DOCUMENTAȚIE

TEMA NUMĂRUL_1

CALCULATOR DE POLINOAME

NUME STUDENT: ARDELEAN RALUCA-NICOLETA

GRUPA: 30227

Cuprins

1. Obiectivul temei

Obiectivul principal

Obiectivul principal al acestei teme este crearea și implementarea unui calculator de polinoame cu o interfață grafică interactivă prin intermediul căreia utilizatorul poate insera două polinoame iar mai apoi poate alege una din cele șase operații disponibile. În urma alegerii operației, un rezultat va fi afișat.

Obiectivele secundare

Pentru îndeplinirea obiectivului principal au fost urmați următorii pași:

- Analiza problemei, modelarea, scenarii, cazuri de utilizare
- Proiectarea calculatorului de polinoame
- Implementarea propriu-zisa a calculatorului de polinoame
- Testarea calculatorului de polinoame

Aceste aspecte vor fi detaliate în capitolele următoare.

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

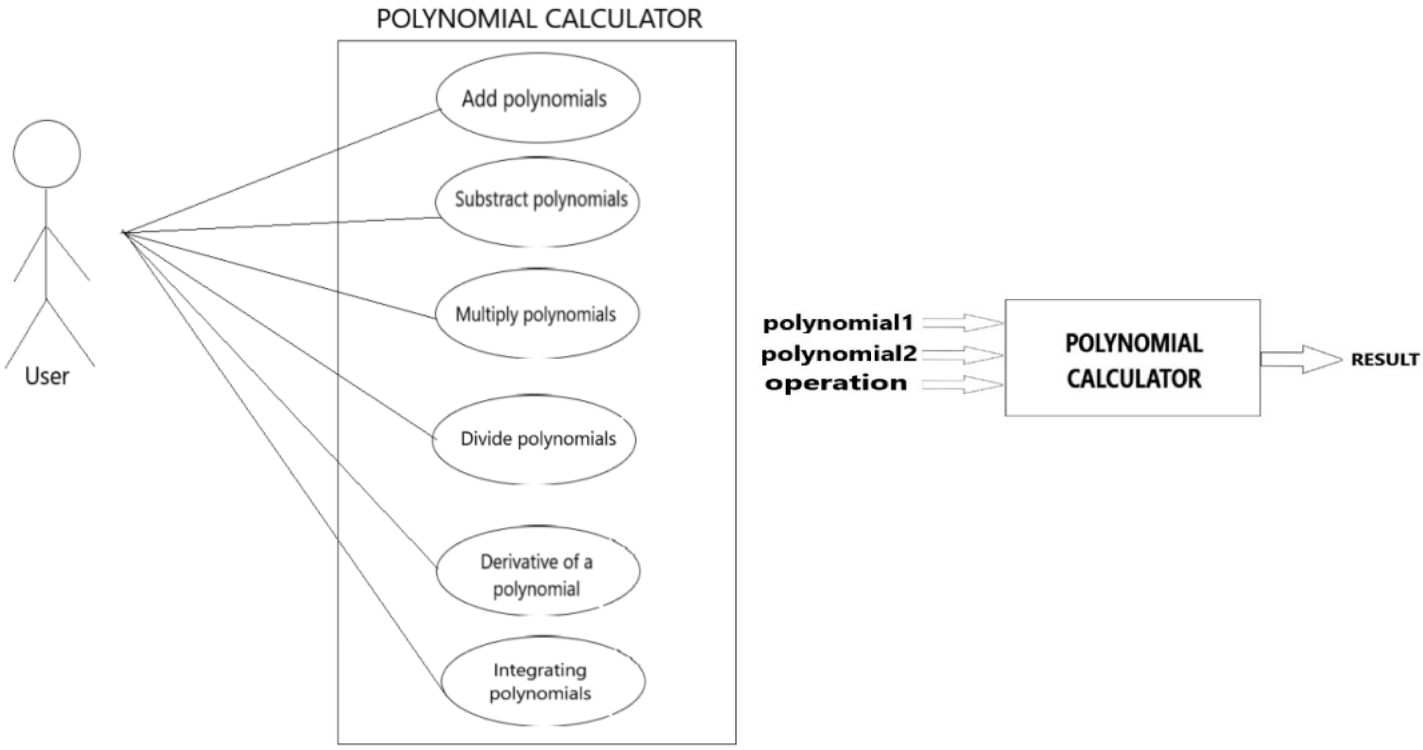
Cerințe funcționale

- Calculatorul de polinoame trebuie să permită utilizatorului să introducă cele două polinoame.
- Calculatorul de polinoame trebuie să permită utilizatorului să selecteze operația dorită din cele șase puse la dispoziție.
- Calculatorul de polinoame trebuie să poată aduna doua polinoame.
- Calculatorul de polinoame trebuie să poată scădea doua polinoame.
- Calculatorul de polinoame trebuie să poată înmulți doua polinoame.
- Calculatorul de polinoame trebuie să poată împărți doua polinoame.
- Calculatorul de polinoame trebuie să poată calcula derivata unui polinom.
- Calculatorul de polinoame trebuie să poată calcula integrala unui polinom.

Cerințe non-funcționale

- Calculatorul de polinoame trebuie să fie intuitiv și ușor de utilizat.
- Calculatorul de polinoame trebuie să aibă o interfață grafică interactivă care să faciliteze utilizarea lui.

- Cazuri de utilizare



Cazuri de utilizare: Adunarea, Scăderea, Înmulțirea, Împărțirea a doua polinoame

1. Utilizatorul introducere doua polinoame în interfața de utilizator. Dacă se întâmplă ca utilizatorul să introducă doar unul, celălalt va fi in mod automat polinomul nul (P=0).

2. Utilizatorul selectează una din operațiile Adună/ Scade/ Înmulțește/ Împarte.

3. Odată apăsat unul din butoanele de operații, calculatorul de polinoame realizează operația cerută și afișează rezultatul. Rezultatul va fi un nou polinom, excepție făcând împărțirea, unde rezultatul este sub forma a două polinoame: cât și rest.

Cazuri de utilizare: Derivarea, Integrarea unui polinom

1. Utilizatorul poate introduce unul sau două polinoame pentru a calcula derivata, respectiva integrala.

2. Utilizatorul selectează una din operațiile Derivează/ Integrează.

3. Odată apăsat unul din butoanele de operații, calculatorul de polinoame realizează operația cerută și afișează rezultatul. Dacă se introduce doar un polinom în unul din cele 2 câmpuri disponibile, operația se realizează doar asupra lui și se ignoră celălalt câmp liber, iar rezultatul va fi un polinom. În schimb, dacă se introduce polinom in fiecare câmp, se realizează operația aleasa asupra fiecăruia, iar rezultatul va fi sub forma a doua polinoame.

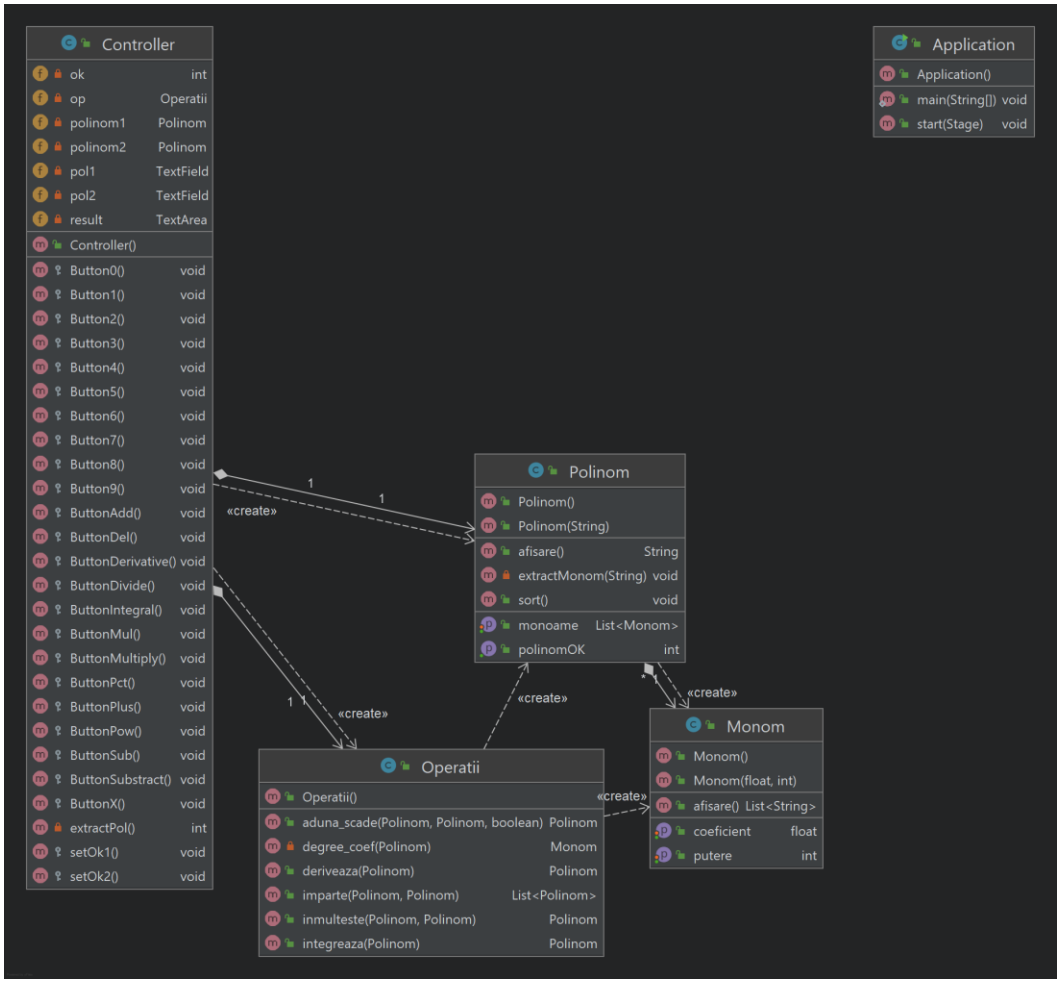
Scenariu alternativ: Polinoame invalide

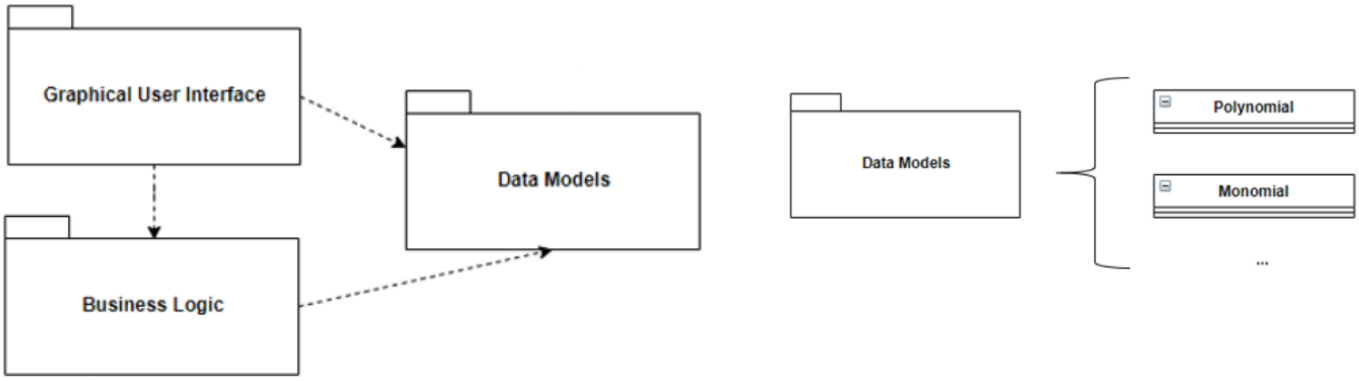
Pentru toate cele șase cazuri de utilizare, dacă ceea ce introducere utilizatorul nu reprezintă polinoame valide, operațiile nu vor putea fi realizare, iar atunci când utilizatorul apasă unul din butoanele de operații va fi afișat un mesaj și utilizatorul va trebui să reintroducă polinoamele.

Pentru ca un polinom să fie valid, acesta trebuie sa fie doar de variabila “X” și să nu conțină spatii libere.

3. Proiectarea

Diagrama UML generată direct din IntelliJ:





Graphical User Interface conține clasele care fac posibilă realizarea unei interfețe utilizator interactive.(clasele Controller și Application)

Business Logic conține clasele prin care se implementează operațiile matematice asupra polinoamelor.(clasa Operatii)

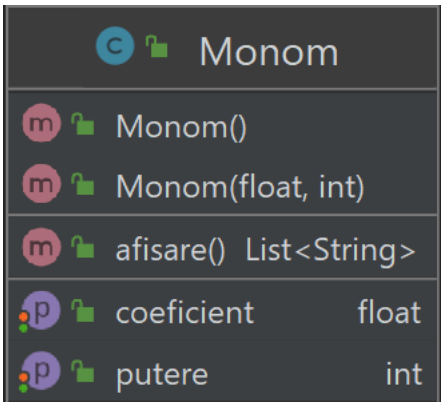
Data Models conține clasele care gestionează datele referitoare la polinoame, ținându-se cont de faptul că orice polinom este alcătuit dintr-o serie de monoame. (clasele Monom și Polinom). Am utilizat astfel ca structură de date un ArrayList de monoame, reprezentând polinomul. Alegând această structură pentru a defini polinomul am avut acces la metode precum “add()”, “remove()”, “get()”, ușurându-mi astfel munca.

Așadar, aplicația este formată din două părți principale: partea de control și partea de vizualizare.

Partea de control este mai complexă și cuprinde întreaga funcționalitate internă a calculatorului de polinoame, în timp ce partea de vizualizare este reprezentată de interfața utilizator care folosește partea de control pentru a manipula butoanele și restul elementelor existente în interfață.

4. Implementarea

- Clasa “Monom”

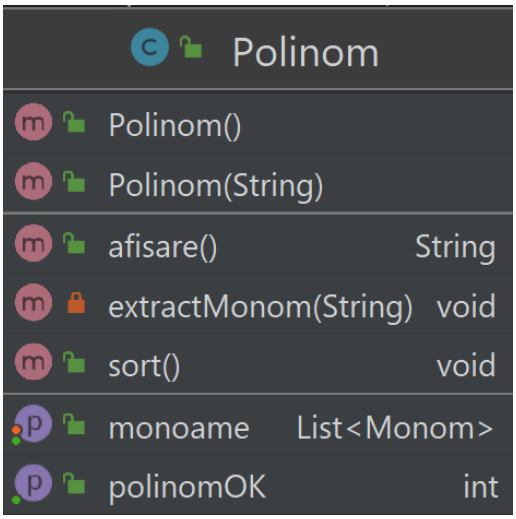


Clasa “Monom” are doua proprietăți: un coeficient de tip real(float) și o putere de tip întreg. (coeficient * X ^ putere)

Obiectele, instanțe ale clasei “Monom” pot fi create în două moduri prin intermediul celor doi constructori: unul fără parametrii și unul cu parametrii. Folosind constructorul fără parametrii se va crea monomul nul (M = 0 ^ 0 = 0) și folosind constructorul cu parametrii se va crea monomul cu puterea și coeficientul primite ca parametri: M = paramCoeficient * X ^ paramPutere.

De asemenea, clasa conține și o metodă utilă pentru a afișa monomul sub formă matematică: coeficient X ^ putere, în cazul în care coeficientul si puterea sunt diferite de 0 și 1. Sunt tratate și restul cazurilor în care puterea și/sau coeficientul sunt 0 sau 1 (X, coeficient * X, X ^ putere, respectiv 0). Metoda returnează o listă de două elemente, în care primul element este o variabilă de control care indică dacă este un monom nul sau nu, iar al doilea element este monomul sub formă de string în reprezentare matematică. Variabila de control va fi utilă în clasa “Polinom” pentru afișarea polinomului nul sub forma P=0.

- Clasa “Polinom”



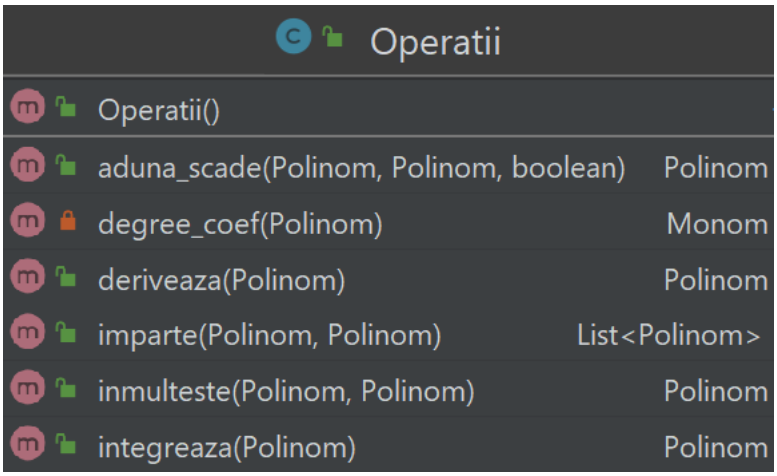
Clasa “Polinom” are două proprietăți: o listă de monoame și o variabilă de control pentru a verifica dacă polinomul este valid.

Obiectele, instanțe ale clasei “Polinom” pot fi create în două moduri prin intermediul celor doi constructori: unul fără parametrii și unul cu parametrii. Folosind constructorul fără parametrii se va crea monomul nul (P = 0 ^ 0 = 0) care este de fapt alcătuit dintr-o listă de monoame ce conține un singur monon nul, iar folosind constructorul cu parametrii se va crea polinomul corespunzător stringului dat ca parametru, adică se vor extrage monoamele din acel string și se vor adăuga în lista de monoame a polinomului. În final, se vor sorta monoamele din listă în ordinea descrescătoare a puterilor pentru efectuarea eficientă a operațiilor ulterioare.

Clasa conține o metodă privată care realizează extragerea monoamelor dintr-un string și le adaugă în lista de monoame a polinomului. Acest lucru se realizează cu ajutorul unei pattern.

La fel ca în clasa “Monom” și clasa polinom are o metoda de afișare sub formă matematică a polinomului. Aceasta ia fiecare monom din lista de monoame a polinomului și îl afișează, existând în plus afișări de operatori “+”, “-“ între monoame.

- Clasa “Operații”



Clasa “Operații” conține doar metode care reprezintă de fapt operațiile ce se pot realiza asupra polinoamelor care sunt transmise ca parametrii.

Metoda “aduna_scade” primește două polinoame ca parametrii și o variabilă de tip boolean care va indica dacă metoda realizează adunarea polinoamelor sau scăderea lor. Dacă variabila este “true” se realizează adunarea, iar dacă variabila este “false” se realizează scăderea. Aceasta returneaza un nou polinom în care este reținut rezultatul adunării/scăderii.

Metoda “degree_coef” primeste ca parametru un polinom și returneaza monomul din lista de monoame a polinomului cu cea mai mare putere care are coeficientul diferit de 0. Această metodă este utilă în metoda “imparte” și din acest motiv aceasta nu trebuie să fie vizibilă în afara clasei, modificatorul de acces “private” fiind cel mai potrivit.

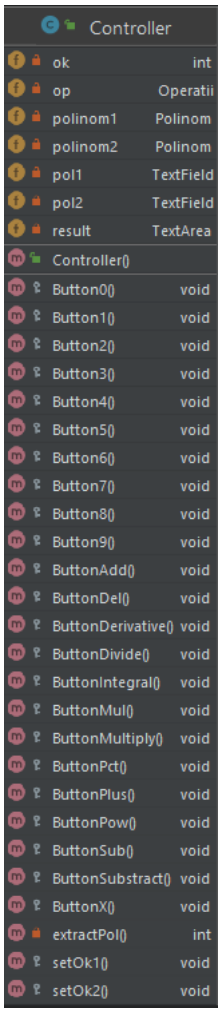
Metoda “derivează” primește ca parametru un polinom și returnează un alt polinom care reprezintă polinomul derivat primit ca parametru.

Metoda “imparte” primește ca parametrii două polinoame și realizează împărțirea primului polinom primit ca parametru la al doilea. Această metodă returnează o listă de polinoame cu două elemente: primul polinom din listă reprezintă câtul împărțirii, iar cel de-al doilea reprezintă restul.

Metoda “inmulteste” primește și ea două polinoame ca parametrii și returnează rezultatul înmulțirii sub forma unui nou polinom. Metoda ia fiecare element din primul polinom primit ca parametru și îl înmulțește cu fiecare element din cel de-al doilea polinom rezultând câte un polinom care se aduna cu polinomul rezultat, care inițial este polinomul nul, dar se actualizează mai apoi la fiecare pas.

Metoda “integrează” primește ca parametru un polinom și returnează un alt polinom care reprezintă polinomul integrat primit ca parametru.

- # Clasa “Controller”



Clasa “Controller” preia practic comenzile utilizatorului asupra câmpurilor text și asupra butoanelor din interfața grafică și le execută.

TextFiled-urile “pol1” și “pol2” sunt câmpurile unde se scriu polinoamele sub formă de string-uri ca mai apoi să fie preluate pentru a se transforma în polinom1 și polinom2 care sunt de tip Polinom cu ajutorul metodei “extractPol”. În cazul în care datele introduse nu reprezintă polinoame valide se va afișa un mesaj corespunzător.

Variabila de control „ok” dictează în care din cele două TextField-uri va fi scrisă informația în urma apăsării butoanelor destinate scrierii polinoamelor. Această variabilă este setată prin intermediul metodelor “setOk1” și “setOk2” la apăsarea pe căsuțele de text, semnalizând că acolo se dorește scrierea. “setOk1” setează ok=0, adică se scrie în căsuța destinată primului polinom, iar . “setOk2” setează ok=1, adică se scrie în căsuța destinată celui de-al doilea polinom.

TextField-ul “result” reține rezultatul obținut în urma executării operațiilor dorite. Acesta va fi afișat pe interfața utilizator la apăsarea unui buton destinat executării unei operații.

Prin intermediul metodelor “Button0”…”Button9” se scrie în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” cifra corespunzătoare apăsării butonului.

Prin apăsarea butonului de punct este apelată metoda “ButtonPct” ce scrie în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” caracterul punct(“.”) util pentru a introduce coeficienți care reprezintă numere reale.

Prin apăsarea butonului de delete este apelată metoda “ButtonDel” care sterge din câmpurile de TextField, fie din “pol1”, fie din “pol2” în funcție de variabila “ok”, ultimul caracter introdus.

Prin apăsarea butonului de înmulțire a numerelor este apelată metoda “ButtonMul” care execută scrierea în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” caracterul steluță(“*”) util pentru a semnaliza o înmulțire cu X.

Prin apăsarea butonului de plus este apelată metoda “ButtonPlus” care scrie în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” caracterul plus(“+”) util pentru a semnaliza o adunare între două monoame de grad diferit.

Prin apăsarea butonului de minus este apelată metoda “ButtonSub” ce scrie în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” caracterul minus(“-”) util pentru a semnaliza o scădere între două monoame de grad diferit.

Prin apăsarea butonului de putere este invocată metoda “ButtonPow” prin intermediul căreia se scrie în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” caracterul căciuliță(“^”) util pentru a semnaliza o ridicare la putere a lui X.

Prin apăsarea butonului de X este chemată metoda “ButtonX” ce scrie în câmpurile de TextField, fie în “pol1”, fie în “pol2” în funcție de variabila “ok” caracterul X(“X”) util pentru a indica variabila polinomului.

Prin apăsarea butonului destinat executării operației de adunare se apelează metoda “ButtonAdd” care realizează adunarea celor două polinoame introduse și afișează rezultatul în căsuța de text cu id-ul “result”.

Prin apăsarea butonului destinat executării operației de scădere se apelează metoda “ButtonSubstract” ce realizează scăderea celor două polinoame introduse și afișează rezultatul în căsuța de text cu id-ul “result”.

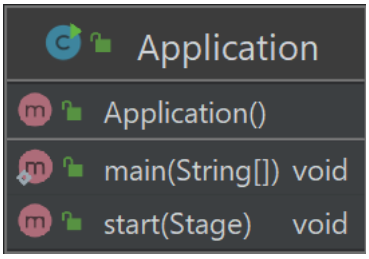
Prin apăsarea butonului destinat executării operației de înmulțire este apelată metoda “ButtonMul” prin intermediul căreia se realizează înmulțirea celor două polinoame introduse și afișearea rezultatul în căsuța de text cu id-ul “result”.

Prin apăsarea butonului destinat executării operației de împărțire este invocată metoda “ButtonDivide” care realizează împărțirii celor două polinoame introduse și afișează rezultatul în căsuța de text cu id-ul “result”.

Prin apăsarea butonului destinat executării operației de derivare se apelează metoda “ButtonDerivative” ce realizează derivarea fie a unui polinom dacă doar unul a fost introdus(în câmpul primului sau celui de-al doilea polinom), fie a ambelor polinoame introduse dacă s-au introdus date în ambele câmpuri și în final se afișează rezultatul în căsuța de text cu id-ul “result”.

Prin apăsarea butonului destinat executării operației de integrare se cheamă metoda “ButtonIntegral” care realizează integrarea fie a unui polinom dacă doar unul a fost introdus(în câmpul primului sau celui de-al doilea polinom), fie a ambelor polinoame introduse dacă s-au introdus date în ambele câmpuri și în final se afișează rezultatul în căsuța de text cu id-ul “result”.

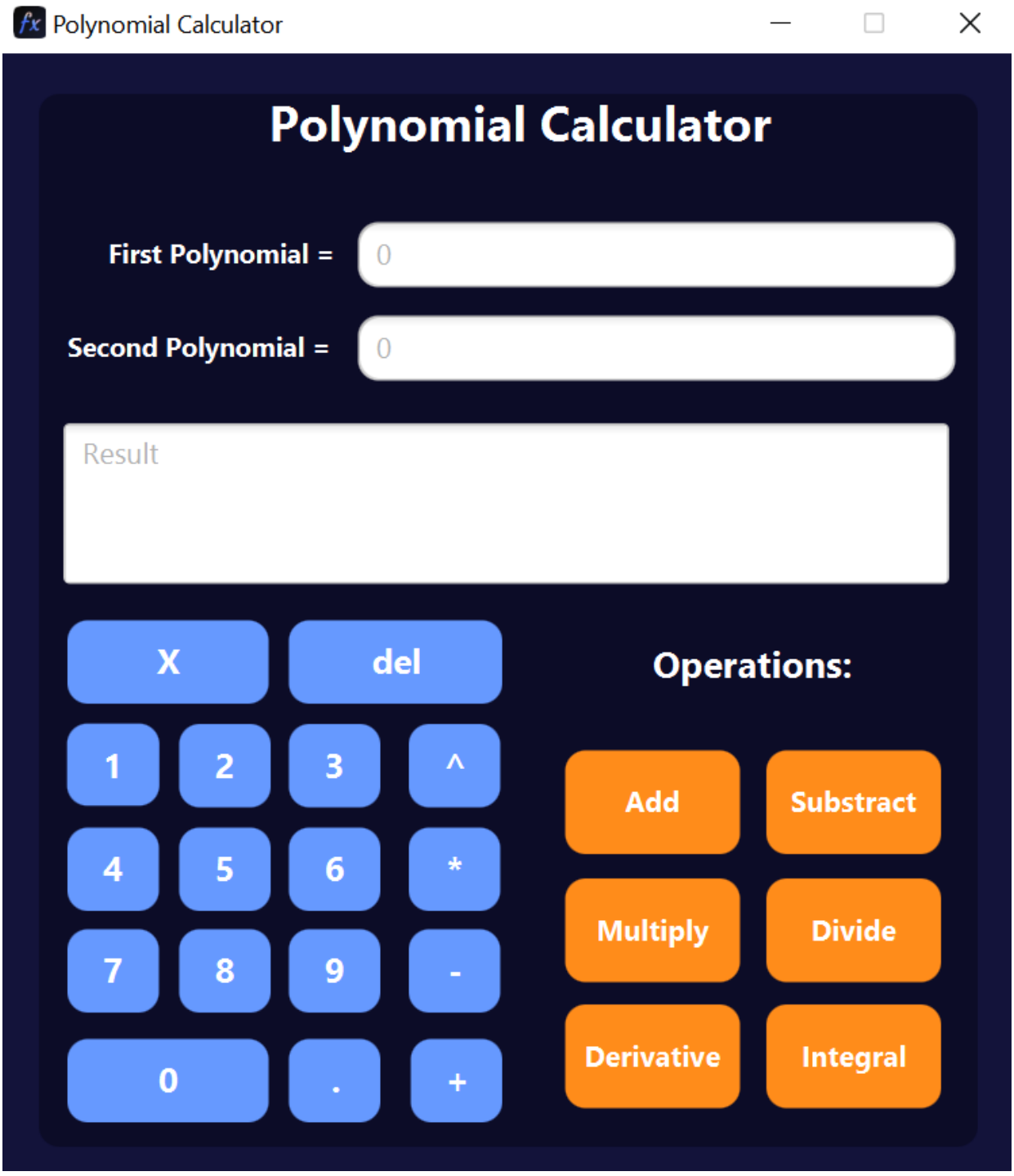
- # Clasa “Application”



Clasa “Application” conține metoda “start” care creează fereastra aplicației, setându-i o iconiță și un titlu și adaugă prin intermediul unui fisier “.fxml” toate butoanele descrise în clasa “Controller”, această clasă fiind astfel legată și controla de clasa “Controller”.
Prin metoda “main” se execută aplicația.

- ## Implementarea interfeței utilizator

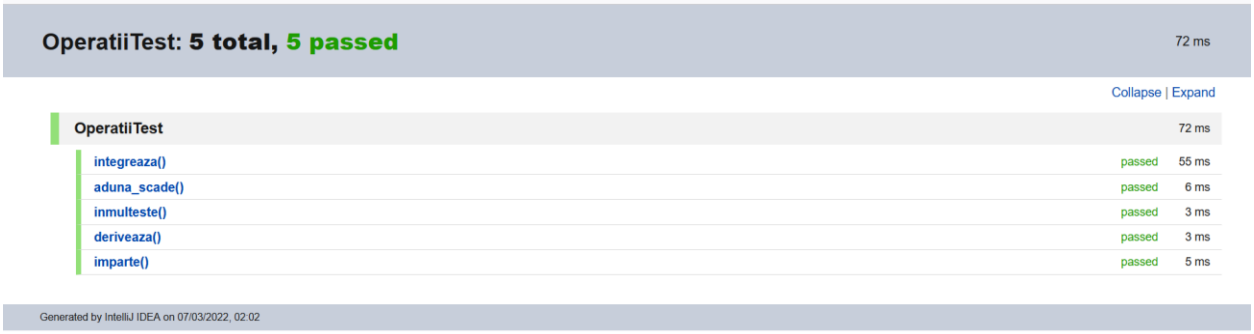
  Interfața utilizator am creat-o utilizând JavaFx, întrucât oferă o gamă largă de opțiuni de personalizare a elementelor cum ar fi butoane, TextFields, TextAreas, etc. Mai mult decât atât este foarte ușor de lucrat cu JavaFx întrucât se bazează pe Drag and Drop și oferă și o previzualizare a produsului final.
  Atât clasa "Controller" cât și clasa "Application" importa biblioteci din javafx, astfel că practic interfața este dată de conectarea și comunicarea celor doua clase.



5. Rezultate

Pentru testarea calculatorului de polinoame am realizat o testare a clasei “Operații” folosind JUNIT.

În clasa de testare am luat fiecare metodă din “Operații”, care reprezintă de fapt câte o operație asupra polinoamelor și am verificat dacă rezultatul returnat de aceste metode coincide cu rezultatul calculat de mine. Am realizat câte 3,4 teste pentru fiecare metodă încercând să tratez toate cazurile posibile. Pentru acest lucru am folosit metodele assertAll și assertEquals din clasa Assertions din JUNIT.



6. Concluzii

Realizând această primă temă mi-am sedimentat mai bine cunoștințele legate de conceptele programării orientate pe obiect ceea ce îmi va fi foarte de folos pentru următoarele proiecte. De asemenea, am învățat să lucrez cu JavaFX, respectiv cu SceneBuilder-ul asociat acestuia și tocmai din acest motiv am ales pentru interfață JavaFX și nu Java Swing, deoarece cu Java Swing știam deja să lucrez.

Prin intermediul acestei teme am descoperit cât de important e în procesul de dezvoltare al unui program testarea metodelor, deoarece prin utilizarea metodelor de testare ne putem verifica funcționalitatea codului scris, făcând astfel mult mai eficientă detectarea și corectarea posibilelor erori.

- Posibile dezvoltări ulterioare
  - adăugarea unor operații noi cum ar fi: extragerea rădăcinilor polinomului, calculul derivatei de un anumit ordin, calculul unor integrale definite, calculul unor limite din polinom1/polinom2, calculul transformatei Laplace, etc.
  - îmbunătățiri la nivel de interfață grafică.

7. Bibliografie

1. Stack Overflow

2. YouTube

3. JUnit 5 | IntelliJ IDEA (jetbrains.com)