DOCUMENTATIE

TEMA 1

CALCULATOR DE POLINOAME

NUME STUDENT: HĂDĂRĂU IOANA
GRUPA:30223

CUPRINS

1. Obiectivul temei

Această temă are ca obiectiv principal crearea și implementarea unui calculator de polinoame cu o interfață grafică prin care utilizatorul poate insera polinoame și poate selecta operația matematică care trebuie efectuată vizualizând rezultatul sub forma unui polinom.
Obiectivele secundare care trebuie atinse pe parcursul rezolvării temei sunt:

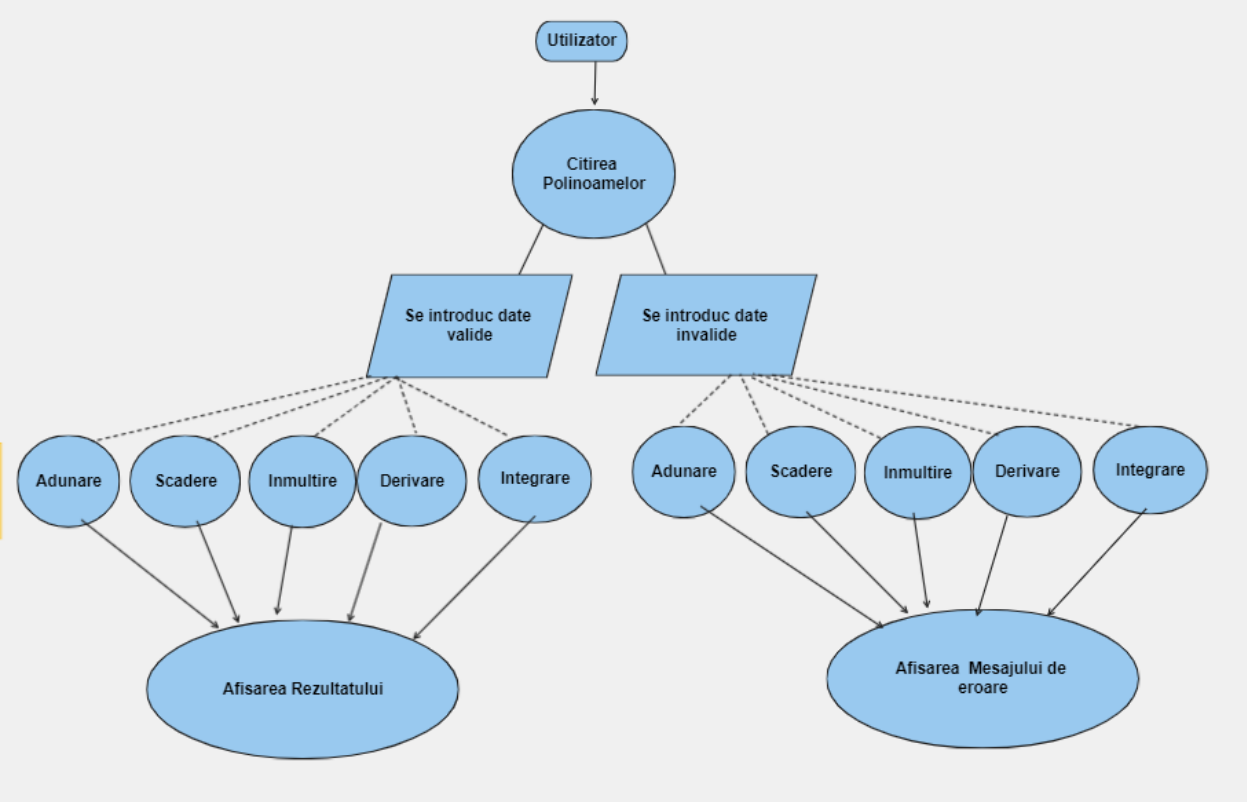
- Proiectarea și implementarea clasei Polinom
- Proiectarea și implementarea Interfetei Grafice de Utilizator (GUI)
- Proiectarea și implementarea operațiilor de citire si afisare a unui polinom
- Proiectare clasei de Operații și implementarea metodelor de adunare, scădere, înmulțire, derivare și integrare

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

Cerințele funcționale:
- Calculatorul de polinoame ar trebui să permită utilizatorilor să introducă două polinoame
- Calculatorul de polinoame ar trebui să avertizeze utilizatorul în cazul unei sintaxe greșite
- Calculatorul de polinoame ar trebui să permită utilizatorilor să selecteze
operația matematică dorită

Cerințele non funcționale
- Calculatorul de polinoame ar trebui să ofere un rezultat corect, în timp util
-Calculatorul de polinoame ar trebui să fie intuitiv de folosit
- Calculatorul de polinoame ar trebui să fie adaptabil noilor cerințe

Cazurile de utilizare sunt: Adunarea, Scăderea, Înmulțirea, Derivare și Integrarea Polinoamelor
Nota: Coeficienții primiți de la utilizatori trebuie sa fie întregi.



Pașii de urmat pentru un scenariu de succes

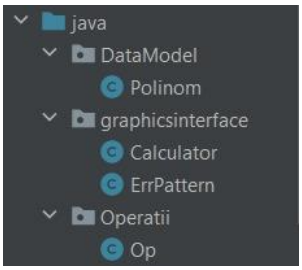
1. Utilizatorul trebuie să introducă cele două polinoame in JTextField-urile destinate, conform sintaxei de scriere a unui polinom.
2. Utilizatorul trebuie să apese butonul destinat operației pe care dorește să o facă asupra polinoamelor. În cazul operatiei de Derivare și Integrare, primul polinom este cel ales implicit pentru aplicarea acestora.
3. Polinomul rezultat se afișează ordonat în cel de al treilea JTextField din interfața grafică.
4. După afișarea unui rezultat, utilizatorul poate selecta altă operație fără a mai introduce alte polinoame.

Scenarii alternative

1. Utilizatorul introduce doar un polinom, lăsând gol celălalt câmp.
   Programul va genera o fereastră de eroare care informează utilizatorul că nu se respectă formatul, oferind posibilitatea reluării programului.
   Nici o operatie dintre cele specificate mai sus nu se vor efectua până când ambele câmpuri nu sunt completate corect, după formatul impus.
2. Utilizatorul introduce caractere în afara celor acceptate în format ([^x0123456789+-\^]).
   Programul va genera o fereastră de eroare care informează utilizatorul că nu se respectă formatul dorit, oferind posibilitatea reluării programului.
   Nici o operatie dintre cele specificate mai sus nu se vor efectua până când ambele câmpuri nu sunt completate corect, după formatul impus.

3. Proiectare

Imaginea următoare surprinde pachetele proiectului.

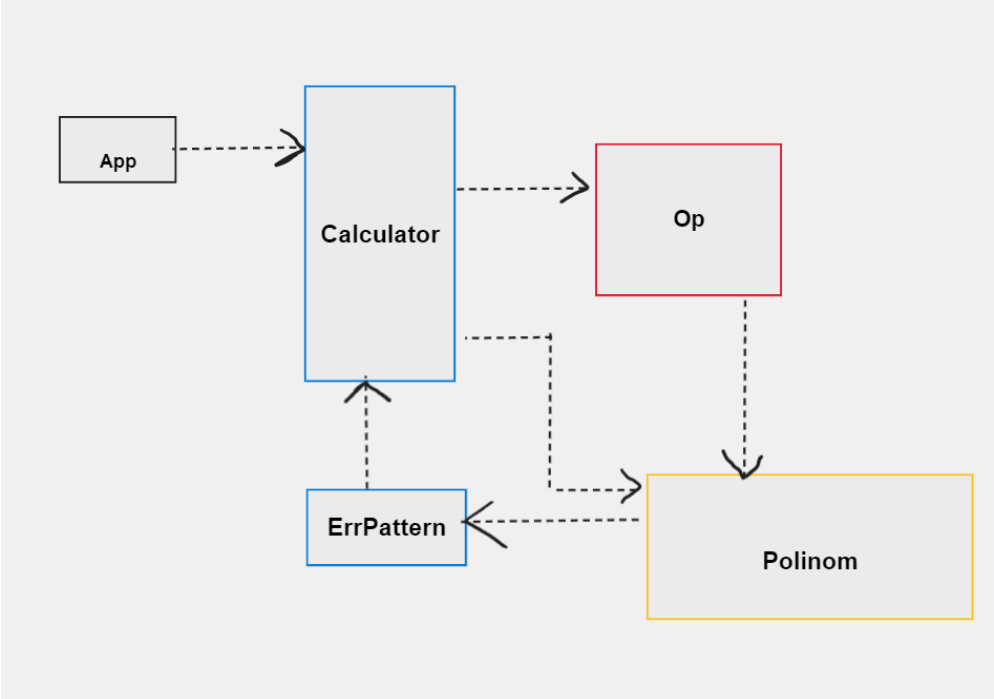


După cum se poate observa, am declarat 3 pachete, având fiecare un rol clar, în functie de modul de funcționare a claselor pe care le cuprind.

Pachetul “Operații” contine clasa „Op”care se ocupă de implementarea metodelor de adunare, scădere, înmulțire, derivare, integrare și afișare.

Pachetul „graphicsinterface” contine clasa „Calculator”, care include descrierea ferestrei și funcționalitatea butoanelor , și clasa „ErrPattern” care cuprinde descrierea ferestrei de eroare în cazul întâmpinării acesteia, iar Pachetul „DataModel” conține clasa „Polinom” reprezentativp tipului de obiect pe care dorim sa îl manipulam pe tot parcursul temei.

În figura de mai gos este reprezentată diagrama claselor. Astfel, se pot observa relațiile de dependență dintre acestea. Mai exact, clasa App crează un obiect de tipul Calculator, care la rândul lui creează alte obiecte de tipul Polinom si Op. În ceea ce privește clasa Op, clasă destinată implementării operțiilor, aceasta folosește obiecte de tipul Polinom, iar clasa Polinom, unde se produce manipularea polinoamelor introduse de utilizator, instanțează un obiect de tipul ErrPattern, in cazul descoperirii unei sintaxe greșite. Nu în ultimul rând, clasa ErrPattern folosește un obiect de tipul Calculator, astfel îi oferă utilizatorului șansa de a relua procesul de calcul.



4. Implementare

Pachetul DataModel conține

Clasa Polinom are doi constructori Polinom() si Polinom(String), având ca variabilă instanță un obiect HashMap<Integer,Float> care reține prechea (gradul,coeficientul). Gradul polinomului este un număr întreg, în timp ce coeficientul este un număr real.

- Constructorul Polinom()- creează un obiect de tipul polinom gol.
- Constructorul Polinom(String) – creează un polinom după string-ul primit ca parametru. Prin parsarea cu Regex a acestuia am manipulat string-ul ca perechi de monoame. Am folosit un pattern specific monoamelor, iar pe baza unor verificări pe stringurile generate de matcher salvez perechiile (key,value) corespunzatoare. Aici se disting 4 cazuri speciale de monoame pe lânga cel deja menționat, acestea fiind: “x “care se interpretează ”1x^1” , ”-x” care se interpretează “1x^1”, ”2x” care se interpretează” 2x^1” și ”1 “care se interpretează “ x^0”.

Pachetul “Operatii”

Clasa “Op” coține un constructor Op(Polinom, Polinom) și implementează metodele:

- String Adunare( )- care abordeză problema prin crearea unui nou obiect HashMap pe care îl inițializez cu HashMap-ul primului polinom. Apoi, parcurg hashmap-ul celui de al doilea polinom verficând condiția de egalitate între gradele celor două monoame. În cazul egalității adun coeficienții, iar altfel avansez.

  Variabila “flag” îndeplinește rolul unui semafor, care se “aprinde” (ia valoarea 1) dacă s-a efectuat operația de adunare, adică s-a găsit un grad “= “, altfel rămâne stins (valoarea 0) ceea ce indică faptul că trebuie să introduc în HashMapu-ul rez perechea (grad, coeficient) din polinomul al diolea.

  La final, apelez funcția „Afisare” care primește ca parametru HashMap-ul rezultat și returnează stringul corespunzător.

- String Scadere( )- care abordeză problema prin crearea unui nou obiect HashMap pe care îl inițializez cu HashMap-ul primului polinom. Apoi, parcurg hashmap-ul celui de al doilea polinom

verficând condiția de egalitate între gradele celor două monoame. În cazul egalității scad coeficienții, iar altfel avansez.

Variabila “flag” îndeplinește rolul unui semafor, care se “aprinde” (ia valoarea 1) dacă s-a efectuat operația de scadere, adică s-a găsit un grad “= “, altfel rămâne stins (valoarea 0) ceea ce indică faptul că trebuie să introduc în HashMapu-ul rez perechea (grad, coeficient) din polinomul al diolea.

La final, apelez funcția „Afisare” care primește ca parametru HashMap-ul rezultat și returnează stringul corespunzător.

- String Inmultire( )- care abordeză problema prin crearea unui nou obiect HashMap, destinat reținerii rezultatului. Apoi aplic înmulțirea polinoamelor prin adunarea gradelor și înmulțirea coeficienților, verifcând dacă gradul rezultat se mai găsește în map pentru a însuma coeficienții.

  La final, apelez funcția „Afisare” care primește ca parametru HashMap-ul rezultat și returnează stringul corespunzător.

- String Derivare( )- care abordeză problema prin crearea unui nou obiect HashMap destinat reținerii rezultatului. Apoi, parcurg hashmap-ul polinomului formând noul grad (noul grad= grad_polinom-1) si noul_coeficient(noul coeficient= coeficient_polinom* grad_polinom),

  La final, apelez funcția „Afisare” care primește ca parametru HashMap-ul rezultat și returnează stringul corespunzător.

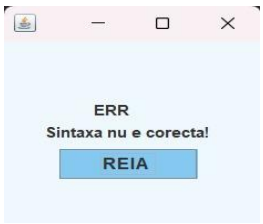
- String Integrare( )- abordeză problema prin crearea unui nou obiect HashMap destinat reținerii rezultatului. Apoi, parcurg hashmap-ul polinomului formând noul grad (noul grad= grad_polinom+1) si noul_coeficient(noul coeficient= coeficient_polinom / grad_polinom).
  Deoarece rezultatul împărțirii poate avea un număr nedefinit de zecimale, prin intermediul pattern-ului \\d+\\.\\d{2} extrag rezultatul cu exact primele 2 zecimale.

  La final, apelez funcția „Afisare” care primește ca parametru HashMap-ul rezultat și returnează stringul corespunzător.

-String Afisare(HashMap)- construiește un string prin utilizarea unui obiect de tipul StringBuilder și prin manipularea acestuia cu metoda sa specifică: append. Astfel, pe baza multiplelor verificări pe valorile de cheie si valoare, a HashMap-ului, reușesc să construiesc stringul conform formatului unui polinom.

Din punct de vedere estetic, ambele clase din Pachetul graphicsinterface folosesc elemente ale bibliotecii javax.swing.*

-Clasa Calculator implementează interfața ActionListener pentru funcționalitatea celor cinci butoane vizibile în interfață. Fiecare buton fiind sugestiv destinat unei operații.
Constructorul Calculator() surprinde detaliile ferestrei și permite ajustarea acestora. Prin definirea constantelor xs, dist, h, y, am simplificat urmărirea coordonatelor fiecărei componente.



-Clasa ErrPattern implementează interfața ActionListener pentru funcționalitatea butonului REIA, care oferă posibilitate utilizatorului să reintroducă date valide pentru rularea programului. Fereastra care se deschide în urma instanțierii unui obicet de tipul ErrPattern conține două mesaje sugestive și un singur buton.
Constructorul ErrPattern() surprinde detaliile ferestrei și permite ajustarea acestora.

5. Rezultate

Aplicația este testată folosind JUnit, dar și prin testarea manuală.

Operațiile de adunare, scădere, înmulțire, derivare și integrare au fost testate prin introducerea unor polinoame uzuale și a unor polinoame care acoperă cazurile speciale, dar și pentru polinoame care nu se muleaza pe formatul cerut.

Cazurile de testare au fost:

-Polinoame uzuale, in care semnele altereaza

-Polinoame care contin monoame “speciale”(“x”,”-x^2”,”4”)

-Polinoame care nu îndeplinesc sintaxa dorită

6. Concluzii

Prin rezolvarea acestei teme am învățat cum să abordez metoda de parsare Regex și cum se lucrează eficient cu un obiect de tipul HashMap, beneficiind de avantajele perechii (key, value). Am descoprit utlitatea obiectului StringBuilder.

Mai mult, mi-am amintit elementele esențiale de interfață grafică de utilizator(GUI), punându-mi la contributie simțul estetic.

7. Bibliografie

https://www.jetbrains.com/help

https://www.baeldung.com/junit-failure-vs-error

https://www.javatpoint.com/StringBuilder-class

https://www.geeksforgeeks.org/stringbuilder-class-in-java-with-examples/