**TEHNICI DE PROGRAMARE**

**ASSIGNMENT 1**

**POLYNOMIAL CALCULATOR**

**Chelemen Stefana-Bianca**

**GRUPA 30221**

**1.Obiectivul temei**

Aceasta tema are ca obiectiv implementarea unui calculator de polinoame ,,prin intermediul unui program Java,in cadrul carei vom putea realiza operatiile de baza ale polinoamelor:adunarea,scaderea,inmultiea,impartirea.,In plus, vom putea realiza si operatii mai complexe cum ar fi integrarea si derivarea. Vom considera polinoamele introduce in acest calculator cu coeficienti intregi,si de o variabila.

Ca obiective secundare amintim:

1. Descrierea clasei “Monomial”:care reprezinta clasa cu ajutorul careia se pot realiza celelalte clase
2. Descrierea claselor “IntMonomial” si “DoubleMonomial” care mostenesc clasa “Monomial”

si reprezinta monomae co coeficienti intregi/reali.Am ales sa folosesc doua clase diferite deoarece la unele operatii polinoamele rezultate se afla cu coef reali,nu intregi.

1. Descrierea clasei “Polynomial” ,care reprezinta o lista de monoame
2. Descrierea unor metode care transforma un sir de caractare intr-un polinom(lista de monoame) si invers.
3. Descrierea unor metode care implementeaza niste operatii auxiliare folosite pentru operatii cu polinoame(de ex puterea cea mai mare dintr-un polinom).
4. Descrierea clasei “Operations”,unde se intalnesc toate operatiilerealizate de calculatorul de polinoame:adunare,scadere,inmultire,impartire,derivare,integrare.
5. Descrierea clasei “TestPolyOperations”,unde sunt testate cu Junit operatiile implementate in clasa “Operations”.
6. Descrierea claselor “Viewer”,”Model”,”Controller”,care impreuna reprezinta interfata grafica servita perntru comunicarea intre user si calculator.

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

***2.1.Analiza problemei***

Obiectivul principal al acestui proiect este de a realiza un calculator polinomial cu ajutorul limbajului de programare Java, folosind paradigmele programarii orientate pe obiect. Acest calculator trebuie sa execute corect operatiile simple cu polinoame si anume adunarea,scaderea,inmultirea si impartirea dar si unele mai complexe cum ar fi derivarea si integrarea. Pentru a realiza aceste operatii, avem nevoie de operanzi si anume polinoame. Eu am ales sa reprezint aceste polinoame sub forma de lista de monoame. Pentru reprezentarea monoamelor am ales sa numesc o clasa Monomial care are ca atribut puterea monomului. In plus,am ales sa creez si alte 2 clase, IntMonomial si DoubleMonomial care sa mosteneasca clasa Monomial sis a aiba ca atribute coeficienti de tip integer/double dupa caz. Esentiale in aceasta aplicatiile sunt operatiile, dar si extragerea corecta a datelor introduse de catre utilizator pentru a putea realizarea operatiilor.

***2.2.Modelare***

Dupa cum am mentionat anterior, este foarte importanta extragerea corecta a datelor introduse de catre utilizator cu scopul de a le prelucra si folosi in cadrul operatiilor efectuate de catre calculatorul polinomial.

Utilizatorul va introduce in interfata un sir de caractere, dupa care acesta va fi transformat intr-o lista de monoame cu atributele pow(putere) si coefficient(coeficient).

***2.3.Scenariul***

Daca dorim ca aplicatia sa ruleze corect, fara erori,utilizatorul trebuie sa urmareasca anumite reguli la introducerea polinoamelor. Trebuie folosita litera “X” ca variabila pentru polinom,iar , unmonom al polinomului trebuie sa aiba urmatorul pattern: coeficientX^putere. Dupa cum am spus, un polinom este reprezentat ca sir de monoame, deci la introducerea polinomului monoamele trebuie delimitate de un symbol. Acest simbol poate fi “+” sau “-”. De asememea trebuie sa tinem cont de faptul ca primul termen din polinom nu poate s ainceapa cu “-“. Pentru intelegerea structurii corecte a datelor ce trebuie introduse, urmarim exemplele urmatoare:

“X^8+5X^1+2X^0” –corect

“1X^9”-corect

“X^2 + X^7” –gresit (nu este permisa folosirea spatiilor intre putere si semnul+(sau -))

***2.4.Cazuri de utilizare***

La rularea aplicatiei, pe ecran va aparea fereastra calculatorului polinomial(interfata grafica).

Aceasta fereastra contine un panou pe care sunt asezate toate componentele cum ar fi butoane sau textField-uri.

Acest panou contine 2 etichete care ne indeamna sa introducem polinoamele, urmate de textField-uri destinate scrierii polinoamelor. In partea dreapta a panoului putem observa butoanele aplicatiei, numite sugestiv pentru fiecare operatie care poate fi executata de catre calculator. Dupa introducerea polinoamelor, la apasarea unuia dintre aceste butoane, se va afisa pe o eticheta (indicate de catre simbolul ↵) rezultatul operatiei alese. De asemnea, exista si un buton de “Clear” care va scapa de fereastra curenta si va genera una noua.

**3.Proiectare**

Clasa Monomial

Acesta clasa este punctul de plecare in constructia polinoamelor. Este o clasa parinte pentru 2 sublase, si are ca atribut “pow”, care defineste puterea fiecarui monom si atributul “op” folosit ca un semafor la operatiile de adunare/scadere. Ca metode avem doar metode accesoare si mutatoare: getPow-returneaza valoare puterii monomului actual,setPow-seteaza valoare puterii monomului actual, setOp-seteaza “semaforul ” monomului actual, getOp-returneaza “semaforul” polinomului actual.

Clasa IntMonomial

Aceasta clasa mosteneste clasa Monomial. Pe langa atributele si metodele mostenite, aceasta clasa are si atrbutul coefficient care tine coeficientul intreg al monomului. Ca metode avem metode accesoare si mutatoare getCoefficient/setCoefficient –returneaza/seteaza coeficientul intreg al monomului actual. In plus, mai avem metoda compareExponentsIntCoeff care compara monoamele intregi in functie de puterea monoamelor; o vom folosi la sortarea unui polinom. Exista si o metoda suprascrisa :toString()

care ajuta la afisarea monoamelor.

Clasa DoubleMonomial

Aceasta clasa mosteneste clasa Monomial. Pe langa atributele si metodele mostenite, aceasta clasa are si atributul coefficient care tine coeficientul real al monomului. Ca metode avem metode accesoare si mutatoare getCoefficient/setCoefficient –returneaza/seteaza coeficientul real al monomului actual. In plus, mai avem metoda compareExponentsDoubleCoeff care compara monoamele reale in functie de puterea monoamelor; o vom folosi la sortarea unui polinom. Exista si o metoda suprascrisa :toString()

care ajuta la afisarea monoamelor.

Clasa Polynomial

Aceasta clasa este folosita pentru a defini un polinom. Reprezentam un polinom ca si o lista de Monoame(intregi sau double). Metodele din aceasta clasa sunt:”addMonomToPoly” care a fost implementata pentru a putea adauga un monom nou in lista de monoame deja existenta aferenta unui Polinom,”printPoly” care a fost implementata pentru printarea unui polinom sub forma de string(“P(X)= ”), “reduceIntegerPolynom” , care sorteaza un polinom cu coeficienti intregi si il reduce (de exemplu daca intr-un polinom am avea 2X^4 + 3X^4 metoda il va reduce la 7X^4), “reduceDoublePolynom” care face acelasi lucru ca si reduceIntegerPolynom numai ca schimba coeficientii intregi cu coeficieti reali, “maxMonom” care afla monomul cu puterea maxima dintr-un polinom, “isEmpty” care verifica daca un polinom cu coeficienti intregi este gol, “isDoubleEmpty” care verifica daca un polinom cu coeficienti reali este gol, “maxPow” care afla cea mai mare putere din polinom,”mulMonom” care inmulteste un polinom cu un monom transmis ca parametru,”toString()” care afiseaza un polinom asa cum este aceasta introdusa de catre un utilizator, si “tooPoly” care transforma un string intr-un polinom valid(lista de monoame).

Clasa Operations

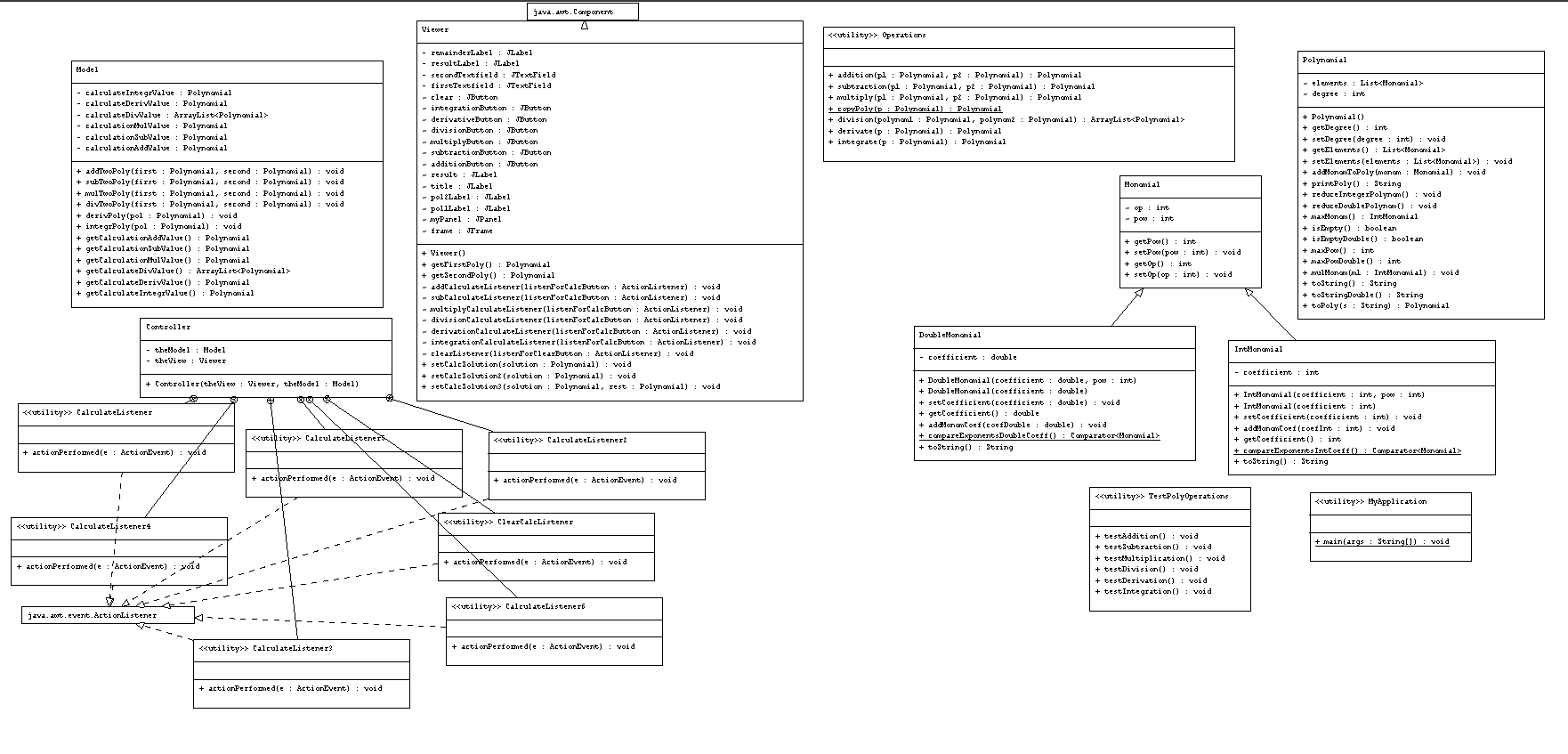
Clasa Operations a fost implementata pentru a realiza separat fata de clasa Polynomial, operatiile unui calculator polynomial. Aceasta are metode cu nume specifice pentru fiecare operatie: “addition”,”subtraction”,”multiply”,”division”,”derivate”, “integrate” .Pe langa aceste metode care realizeaza operatiile cu polinoame am mai definit o metoda copyPoly care copiaza continutul unui polinom in alt polinom(deoarece nu puteam implementa aceasta operatie cu ” = “)

Al doilea pachet se numeste **GUI**, si acesta contine 3 clase:Model,Viewer si Controller. In aceste clase se creeaza interfata grafica folosind un pattern MVC.

Clasele Viewer,Model,Controller

Structura detaliata a acestor clase o voi explica in sectiunea urmatoare. Am ales sa implementez modelul “Model Viewer Controller” in realizarea interfetei grafice din acest proiect, deoarece am dorit sa izolez implementarea de interfata vizibila utilizatorului. Astfel, a rezultat o aplicatie mai usor de depanat si modificat ulterior.

In urmatoarea imagine este prezentata diagram UML a proiectului, genereata in IntelliJ:



**4.Implementare**

In ceea ce priveste implementarea, am ales ca programul sa fie impartit in mai multe pachete, iar clasele din fiecare pachet sa aiba aceleasi caracteristici, pentru a respecta paradigmele programarii orientate pe obiect..

Primul pachet se numeste **DataModels**, si acesta contine clasele:Monomial,IntMonomial,DoubleMonomial,Polynomial si Operations.

Clasa Monomial

* Defineste un monom
* Are ca atribute “pow”, care semnifica puterea monomului si “op” care semnifica un semafor si il folosim la operatiile de scadere/adunare sa verificam daca un anumit termen a fost adunat/scazut
* Ca metode are doar setteri si getteri

Clasa IntMonomial

* Defineste un monom cu coeficienti intregi
* Mosteneste clasa Monomial
* Are ca atribut “coefficient”, care semnifica coeficientul intreg al monomului
* Are getteri si setter
* Pe langa metodele accesoare/mutatoare avem o metoda “compareExponentsIntCoeff”, care o s-o folsim la sortarea unui polinom. Aceasta compara monoamele in functie de putere
* Contine si o metoda toString (suprascrisa) utila la afisare

Clasa DoubleMonomial

* Defineste un monom cu coeficienti reali
* Mosteneste clasa Monomial
* Are ca atribut “coefficient”, care semnifica coeficientul real al monomului
* Are getteri si setter
* Pe langa metodele accesoare/mutatoare avem o metoda “compareExponentsDoubleCoeff”, care o s-o folsim la sortarea unui polinom. Aceasta compara monoamele in functie de putere
* Contine si o metoda toString (suprascrisa) utila la afisare

Clasa Polynomial

* Defineste un monom ca lista de monoame
* Are setter si getteri
* Metode:
  + - * “addMonomToPoly”-adauga monomul transmis ca paramtru in lista de monoame
      * “printPoly”- metoda care afiseaza un polinom sub forma “P(X)=…”
      * “ReduceIntegerPolynom”/”reduceDoublePolynom”-metode care sorteaza si simplifica un polinom
      * “maxMonom”- metoda care afla monomul cu puterea maxima dintr-un polinom
      * “isEmpty”/”isEmptyDouble”-metode care verifica daca un polinom este gol
      * “maxPow”/”maxPowDouble”-afla puterea maxima dintr-un polinom
      * “toString()”/”toStringDouble”-returneaza polinomul transmis ca si parametru ca string
      * “toPoly”-transforma un string intr-un polinom(lista de monoame)
      * “mulMonom “-inmulteste un polinom cu monomul dat ca parametru

Clasa Operations

* Defineste operatiile effective ale calculatorului
* Metode:
  + - * “addition”-adauna doua polinoame
      * “subtraction”-scade doua polinoame
      * “multiply”-inmulteste doua polinoame
      * “division”-imparte doua polinoame
      * “derivate”-deriveaza un polinom
      * “integrate”-integreaza un polinom

Al doilea pachet se numeste **GUI**, si acesta contine 3 clase:Model,Viewer si Controller. In aceste clase se creeaza interfata grafica folosind un pattern MVC.

Clasa Viewer

* + Sarcina acestei clase este de a afisa tot ce trebuie sa vada utilizatorul in interfata grafica
  + Aici nu se efectueaza colculele propriu-zise, ci preia informatiile introduse de catre utilizatori si le paseaza mai departe catre clasa Controller
  + Atribute:aceasta clasa are ca atribute, toate componentele afisate in interfata grafica
* Metode:
  + - * “getFirstPoly”- metoda care ia din primul textField si transforma stringul in polinom
      * “getSecondPoly”- metoda care ia din al doilea textField si transforma stringul in polinom
      * “addCalculateListener”

Adauga ActionListeneri la fiecare buton din interfata. Daca un anumit buton este apasat, se executa metoda in clasa Controller

* + - * “subCalculateListener”
      * “multiplyCalculateListener”
      * “divisionCalculateListener”
      * “derivationCalculateListener”
      * “integrationCalculateListener”
      * “clearListener”

Afiseaza in eticheta corespunzatoare rezultatului,rezultatul operatiei.

* + - * “setCalcSolution”
      * “setCalcSolution2”

Clasa Model

* + Efectuează toate calculele necesare.
  + Se comporta ca si cand nu stie de existenta clasei Viewer
  + Atribute:
    - * “calculationAddValue”-pentru rezultatul adunarii
      * “calculationSubValue”-pentru rezultatul scaderii
      * “calculationMulValue”-pentru rezultatul inmultirii
      * “calculationDivValue”-pentru rezultatul impartirii
      * “calculationDerivValue”-pentru rezultatul derivarii
      * “calculationIntegrValue”-pentru rezultatul integrarii
  + Metode:
    - * Getteri si setter
      * “addTwoPoly”-efectueaza operatia de adunare
      * “subTwoPoly”-efectueaza operatia de scadere
      * “mulTwoPoly”-efectueaza operatia de inmultire
      * “divTwoPoly”-efectueaza operatia de impartire
      * “derivTwoPoly”-efectueaza operatia de derivare
      * “integrTwoPoly”-efectueaza operatia de integrare

Clasa Controller

* + Controleaza interactiunile dintre clasa Viewer si clasa Model
  + Atribute:
    - * “theView”-pentru View-ul interfetei
      * “theModel”-pentru Modelul interfetei
  + Are un constructor in care initializeaza view-ul si modelul si creeaza noi instante ale unor clase interioare in care se executa metode de tip Listener
  + Contine 6 clase interioare:

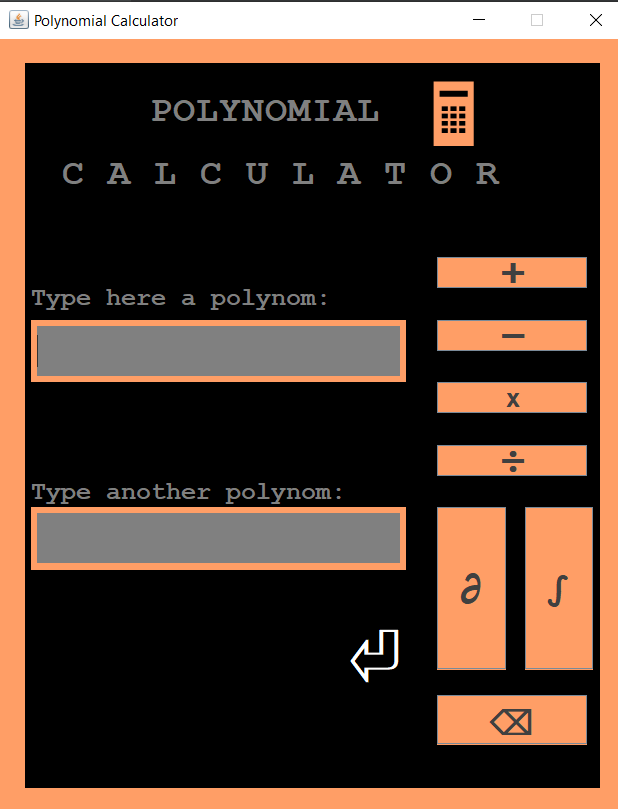
In fiecare clasa interioara este implementata o metoda de tip ActionPerformed.

* + - * “CalculateListener”
      * “CalculateListener2”
      * “CalculateListener3”
      * “CalculateListener4”
      * “CalculateListener5”
      * “CalculateListener6”
      * “ClearCalcListener”

In al treilea pachet “ap”, am implementat o singura clasa: “MyApplication”. In aceasta clasa am creat doar o metoda statica main in care am lansat in executie aplicatia creata.

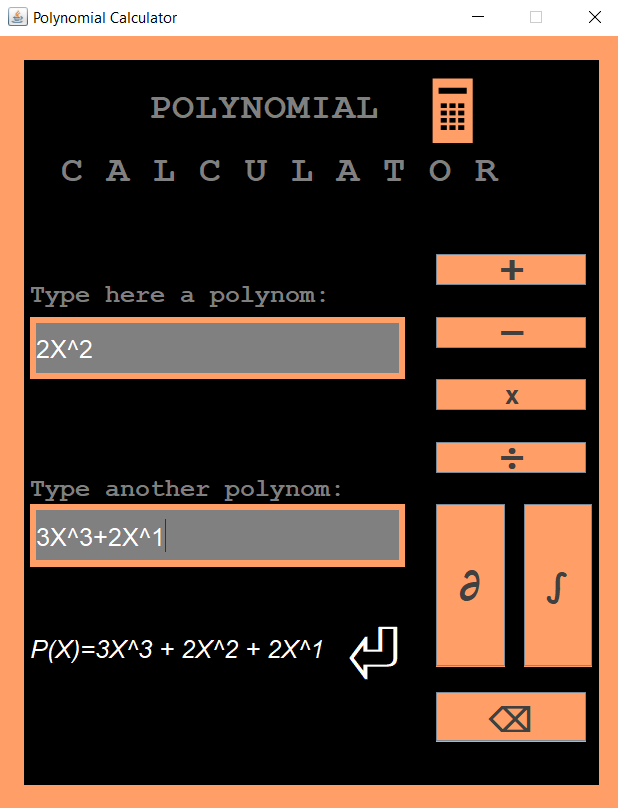
**5.Rezultate**

Pentru a exemplifica functionarea aplicatiei, incep cu prezentarea ferestrei aparute la pornirea aplicatiei:



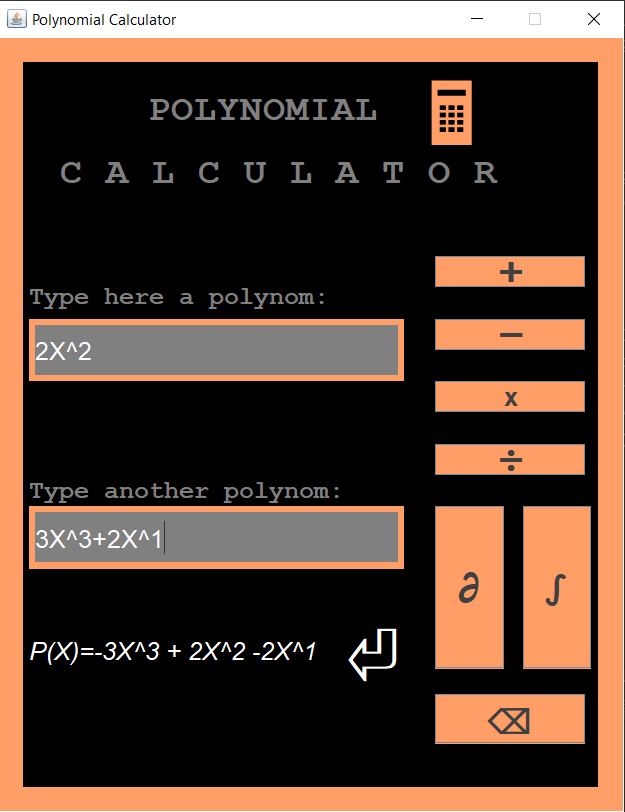
Dupa cum putem observa din imaginea atasata, aplicatia contine doua Field-uri unde utilizatorul poate sa introduca date de intrare catre program si 7 butoane care pot si apasate pentru a genera un rezultat dorit.

In urmatoare imagine voi demonstra functionalitatea de adunare a calculatorului:



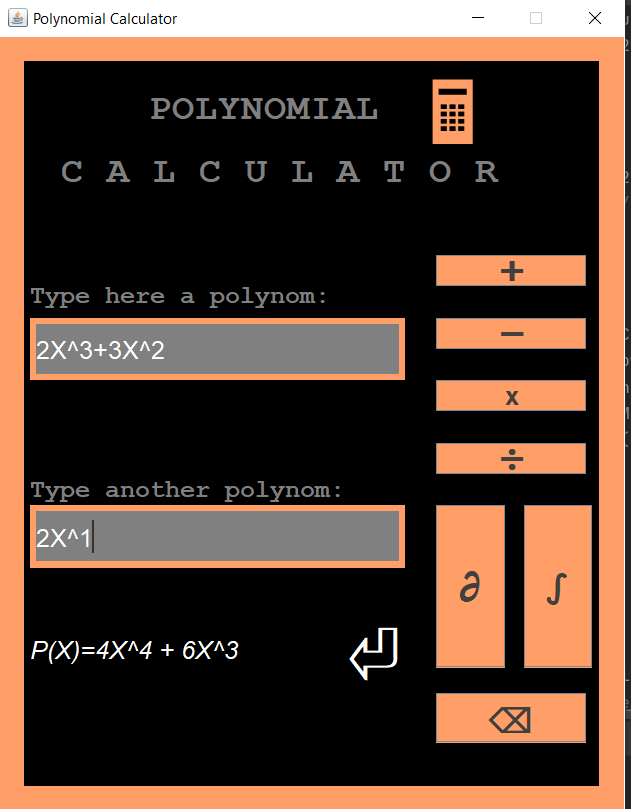
Ca date de intrare am ales sa introduc 2X^2 si 3X^3+2X^1. Dupa apasarea butonului “+”, se poate observa ca rezultatul afisat (P(X)=3X^3+2X^2+2X^1) este corect din punct de vedere matematic.

In urmatoare imagine voi demonstra functionalitatea de scadere a calculatorului:



Ca date de intrare am ales sa introduc 2X^2 si 3X^3+2X^1. Dupa apasarea butonului “-”, se poate observa ca rezultatul afisat (P(X)=-3X^3+2X^2-2X^1) este corect din punct de vedere matematic.

In urmatoare imagine voi demonstra functionalitatea de inmultire a calculatorului:



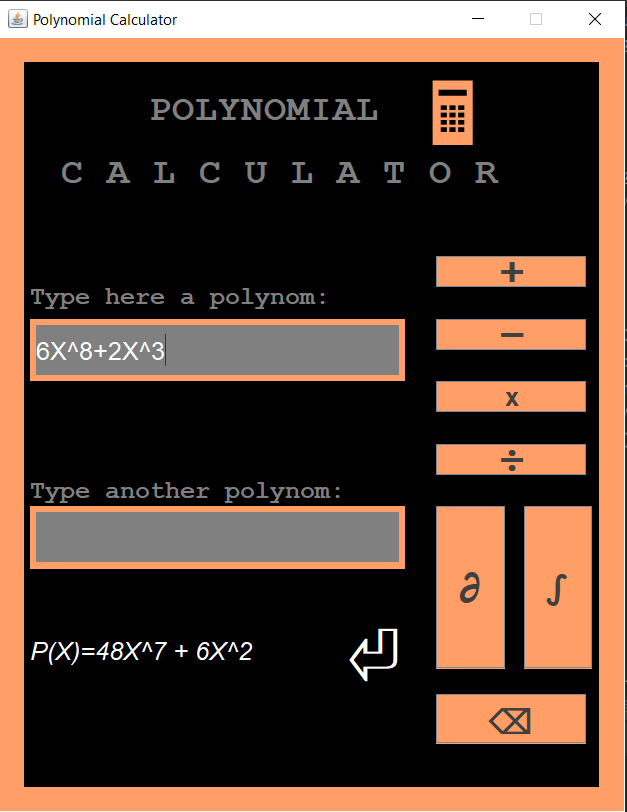
Ca date de intrare am ales sa introduc 2X^3+3X^2 si 2X^1. Dupa apasarea butonului “X”, se poate observa ca rezultatul afisat (P(X)=-4X^4+6X^3) este corect din punct de vedere matematic.

In continuare, voi demonstra functionalitatea de impartire a calculatorului:



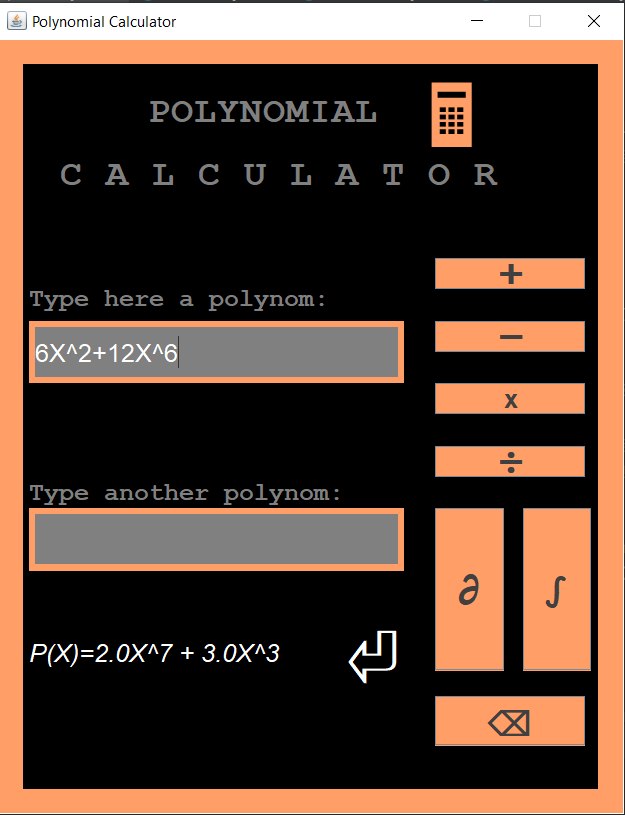
Ca date de intrare am ales sa introduc 6X^3+3X^1 si 2X^3+1X^0. Dupa apasarea butonului pentru impartire, se va afisa Q(X)=3X^0 –care este corespunzator catului impartirii si R(X) care este corecpunzator restului impartirii. Daca efectuam calculele matematice. Observam ca si aceste rezultate sunt corecte.

In continuare, voi demonstra functionalitatea de derivare a calculatorului:



Ca date de intrare am ales sa introduc 6X^8+2X^3. Dupa apasarea butonului de derivare rezultatul afisat este P(X)=48X^7+6X^2. Si acest rezultat este corect.

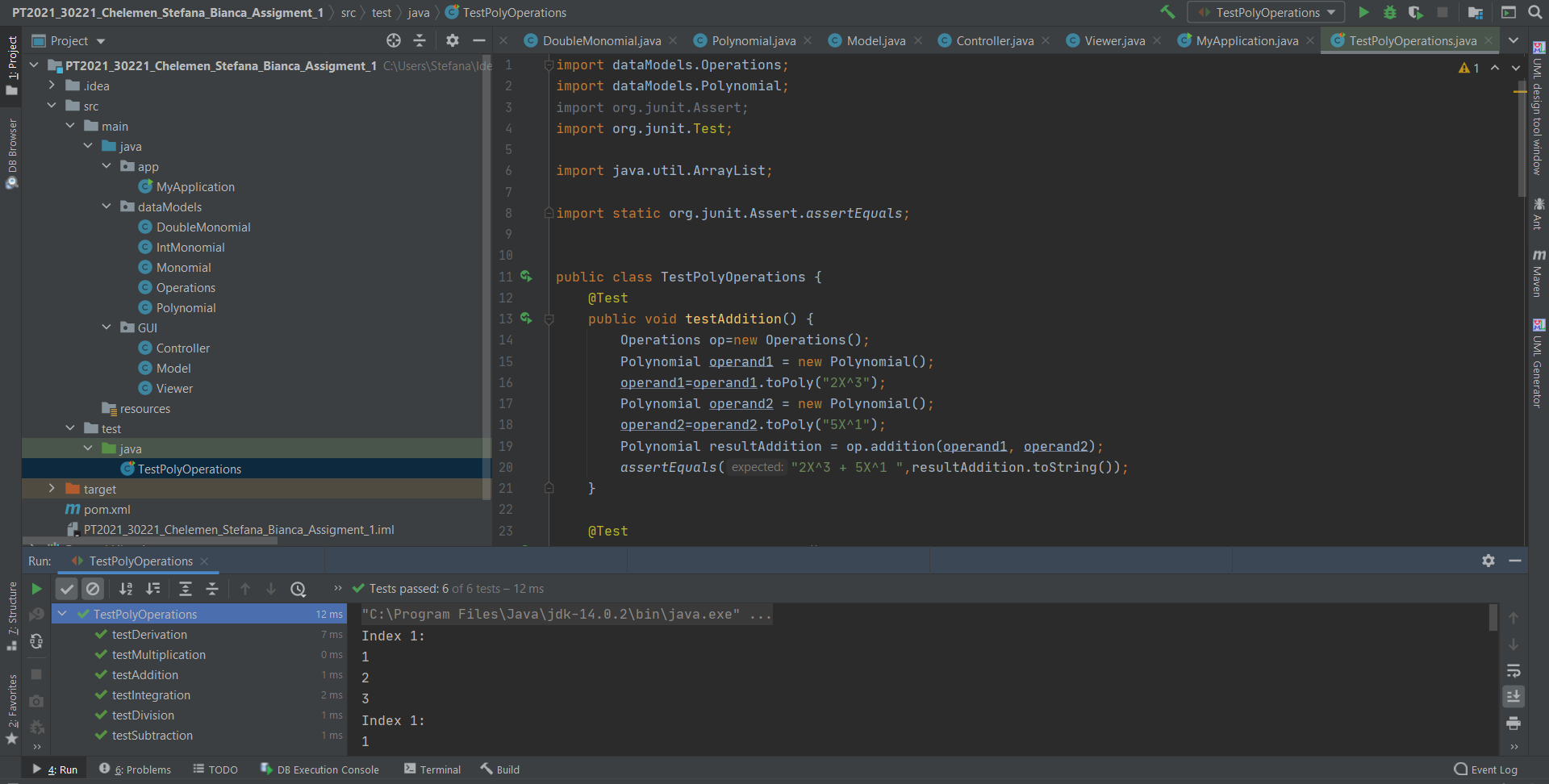
In urmatoarea imagine, voi demonstra functionalitatea aplicatiei la apasarea butonului de integrare:



Pentru intrare am ales sa folosesc un polinom simplu, pentru ca impartirile sa se faca exact. Am introdus 6X^2+12X^6. Dupa apasarea butonului de intregare, obtinem rezultatul dorit.

La apasarea butonului de Clear va aparea o noua fereastra “curatata”.

Rezultatele testarii cu Junit:



**5.Concluzii**

In concluzie, in opinia mea aceasta tema a fost utila pentru reimprospatarea conostiintelor dobandite pana acum inceea ce pirveste limbajul de programare Java, paradingmele programarii orientate pe obiect dar si a interfetelor grafice. Sunt multumita de ce am realizat deoarece am reusit sa implementez toate cerintele listate la prezentarea temei si am implemenatat o aplicatie usor de utilizat pentru un user neexperimentat.

**6.Bibliografie**

1.Wikipedia

2.Youtube

3. https://www.geeksforgeeks.org/

4. https://stackoverflow.com/