MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

CALCULATOR DE POLINOAME

DOCUMENTAȚIE

Tirlea Maria Cristina

Grupa 30225 | An 2 semestrul 2

Cuprins

1. Obiectiv

Scopul acestui proiect este de a implementa funcțiile de bază pe polinoame de o singura variabila cu coeficienti intregi, pentru a reliefa modul în care acestea pot fi create și manipulate prin intermediul paradigmei programării orientate pe obiecte. Printre operațiile de bază se vor număra cele de adunare, scădere, înmulțire și împărțire, integrare si derivare.

Ca oricare alt sistem de calcul, si acesta prezinta date de intrare si date de iesire, generate dupa efectuarea calculelor.

Pe langa partea tehnica, care realizeaza operatiile intre doi operanzi de tip polinom si descrierea polinoamelor, am implementat o interfata grafica intuitiva, user-friendly, care poate fi utilizata usor si confortabil de catre orice utilizator.

Pasii urmati pentru indeplinirea obiectivului:

- Impartirea sarcinilor de implementat intr-un pattern architectural, Model-View-Controller
- Realizarea interfetei prin intermediul careia se colecteaza operanzii si operatiile alese de catre utilizator
- Descompunerea problemei prin definirea claselor necesare reprezentarii polinoamelor (clasele Polynomial si Monomial)
- Transpunerea operanzilor transmisi de catre utlizator sub forma de sir de caractere in structurile de date definite prin clasele Polynomial si Monomial (metoda stringToPoly() din clasa Model)
- Implementarea conversiei polinoamelor din reprezentarea sub forma de lista de monoame in format string, pentru a face posibila verificarea corectitudinii urmatorilor pasi spre rezolvare
- Implementarea operatiilor de adunare,scadere, impartire, inmultire, integrare si derivare a doua polinoame
- Pregatirea rezultatelor pentru a fi prezentate utilizatorului in interfata
- Testarea rezultatelor obtinute

2. Analiza problemei

Cerinte functionale

- Calculatorul polinomial ar trebui să permită utilizatorilor să introducă polinoame

- Calculatorul polinomial trebuie să permită utilizatorilor să selecteze operația matematică de efectuat supra polinoamelor introduse

- Calculatorul polinomial ar trebui să adune, scada, inmultеasca, imparta, deriveze si/sau integreze correct două polinoame

- Calculatorul ar trebui sa returneze rezultatul operatiei, deasemenea sub forma de polinom, catre utilizator, in interfata grafica

- Calculatorul ar trebui sa permita utilizatorului sa readuca interfata la starea sa initiala (reset)

Cazuri de utilizare/ Use-cases

Use-case: adunarea a doua polinoame

Actor principal: utilizator

Scenariul principal de succes:

1. Utilizatorul introduce cele 2 polinoame în interfata grafica
2. Utilizatorul selectează operațiunea de "Add"
3. Calculatorul polinomial efectuează adunarea celor două polinoame și afișează rezultatul
4. Optional, utilizatorul selecteaza butonul “Reset” pentru a sterge input-ul anterior si rezultatul corespunzator, pregatind astfel calculatorul pentru urmatoarele operatii

Secvență alternativă: Polinoame incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)
- Scenariul revine la pasul 1

Use-case: scaderea a doua polinoame

Actor principal: utilizator

Scenariul principal de succes:

1. Utilizatorul introduce cele 2 polinoame în interfata grafica

2. Utilizatorul selectează operațiunea de “Subtract”

3. Calculatorul polinomial efectuează scaderea celor două polinoame și afișează rezultatul

4. Optional, utilizatorul selecteaza butonul “Reset” pentru a sterge input-ul anterior si rezultatul corespunzator, pregatind astfel calculatorul pentru urmatoarele operatii

Secvență alternativă: Polinoame incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)

- Scenariul revine la pasul 1

Use-case: inmultirea a doua polinoame

Actor principal: utilizator

Scenariul principal de succes:

1. Utilizatorul introduce cele 2 polinoame în interfata grafica

2. Utilizatorul selectează operațiunea de "Mutiply"

3. Calculatorul polinomial efectuează inmultirea celor două polinoame și afișează rezultatul

4. Optional, utilizatorul selecteaza butonul “Reset” pentru a sterge input-ul anterior si rezultatul corespunzator, pregatind astfel calculatorul pentru urmatoarele operatii

Secvență alternativă: Polinoame incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)

- Scenariul revine la pasul 1

Use-case: impartirea a doua polinoame

Actor principal: utilizator

Scenariul principal de succes:

1. Utilizatorul introduce cele 2 polinoame în interfata grafica

2. Utilizatorul selectează operațiunea de "Devide"

3. Calculatorul polinomial efectuează impartirea polnomului avand gradul mai mare sau egal, la polinomul avand gradul mai mic, și afișează rezultatul

4. Optional, utilizatorul selecteaza butonul “Reset” pentru a sterge input-ul anterior si rezultatul corespunzator, pregatind astfel calculatorul pentru urmatoarele operatii

Secvență alternativă: Polinoame incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)

- Scenariul revine la pasul 1

Use-case: derivarea a doua polinoame

Actor principal: utilizator

Scenariul principal de succes:

1. Utilizatorul introduce unul sau 2 polinoame în interfata grafica

2. Utilizatorul selectează operațiunea de "Devide"

3. Calculatorul polinomial efectuează derivarea polinomului/ polinoamelor introduse și afișează rezultatul

4. Optional, utilizatorul selecteaza butonul “Reset” pentru a sterge input-ul anterior si rezultatul corespunzator, pregatind astfel calculatorul pentru urmatoarele operatii

Secvență alternativă: Polinoame incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)

- Scenariul revine la pasul 1

Use-case: integrarea a doua polinoame

Actor principal: utilizator

Scenariul principal de succes:

1 Utilizatorul introduce unul sau 2 polinoame în interfata grafica

2. Utilizatorul selectează operațiunea de "Integrate"

3. Calculatorul polinomial efectuează integrarea polinomului/ polinoamelor introduse și afișează rezultatul

4. Optional, utilizatorul selecteaza butonul “Reset” pentru a sterge input-ul anterior si rezultatul corespunzator, pregatind astfel calculatorul pentru urmatoarele operatii

Secvență alternativă: Polinoame incorecte

- Utilizatorul introduce polinoame incorecte (de exemplu, cu 2 sau mai multe variabile)

- Scenariul revine la pasul 1

3. Proiectare

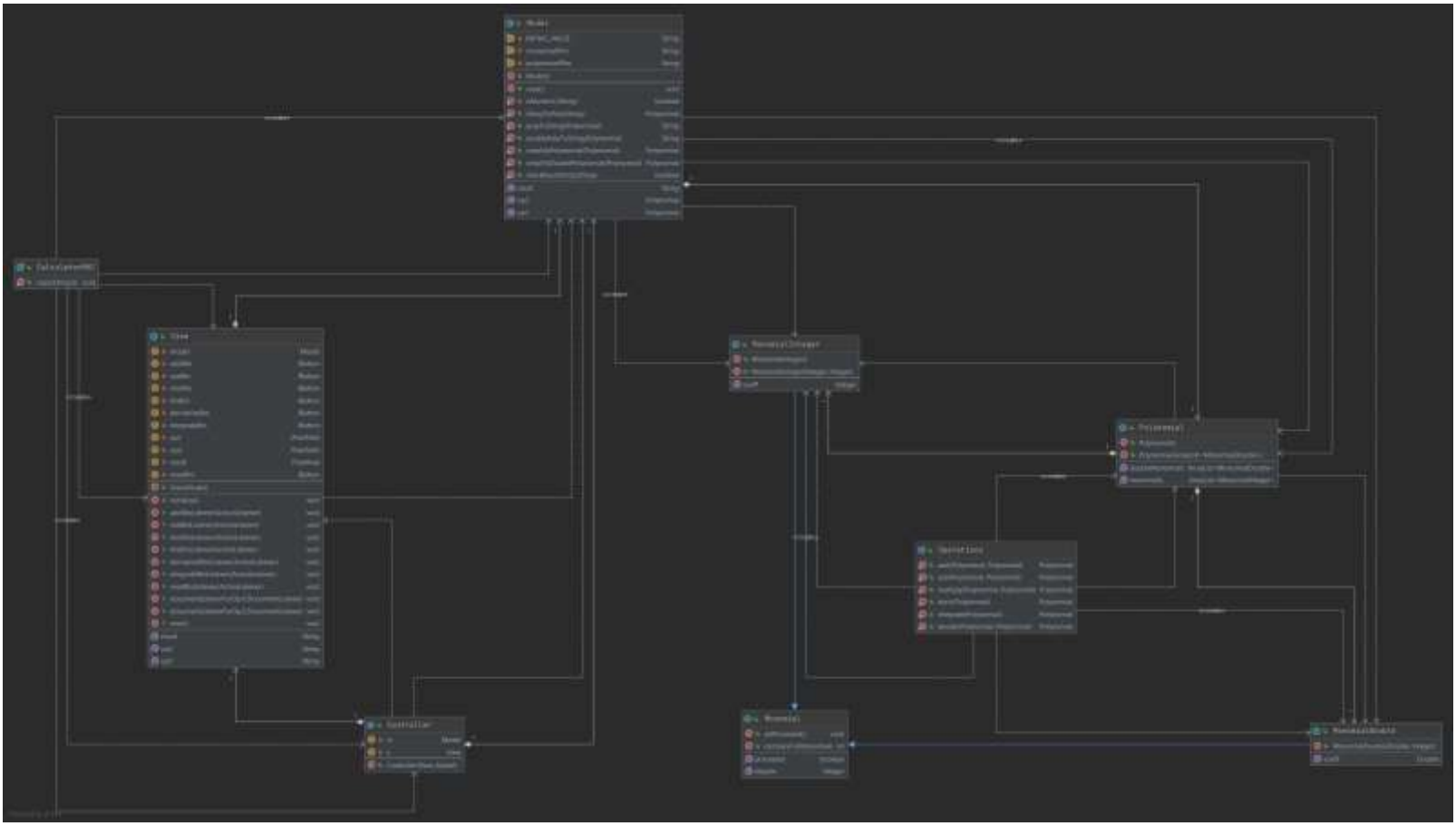
3.1. Diagrama UML

Unified Modeling Language sau UML 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 problemelor in clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat si dincolo de domeniul IT.

Diagrama de clase oferă o notaţie grafică pentru reprezentarea:

- claselor - entităţi ce au caracteristici comune
- relațiilor - relațiile dintre două sau mai multe clase

Diagrama UML de clase desfasurata:



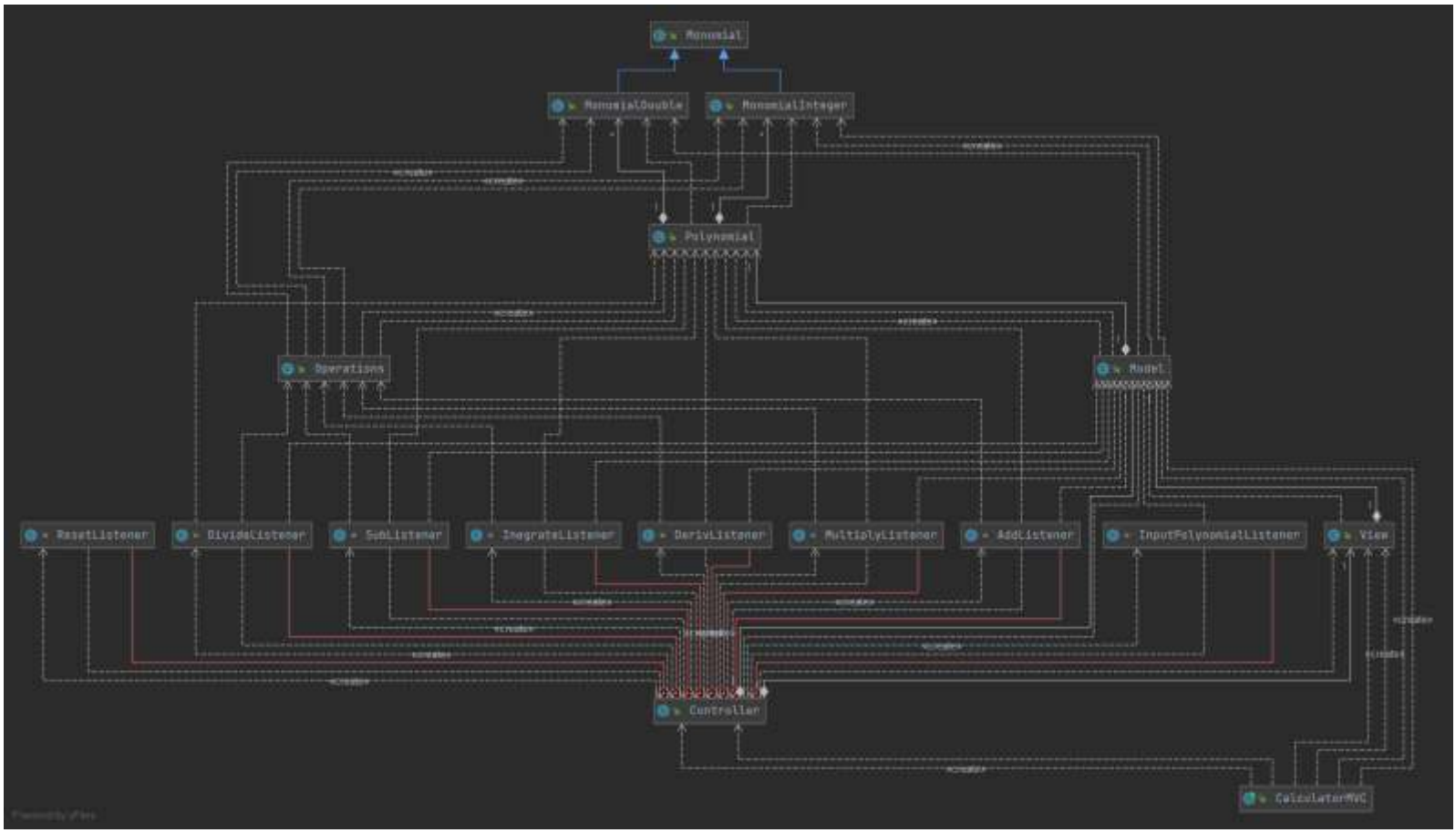


Diagrama UML de clase restransa:

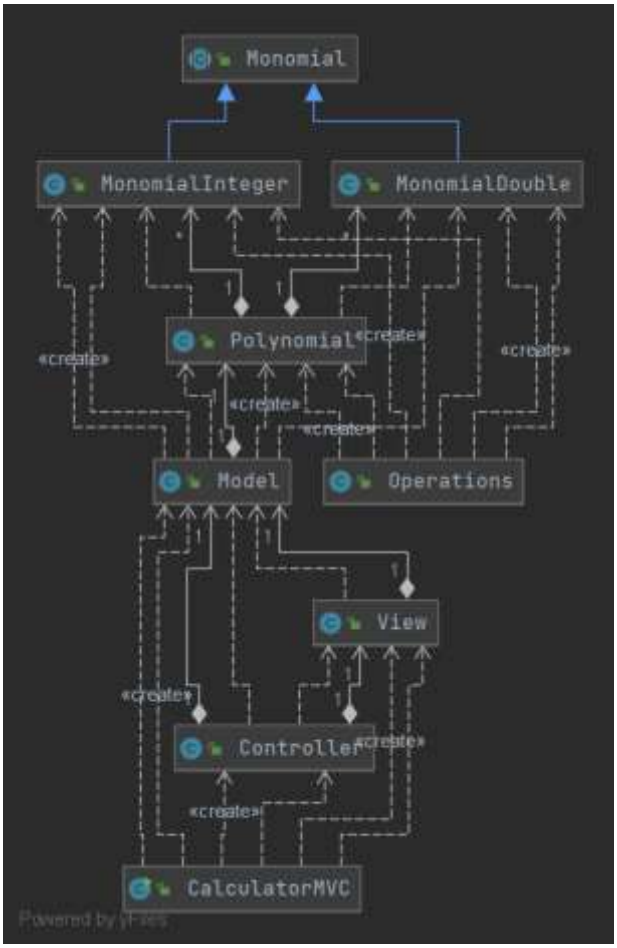
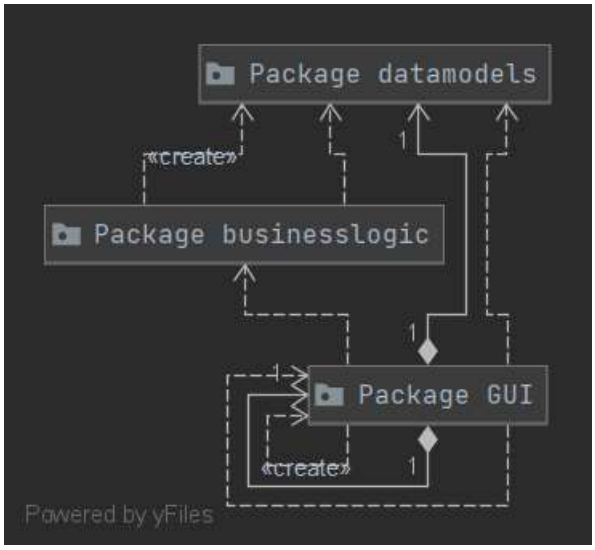


Diagrama UML de pachete:



Proiectarea OOP a aplicatiei:

Calculatorul polinomial este structurat pe baza pattern-ului architectural Model-View-Controller (MVC) divizat in 3 zone functionale: procesarea datelor (model), colectarea datelor de intrare, respectiv prezentarea catre utlizator a datelor de iesire (view) si o parte care se ocupa de comunicarea dintre cele doua componente mentionate anterior. Problema este impartita in 3 pachete:

- Pachetul **datamodels:** cuprinde clasele Polynomial, Monomial, MonomialInteger, MonomialDouble si Model care incapsuleaza datele de baza si functionalitatea calculatorului. Acest pachet reprezinta componenta Model a arhitecturii MVC.
- Pachetul **bussinesslogic:** cuprinde clasa Operations care implementeaza operatiile matematice asupra datelor modelate in clasele din pachetul **datamodels**.
- Pachetul **GUI**: cuprinde clasele care implementeaza interfata grafica. Clasa View contine componente din biblioteca Swing care alcatuiesc interfata si permit atat vizualizarea rezultatelor obtinute de la model, cat si obtinerea datelor de la utilizator. Clasa Controller este asociata vederii, primeste datele introduse de utilizator in interfata ca evenimente care denotă mișcarea mouse-ului, activarea butoanele mouse-ului sau de la tastatura. Evenimentele sunt ulterior traduse în cereri de servicii, care sunt trimise fie la model, fie la vedere.

4. Implementare

4.1. Clase

Pentru rezolvarea acestui task am ales structurarea datelor sub forma de monoame reprezentate prin grad si coeficient, care ulterior sunt utilizate pentru a reprezenta polinoamele sub forma unei liste de monoame.

Clasa abstracta Monomial

- Aceasta clasa defineste aspectele commune pe care le au cele doua clase derivate ale sale (MonomialInteger si MonomialDouble) si reprezinta in esenta un monom avand definit doar gradul, coeficientul fiind specificat in cadrul fiecarei clase derivate
- Clasa Monomial implementeaza interfata Comparable si suprascrie metoda compareTo() din interfata cu scopul de a sorta sirurile de monoame in functie de grad.

Clasa derivata MonomialInteger

- Aceasta clasa extinde clasa Monomial si reprezinta un monom care mosteneste gradul din clasa parinte si specifica un coeficient de tip Integer. Toate operatiile se aplica asupra unor polinoame cu coeficienti intregi.

Clasa derivata MonomialDouble

- Aceasta clasa extinde clasa Monomial si reprezinta un monom care mosteneste gradul din clasa parinte si specifica un coeficient de tip Double. Avem nevoie de aceasta reprezentare a monoamelor pentru rezultatul operatiei de integrare, deoarece integrarea pesupune impartirea coeficientului la gradul monomului incrementat cu 1. Rezultatul acestei impartiri nu este intotdeauna un intreg, astfel coeficientul monomului integrat trebuie reprezentat ca Double.

Clasa Polynomial

- Aceasta clasa reprezinta un polinom sub forma unei liste (ArrayList) de monoame cu coeficienti intregi sau o lista de monoame cu coeficienti double.

Clasa Model

- Clasa Model incapsuleaza metode statice care se ocupa de transformarea polinoamelor reprezentate sub forma de sir de caractere in formatul listei de structure de tip Monomial si vice-versa. De asemenea contine o metoda care simplifica un

polinom si o metoda care verifica daca un sir de caractere respecta formatul unui polinom cu ajutorul unor constant statice de tip regex.

Clasa Operations

- Aceasta clasa implementeaza operatiile matematice (adunare, scadere, inmultire, impartire, integrare,derivare) intre 2 operanzi de tip Polynomial, date modelate in clasele descries anterior.

Clasa View

- Incapsuleaza componentele care alcatuiesc interfata grafica (vederea).

Clasa Controller

- Clasa Controller primeste datele introduse de utilizator in interfata ca evenimente care denotă mișcarea mouse-ului, activarea butoanele mouse-ului sau de la tastatura. Evenimentele sunt ulterior traduse în cereri de servicii, care sunt trimise fie la model, fie la vedere.
- Permite interactiunea intre utilizator si sistemul de calcul.
- Face lagatura intre interfata si functionalitatea oferita de componenta de modelare.

Clasa CalculatorMVC

- Aceasta clasa contine metoda main in care se declara si instantiaza fiecare componenta arhitecturala, astfel facnd posibila rularea intregului program si lansarea interfetei functinale.

4.2. Metode

Metodele clasei Monomial

Aceasta clasa contine metode getters si setters pentru variabilele instanta ‘degree’ (gradul monomului) si ‘processed’ (variabila boolean utilizata la implementarea operatiilor pentru a retine monoamele dintr-un operand care nu au avut corespondent cu acelasi grad in cel de-al doilea operand, deci nu s-a efectuat operatia asupra lui, insa trebuie adaugat la rezultatul final). Monomial extine interfata Comparable si suprascrie metoda compareTo() utlizata la sortarea monoamelor.

Metodele clasei MonomialInteger

Aceasta clasa contine, pe langa metodele mostenite de la clasa Monomial, metode setters si getters pentru coefcientul de tip intreg.

Metodele clasei MonomialDouble

Aceasta clasa contine, pe langa metodele mostenite de la clasa Monomial, metode setters si getters pentru coefcientul de tip real (double).

Metodele clasei Polynomial

Acesta clasa implementeaza metode setters si getters pentru variablilele instanta 'polinom' si polinomDouble' reprezentand liste de monoame cu coeficienti intregi, respective de tip double.

Metodele clasei Model:

1. public static Polynomial stringToPoly(String inp)

– metoda primeste ca parametru un string, ce reprezinta un polinom, pe care il imparte in monoame cu ajutorul unui pattern de regular expression (regex), care la randul lor sunt convertite in variabile de tipul clasei MonomialInteger. Acestea din urma sunt adaugate unei variabile de tip Polynomial, obtinanduse astfel o structura de date de tip Polynomial din string-ul primit la intrare.

2. public static String polyToString(Polynomial pol)

– metoda construieste un polinom reprezentat ca string, pornind de la un polinom reprezentat sub forma unei instante a clasei Polynomial, transmisa ca si parametru. Transformare se face cu ajutorul unui StringBuilder la care se adauga pe rand coeficientul si gradul fiecarui monom din lista de monoame, precum si caractere reprezentand operatorii dintre aceste monoame si caracterul de ridicare la putere '^', avnd in vedere toate cazurile particulare. Pentru aceasta metoda, parametrul pol este un polinom de monoame cu coeficienti intregi.

3. public static String doublePolyToString(Polynomial pol)

- Pentru aceasta metoda, parametrul pol este un polinom de monoame cu coeficienti intregi. Astfel metoda are aceeasi intrebuintare ca si cea prezentata anterior, cu deosebirea ca returneaza un polinom cu ceoficienti de tip double, reprezentat ca string, pornind de la un polinom reprezentat sub forma unei instante a clasei Polynomial, transmisa ca si parametru, deasemenea cu coeficienti double.

4. public static Polynomial simplifyPolynomial(Polynomial p)

-Aceasta metoda statica simplifica un polinom primit ca si parametru: parcurgem polinomul cu doua for-uri, verificand pentru fiecare monom din lista daca in restul listei mai exista monoame cu acelasi grad ca si monomul curent. In caz afirmativ, se aduna

coeficientii celor doua monoame, rezultatul fiind retinut in coeficientul monomului curent, iar monomul gasit avand acelasi grad se elimina din lista.

5. **public static Polynomial simplifyDoublePolynomial(Polynomial p)**

-Simplifica un polinom cu coeficienti reali. Parcurgem polinomul cu doua for-uri, verificand pentru fiecare monom din lista daca in restul listei mai exista monoame cu acelasi grad ca si monomul curent. In caz afirmativ, se aduna coeficientii de tip double ai celor doua monoame, rezultatul fiind retinut in coeficientul monomului curent, iar monomul gasit avand acelasi grad se elimina din lista.

6. **public static boolean checkInputString(String userParsedPolynomial)**

-Verifica daca stringul transmis ca si parametru reprezinta un polinom valid cu ajutorul unui pattern de tip regex : "\\d{0,}(?:x(?:\\^(?:\\d+)?)?)?(?:(?:x?(\\d+)?)[+-]\\d{0,}(?:x(?:\\^(?:\\d+)?)?)?)*$" retinut in constanta statica polynomialPtrn declarata in clasa Model.

7. **private static boolean isNumeric(String str)**

-Compara string-ul primit ca si paramtru cu o regex pentru a stabili daca stringul reprezinta un numar intreg negative sau un numar intreg pozitiv. Returneaza true daca este indeplinita conditia, altfel returneaza false.

8. **public void reset()**

-Reseteaza valorile variabilelor instanta (atributele clasei Model).

- Clasa contine de asemenea metode setter si getter.

Metodele clasei Operations:

1. **Adunare: public static Polynomial add(Polynomial a, Polynomial b)**

-Metoda primeste ca si parametru doi operanzi de tip Polynomial si realizeaza adunarea polinomului a la polinomul b si returneaza rezultatul de tip Polynomial.
- Parcurgem fiecare polinom element cu element, adica fiecare monom din fiecare polinom pentru a cauta monoame cu acelasi grad/ exponent. Odata gasite, se seteaza atributul processed al acelor monoame ca fiind true pentru ambele si continuam prin a efectua adunarea propriu zisa: cream un nou monom cu coeficientul reprezentat de suma celor 2 coeficienti , si exponentul celor doua monoame gasite cu grade egale. Monomul rezultat este adaugat in lista de monoame a polinomului returnat. Ulterior, pentru monoamele care nu au fost

procesate , adica nu au avut pereche ca exponent, sunt introduse si ele in lista de monoame a polinomului rezultat.

2. **Scadere: public static Polynomial sub(Polynomial a, Polynomial b)**

   -Metoda primeste ca si parametru doi operanzi de tip Polynomial si realizeaza scaderea polinomului b din polinomul a si returneaza rezultatul de tip Polynomial.
   - Parcurgem fiecare polinom element cu element, adica fiecare monom din fiecare polinom pentru a cauta monoame cu acelasi grad/ exponent. Odata gasite, se seteaza atributul processed al acelor monoame ca fiind true pentru ambele si continuam prin a efectua scaderea propriu zisa: cream un nou monom cu coeficientul reprezentat de diferenta celor 2 coeficienti , si exponentul celor doua monoame gasite cu grade egale. Monomul rezultat este adaugat in lista de monoame a polinomului returnat. Ulterior, pentru monoamele care nu au fost procesate , adica nu au avut pereche ca exponent, sunt introduse si ele in lista de monoame a polinomului rezultat, insa coeficientii lor sunt inmultiti cu -1.

3. **Inmultire: public static Polynomial multiply(Polynomial a, Polynomial b)**

   -Metoda primeste ca si parametru doi operanzi de tip Polynomial si realizeaza inmultirea polinomului a cu polinomul b si returneaza rezultatul de tip Polynomial.
   - Parcurgem fiecare polinom element cu element, adica fiecare monom din fiecare polinom cream un nou monom cu coeficientul reprezentat de inmultirea celor 2 coeficienti ai perechii de monoame parcurse curent , si exponentul reprezentat de suma celor doi exponenti perechii de monoame parcurse curent. Monomul rezultat este adaugat in lista de monoame a polinomului returnat.

4. **Impartire: public static Polynomial devide(Polynomial a, Polynomial b)**

   -Metoda returneaza rezultatul impartirii polinomului cu grad mai mare dintre a si b la polinom cu grad mai mic dintre cele 2.

   -Pentru implementarea acestei metode am utilizat urmatorul algoritm:
   Pasul 1 - Ordonați monoamele celor două polinoame P și Q în ordine descrescătoare în funcție de gradul lor.
   Pasul 2 - Împarte polinomul cu cel mai înalt grad la celălalt polinom având un grad mai mic (să luăm în considerare că P are cel mai mare grad)
   Pasul 3 - Împarte primul monom al lui P la primul monom al lui Q și obține primul termen al catului
   Pasul 4 - Înmulțiți catul cu Q și scădeți rezultatul înmulțirii din P obținând restul impartirii

Pasul 5 - Repetați procedura de la pasul 2 considerând restul drept noul dividend al diviziunii, până cand gradul restului este mai mic decât gradul lui Q.

5. **Derivare: public static Polynomial deriv(Polynomial a)**

   - Adaugam toate monoamele din polinomul primit ca parametru intr-un alt polinom care va reprezenta rezultatul derivarii la sfarsit. Se parcurge acest polinom, iar pentru fiecare monom din lista acestuia ii schimbam coeficientul ca fiind produsul dintre vechiul coeficient si exponent, iar exponentul devine vechiul exponent – 1.

6. **Intergrare: public static Polynomial integrate(Polynomial a)**

   -Pentru a realiza integrarea am avut nevoie sa definesc clasa MonomialDouble care contine monoame cu coeficienti reali.

   - Adaugam toate monoamele din polinomul primit ca parametru intr-un alt polinom care va reprezenta rezultatul integrarii la sfarsit. Se parcurge acest polinom, iar pentru fiecare monom din lista acestuia ii schimbam coeficientul ca fiind rezultatul impartirii dintre coeficient si exponentul incrementat cu 1, iar exponentul devine vechiul exponent + 1.

4.3. GUI

- Interfata implementata in clasa View contine elemente din pachetul java.Swing si cod pentru layout-ul acestor elemente :

- doua JTextField-uri in care utilizatorul introduce cei doi operanzi de tip polinom (in clasa Controller am implementat un DocumentListener care verifica daca utilizatorul respecta formatul de polinom al inputului iar in caz contrar este atentionat printr-un mesaj de eroare).

- JButtons pentru fiecare operatie si unul pentru resetarea campurilor cu tesxt din interfata

- un JTextField pentru afisarea rezultatelor

- JLabels pentru a indica utilizatorului scopul fiecarui element din interfata, astfel mentinand interfata cat mai intuitiva

Comunicarea intre interfata/ utiliator si codul de funcionalitate din spate este realizata cu ajutorul clasei Controller si al claselor interne definite in clasa Controller care implementeaza interfete precum ActionListener si DocumentListener, care asculta evenimente asupra interfetei precum mișcarea mouse-ului, activarea butoanele mouse-ului sau activarea tastelor de la tastatura. Evenimentele sunt ulterior traduse în cereri de servicii, care sunt trimise fie la model-ul de date, fie la vedere.

5. Testare

Scenariile pentru testarea operatiilor au fost implementate cu framework-ul Junit5.

Am creat clasa de test **OperationsTest.java** in care am importat pachetele **businesslogic** - pentru a avea acces la operatii si **datamodels** – pentru a avea acces la structurile de date definite.

Clasa de test cuprinde 6 metode de test, cate una pentru fiecare operatie, in care am declarat operanzii necesari fiecarei operatii. Deasemenea am declarat o variabila Polynomial pentru retinerea rezultatului asteptat. Apoi folosindu-ma de functia **assertEquals()** am comparat dimensiunea rezultatului asteptat, contra dimensiunii rezultatului obtinut in urma apelarii functiei care efectueaza operatia, dupa care am comparta element cu element lista de monoame a rezultatului asteptat cu lista de monoame a rezultatului propriu-zis.

Toate cele 6 teste implementate ruleaza cu succes.

6. Concluzii

In concluzie, consider ca acest proiect mi-a aprofundat cunostiintele de programare in limbajul Java si de implementare a paradigmelor OOP. Problema a cuprins toate etapele importante ale dezvoltarii unei aplicatii cu interfata grafica respectand standardul OOP, fapt care m-a ajutat sa imi reamintesc aspectele predate la cursul de Programare Orientata pe Obiecte de semestrul trecut si mi-a oferit totodata oportunitatea de a crea de la 0 o aplicatie functionala si intuitiva intr-un timp relativ scurt.

7. Bibliografie

- FUNDAMENTAL PROGRAMMING TECHNIQUES ASSIGNMENT 1 – SUPPORT PRESENTATION
- Regex tutorial — A quick cheatsheet by examples | by Jonny Fox | Factory Mind | Medium
- Clase abstracte și interfețe [Programare Orientată pe Obiecte] (pub.ro)
- Cursuri POO
- Assert Two Lists for Equality Ignoring Order in Java | Baeldung