpormarea din string in polinom, adunare, scadere, inmultire,impartire, derivare si integrare. La final daca clasa de test este rulata toate testele vor fi trecute cu succes.

6.Concluzii

In concluzie, acest proiect m-a invatat si m-a ajuta sa folosesc mult mai bine patternul model view controller, si cum sa leg clasele intre ele pentru o functionare cat mai corecta a modelului si pentru a putea implementa cat mai riguros controlul aplicatiei. Pentru aplicatiile ulterioare s-ar putea implementa si operatia de impartire si si calculul radacinilor polinomului dar de a dezvolta un pattern care sa functioneze si daca nu introducem u coeficient sau un exponent. Viitori utilizatori ar putea fii elevii de clasa a 12-a deoarece acest calculator i-ar ajuta destul de mult, stiu din propria experienta.

7.Bibliografie

Principala sursa de inspiratie a fost wikipedia dar si w3school dar si documentatia de la oracle pentru a putea folosii anumite functionalitati a anumitor metode, am mai consultat si anmite surse de inspiratie pentr-u a-mi amintii unele operatii si cum se realizeaza pe hartie, pentru a reusi sa le implementez.