**TEHNICI DE PROGRAMARE FUNDAMENTALE**



**ASSIGNMENT I:**

**Calculator de polinoame**

**Documentatie**

STUDENT: MARCU MIHAI-ALEXANDRU

GRUPA: 30227

**CUPRINS:**

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

a. Clasa Monom;

b. Clasa Polinom;

c. Clasa Operations;

d. Metodele interfetei.

1. Rezultate;
2. Concluzii;
3. Bibliografie.
4. Obiectivul temei:

Obiectivul principal al temei date este realizarea unui calculator de polinoame care contine o interfata grafica dedicata cu ajutorul careia utilizatorul are posibilitatea de a introduce polinoame, de a selecta operatia pe care calculatorul sa o execute si de a afisa rezultatul obtinut pe ecran.

Obiectivele secundare ale proiectului sunt urmatoarele:

* Realizarea transformarii dintr-un sir de caractere introdus de catre utilizator intr-un polinom, metoda care va fi descrisa in capitolul “Implementare”;
* Implementarea operatiilor care se aplica pe polinoame: acestea vor fi descries la capitolul “Implementare” din acest document;
* Realizarea unei interfete grafice care sa i permita utilizatorului interactiunea cu calculatorul de polinoame, subiect care va fi detaliat in capitolul “Proiectare”;
* Crearea unui test unitar care va verifica corectitudinea operatiilor implementate: subiect care va fi de asemenea tratat la capitolul “Implementare”.

1. Analiza problemei, modelare, scenarii si cazuri de utilizare:

In prima instanta am abordat problema convertirii datelor introduse de utilizator intr-un polinom. Am realizat aceasta transformare prin implementarea unei functii care va fi descrisa mai amanuntit la capitolul de implementare.

Cazurile de utilizare reprezinta o lista de actiuni care descriu interactiunile dintre un utilizator si o aplicatie. Scenariul de utilizare al acestui proiect este urmatorul: utilizatorul introduce 2 polinoame de la tastatura, sub forma de sir de caractere care are posibilitatea de a avea mai multe forme, acestea fiind: “2\*x^2+2”,”2x^2+2”,”2x2+2” sau “2\*x2”. Daca datele introduce nu sunt de una dintre formele descries anterior, atunci sistemul va afisa o fereastra cu mesajul “Incorrect data input!”. Aceasta introducere de date este urmata de selectarea unui buton care reprezinta o operatie, optiunile fiind: addition, subtraction, multiplication, division, derivative sau integration. Dupa selectarea operatiei dorite se va afisa pe ecran rezuzltatul operatiei dupa caz. Utilizatorul are si optiunea de a sterge datele introduse si cele din rezultat prin apasarea butonului “clear”.

1. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator):

Pentru inceput, am creat doua clase Monom si Polinom, in care am descris structura si caracteristicile expresiilor cu acelasi nume, urmand realizarea clasei Operations in care sunt implementate operatiile pe care le poate realiza calculatorul de polinoame. Dupa aceea am creat clasele ViewCalc, ControllerCalc si MainClass, care impreuna cu clasa Operations reprezinta interfata grafica cu numele “Polynomial Calculator”. La final am realizat clasa JUnitTest care are scopul de a verifica corectitudinea operatiilor.

Structura de date pe care am utilizat-o in acest proiect este un arraylist alcatuit din elemente de tipul Monom care se regaseste in clasa Polinom. Un polinom este reprezentat de un arraylist de tipul acesta. Un monom este de una dintre urmatoarele forme: “5\*x^1”, “5x^1”, “5\*x” sau ”5x” si este alcatuit dintr-un coeficient de tip double(Am ales sa utilizez tipul double in defavoarea tipului int pentru rezultatele obtinute dupa operatiile de impartire si de integrare.) si un grad de tip int.

Interfata grafica este alcatuita din doua campuri de introducere a datelor, fiecare specifica unui polinom, sase butoane fiecare reprezentand o operatie, un buton „Clear” pentru eliberarea campurilor de intrare si rezultat si doua campuri pentru rezultate. Al doilea camp de rezultat este utilizat doar la operatiile de impartire, derivare si integrare, la impartire acesta semnifica restul acesteia,iar la derivare si integrare reprezinta derivata, respectiv integrala celui de al doilea polinom introdus de catre utilizator.

Interfata arata in felul acesta:

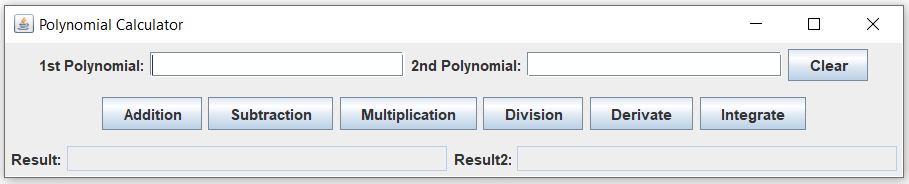
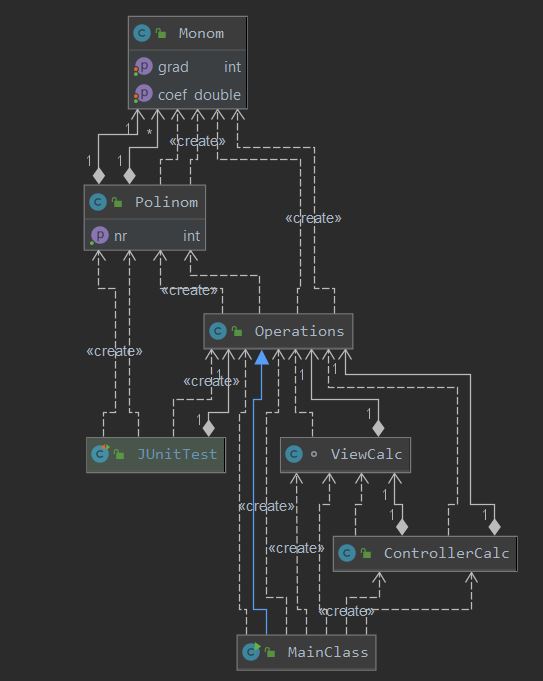


Diagrama UML a claselor este:



1. Implementare:
2. Clasa Monom: Campurile acestei clase sunt coef, care reprezinta coeficientul monomului si este de tip double si grad, care reprezinta gradul monomului si este de tip int. Ambele sunt de tip private.

Metodele acestei clase sunt urmatoarele:

* Un constructor care initializeaza un monom nou cu un coef si grad date;
* Settere si gettere atat pentru coef cat si pentru grad;
* O metoda compareTo care ajuta la sortarea monoamelor descrescator dupa grad;
* toMonom: realizeaza transformarea din sir de caractere s care este dat ca si parametru intr-un obiect de tipul Monom. Am prelucrat aceasta metoda astfel incat utilizatorul are posibilitatea de a introduce sub mai multe forme monomul, cu ajutorul functiei replace(). Am utilizat regex, despartind monomul in coef si grad cu ajutorul functiilor split() si replace().
* toString: realizeaza opusul metodei toMonom, adica transforma un obiect de tip Monom intr-un obiect de tip String. Rezultatul pentru monoamele cu grad mai mare decat unu va fi afisat sub forma: „coef\*x^grad”, de exemplu: „2.0\*x^2”. Deoarece am dat campului coef tipul double afisarea numerelor intregi va avea „.0” la final. Pentru monoame de gradul 1 afisarea va fi de forma „coef\*x”, iar pentru cele de grad 0 va fi „coef”.

1. Clasa Polinom: Clasa Polinom este alcatuita dintr-un arraylist polynomial care contine obiecte de tipul Monom si nr de tip int care reprezinta numarul de monoame din arraylist, acest numar incrementandu-se odata cu adugarea unui monom in arraylist.

Metodele acestei clase sunt urmatoarele:

* Un constructor fara parametri care initializeaza un polinom nou;
* Un getter pentru nr;
* Metoda addMonom: are ca si parametru un obicet de tip Monom si are scopul de a-l adauga la arraylist-ul polinomului care apeleaza aceasta functie. Aceasta functie are si scopul de a restrange polinomul, adica daca utilizatorul introduce de exemplu stringul: „2x+x”, metoda va adauga in arraylist monomul cu grad 1 si coef 3 echivalent stringului „3x”. In aceasta metoda se si modifica nr in functie de adugarea monoamelor in arraylist;
* Metoda toPolynomial: : realizeaza transformarea din sir de caractere s care este dat ca si parametru intr-un obiect de tipul Polinom. Am utilizat functiile split() si replace() pentru a desprati sirul de caractere in expresii, dupa care am transformat sirurile de caractere in monoame cu ajutorul functiei toMonom din clasa Monom, pe care le-am adaugat in arraylist-ul polynomial din polinom folosind functia polynomial.add(monom);
* toString: realizeaza transformarea din obiect de tip Polinom intr-un sir de caractere. Aceasta metoda parcurge arraylist-ul polynomial si apeleaza functia toString din clasa Monom pentru a transforma monoamele parcurse in siruri de caractere. Daca un monom are coeficientul sau gradul egal cu 0 atunci sirului de caractere ii va fi adaugat stringul „0”;
* Metoda delete0: este o metoda pe care am implementat-o cu scopul de a sterge dintr-un arraylist monoamele cu coeficientul 0 pentru a ma ajuta la implementarea metodei de impartire a doua polinoame. Aceasta returneaza un polinom cu polynomial care nu contine monoame cu coef egal cu 0.

1. Clasa Operations: In aceasta clasa sunt implementate metodele care reprezinta operatiile pe care stie sa le faca calculatorul de polinoame.

Metodele clasei sunt:

* addition: primeste ca parametrii doua obiecte de tip Polinom si returneaza un obiect tot de tip Polinom in care este salvat rezultatul adunarii dintre cele doua polinoame date ca parametrii. Pentru realizarea adunarii metoda initializeaza intai un obiect de tip Polinom nou cu numele result dupa care parcurge arraylist-ul primului polinom, initializand de fiecare data un obiect nou de tipul Monom, cu datele monoamelor parcurse din arraylist, pe care il adauga la arraylist-ul polinomului result, dupa care realizeaza parcurgerea si pentru cel de-al doilea polinom.
* subtraction: primeste ca parametrii doua obiecte de tip Polinom si returneaza un obiect tot de tip Polinom in care este salvat rezultatul scaderii dintre cele doua polinoame date ca parametrii. Pentru realizarea scaderii metoda initializeaza intai un obiect de tip Polinom nou cu numele result dupa care parcurge arraylist-ul primului polinom, initializand de fiecare data un obiect nou de tipul Monom, cu datele monoamelor parcurse din arraylist, pe care il adauga la arraylist-ul polinomului result, dupa care realizeaza parcurgerea si pentru cel de-al doilea polinom, insa de aceasta data monoamele create in parcurgearea celui de-al doilea polinom sunt initializate cu coeficientul negat si cu gradul monoamelor parcurse.
* multiplication: primeste ca parametrii doua obiecte de tip Polinom si returneaza un obiect tot de tip Polinom in care este salvat rezultatul inmultirii dintre cele doua polinoame date ca parametrii. Pentru realizarea inmultirii metoda initializeaza intai un obiect de tip Polinom nou cu numele result dupa care parcurge arraylist-ul primului polinom, parcurgere in care este realizata si parcurgerea si celui de-al doilea polinom si initializeaza un monom nou cu gradul egal cu gradele celor doua polinoame adunate si coeficientul egal cu coeficientele celor doua polinoame inmultite, pe care il adauga folosind metoda addMonom in polinomul result.
* division: : primeste ca parametrii doua obiecte de tip Polinom si returneaza un sir de caractere care contine catul si restul impartirii celor doi parametrii. Acest string este de forma: ”cat rest”. Pentru realizarea impartirii metoda creeaza o copie a deimpartitului care va fi modificata pe parcursul acesteia si initializeaza un polinom nou numit result, dupa care verifica conditia ca gradul impartitorului(cel de-al doilea polinom) sa fie mai mic sau egal cu gradul deimpartitului(primul polinom), dupa care se afla un loop de tip while cu conditia ca si copia deimpartitului sa fie diferita de „ ”, adica impartirea sa fi luat sfarsit. Daca conditia este indeplinita atunci este creat un obiect nou de tip Monom care are ca grad diferenta dintre gradul primului monom din arraylist-ul copiei deimpartitului si gradul primului monom din arraylist-ul impartitorului si coeficientul egal cu rezultatul impartirii dintre coeficientele primelor monoame din arraylist-urile copiei deimpartitului si a impartitorului. Acest monom este adaugat cu ajutorul functiei addMonom in arraylist-ul polinomului result si in acelasi timp in arraylist-ul unui polinom p proaspat initializat. La finalul buclei while polinomul ce reprezinta copia deimpartitului primeste diferenta dintre el insusi si produsul dintre impartitor si p, dupa care ii este aplicata metoda delete0 pentru a sterge polinoamele cu coeficientul egal cu 0 din acesta. Catul impartirii este reprezentat de string-ul polinomului result, iar catul este string-ul corespunzator copiei deimpartitului. Metoda trateaza cazul in care numitorul este 0, returnand „Inf” si cazul in care atat deimpartitul cat si impartitorul sunt 0, returnand „Undefined!”.
* derivative: primeste ca parametru un obiect de tipul Polinom si returneaza un alt polinom in care este salvat rezultatul derivatei parametrului. Metoda initializeaza un polinom nou numit result, dupa care parcurge cu foreach monoamele din arraylist-ul polinomului parametru si initializeaza pentru fiecare un monom nou cu gradul egal cu gradul polinomului parametru decrementat si coeficientul echivalent cu produsul dintre coeficientul si gradul polinomului parametru, dupa care il adauga in result.
* integration: primeste ca parametru un obiect de tipul Polinom si returneaza un alt polinom in care este salvat rezultatul integralei nedefinite din parametru. Metoda initializeaza un polinom nou numit result, dupa care parcurge cu foreach monoamele din arraylist-ul polinomului parametru si initializeaza pentru fiecare un monom nou cu gradul egal cu gradul polinomului parametru incrementat si coeficientul echivalent cu rezultatul impartirii dintre coeficientul si gradul incrementat ale polinomului parametru, dupa care il adauga in result.

1. Metodele interfetei:

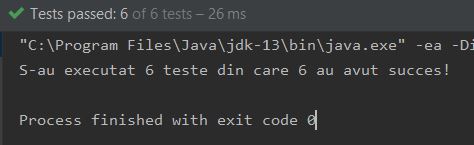
* Clasa ViewCalc: contine trei JPanel-uri de tipul flowlayout si unul de tipul borderlayout, numit content. In primul Jpanel se afla campurile de introducere a datelor si butonul clear si este situat in partea de nord a lui content, al doilea contine butoanele corespunzatoare operatiilor, fiind situat in partea centrala a lui content, iar al treilea contine campurile de rezultat, fiind situat in sudul lui content.

Metode:

* getFirstPolynomial si getSecondPolynomial: returneaza sirul de caractere din primul, respectiv din al doilea polinom;
* setResult1 si setResult2: primesc un parametru de tip String si il afiseaza in campurile de rezultate corerspunzatoare;
* setInput1 si setInput2: seteaza campurile de intrare de date cu un sir de caractere primit ca parametru. Aceasta functie va fi utilizata pentru butonul „clear”;
* showError: afiseaza pe ecran o fereastra cu un mesaj de eroare primit ca parametru;
* addAdditionListener: adauga listener pe butonul pentru adunare;
* addSubtractionListener: adauga listener pe butonul pentru scadere;
* addMultiplicationListener: adauga listener pe butonul pentru inmultire;
* addDivisionListener: adauga listener pe butonul pentru impartire;
* addDerivativeListener: adauga listener pe butonul pentru derivare;
* addIntegrationListener: adauga listener pe butonul pentru integrare;
* addClearListener: adauga listener pe butonul „clear”.
* Clasa ControllerCalc: contone un model\_proj de tipul Operations si un view\_proj de tipul ViewCalc. Clasa are un constructor care seteaza un model\_proj si un view\_proj primite ca parametrii si adauga cate un listener pentru fiecare operatie. Aceasta clasa contine cate o clasa pentru fiecare operatie ce reprezinta listener-ul acesteia, in care sunt initializate trei polinoame, doua in care sunt introduse cu ajutorul metodei toPolinom sirurile de caractere introduse de catre utilizator. Acest proces este realizat cu functiile getFirstPolynomial si getSecondPolynomial care au fost apelate de obiectul view\_proj. Al treilea polinom denumit result reprezinta rezultatul in care este salvat rezultatul returnat de metoda operatiei respective, metoda apelata de obiectul model\_proj. Acest result va fi transformat in sir de caractere cu ajutorul functie toString din clasa Polinom si apoi este dat ca parametru functiei setResult1 care il va afisa in primul camp de rezultat. Depinzand de operatie, listener-ul poate sa contina si un result2 care va fi afisat in al doilea camp de rezultat de catre functia setResult2. Diferit de restul listener-urilor este DivisionListener in care este utilizata functia split(„ ”) pentru a desparti string-ul returnat de metoda division in doua componente: cat si rest, prima fiind parametru pentru setResult1, iar a doua pentru setResult2. Fiecare clasa de listener are un try-catch care returneaza mesajul pentru funcita showError, in caz ca datele de intrare nu sunt corespunzatoare. In clasa ControllerCalc se mai regaseste si ClearListener care utilizeaza functiile de setare a campurilor de intrare si rezultat din view cu parametrul „”.
* Clasa MainClass: initializeaza un model de tip Operations, view de tip ViewCalc si un conroller de tip ControllerCalc noi pentru interfata grafica.

1. Rezultate:

Pentru verificarea corectitudinii operatiilor am utilizat Junit. In acesta am implementat cate un test pentru fiecare operatie, toate reusind.



1. Concluzii:

In concluzie, toate operatiile au fost implementate si testate cu succes.

Prin acest proiect mi-am imbunatatit si reimprospatat cunostintele in domeniul programarii orientate pe obiecte.

1. Bibliografie:
2. Laboratorul din anul II semestrul I de Programare Orientata pe Obiecte(Codul pentru MVC a fost inspirat din aceste laboratoare);
3. <https://google.github.io/styleguide/javaguide.html>;
4. <https://regexr.com/>;
5. <https://stackoverflow.com/>.