



— UNIVERSITATEA TEHNICĂ —
DIN CLUJ-NAPOCA

*Facultatea de Automatică și
Calculatoare*

*Secția Calculatoare și Tehnologia
Informației*

3/11/2021

TEMA 1
CALCULATOR DE POLINOAME
DOCUMENTAȚIE

Cuprins

1. Obiectivul temei	2
2. Analiza problemei, modelare, scenarii, cazuri de utilizare.....	2
3. Proiectare.....	3
4. Implementare.....	3
• Clasa CalculatorMVC	3
• Clasa View.....	4
• Clasa Controller	4
• Clasa Operatii	4
• Clasa Polinom	5
• Clasa Monom.....	6
• Clasa MonomInt	6
• Clasa MonomDou	6
• Interfața utilizator	6
5. Rezultate.....	6
6. Concluzii	7
7. Bibliografie.....	7

1. Obiectivul temei

Obiectivul acestei prime teme este reprezentat de realizarea unui calculator de polinoame care are ca scop îndeplinirea operațiilor de bază pe unul, respectiv două polinoame. Operațiile pe un singur polinom (derivarea și integrarea) țin cont doar de primul polinom introdus de utilizator, integrarea oferind în plus și apariția constantei C la rezultat. În schimb, operațiile pe două polinoame (adunare, scădere, înmulțirea, împărțirea) țin cont de ambele polinoame introduse de utilizator, împărțirea venind în plus cu completarea, în cazul în care există, a restului în spațiul alocat special pentru acesta.

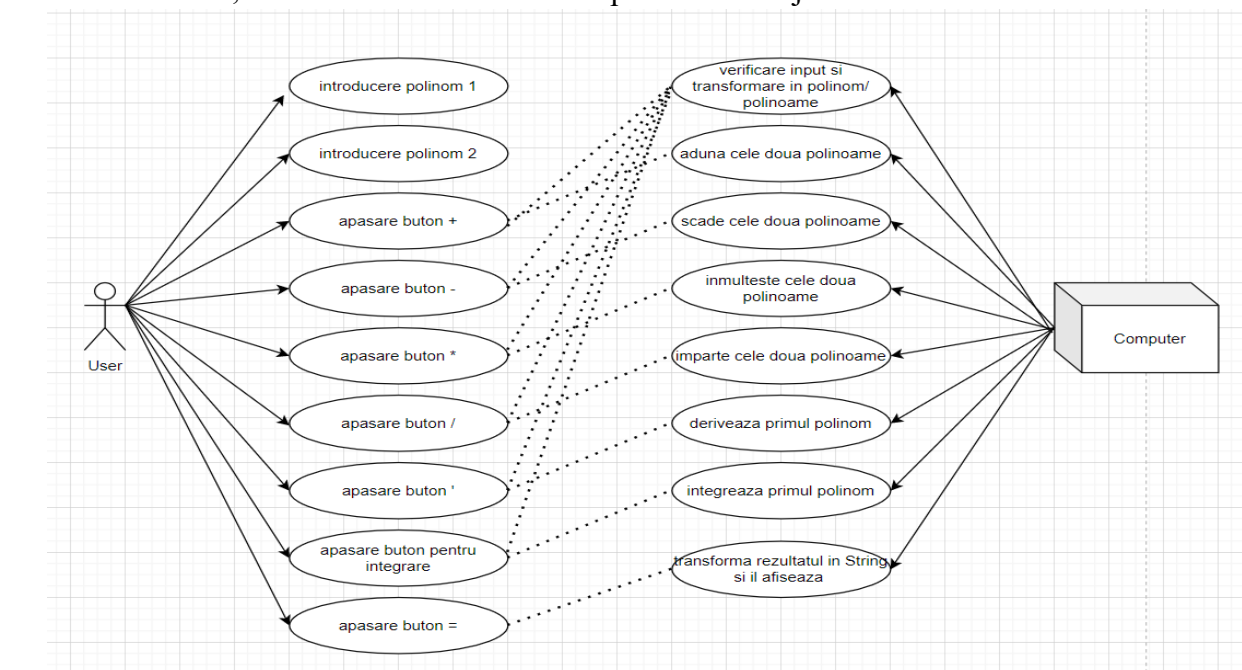
Modul de citire a datelor introduse se face simplu, urmând ca textul introdus să fie convertit conform unei rezolvări în concordanță cu cerințele temei, iar rezultatul se va afișa la fel de simplu, printr-o convertire a datelor rezultate în conformitate cu o afișare inteligibilă oricărui utilizator.

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

Analiza problemei constă, inițial, în înțelegerea cerinței problemei și implicit implementarea unui plan inițial pentru rezolvarea sa care constă, după cum am menționat mai sus, în operațiile de bază ale unui calculator de polinoame, ținând cont de anumite constrângeri ale acestora conform teoriilor matematice învățate. Ca prim pas, vom alege clasele necesare (substantive) și metodele reprezentative pentru fiecare (verbe) pentru a putea realiza o aplicație funcțională și ușor de înțeles pentru orice cunoscător de java.

O altă etapă reprezentativă constă în stabilirea intrărilor și ieșirilor aplicației noastre. În cazul de față avem două intrări reprezentative pentru operațiile de adunare, scădere, înmulțire și împărțire, acestea fiind două String-uri ce constituie cele două polinoame pentru care se vor efectua operațiile specificate mai sus. În cazul derivării și integrării ne vom folosi doar de primul String menționat mai sus. În schimb, pentru ieșire vom avea un String pentru aproape toate cazurile, mai puțin pentru împărțire unde mai avem încă o ieșire, tot String, aceasta reprezentând restul operației efectuate.

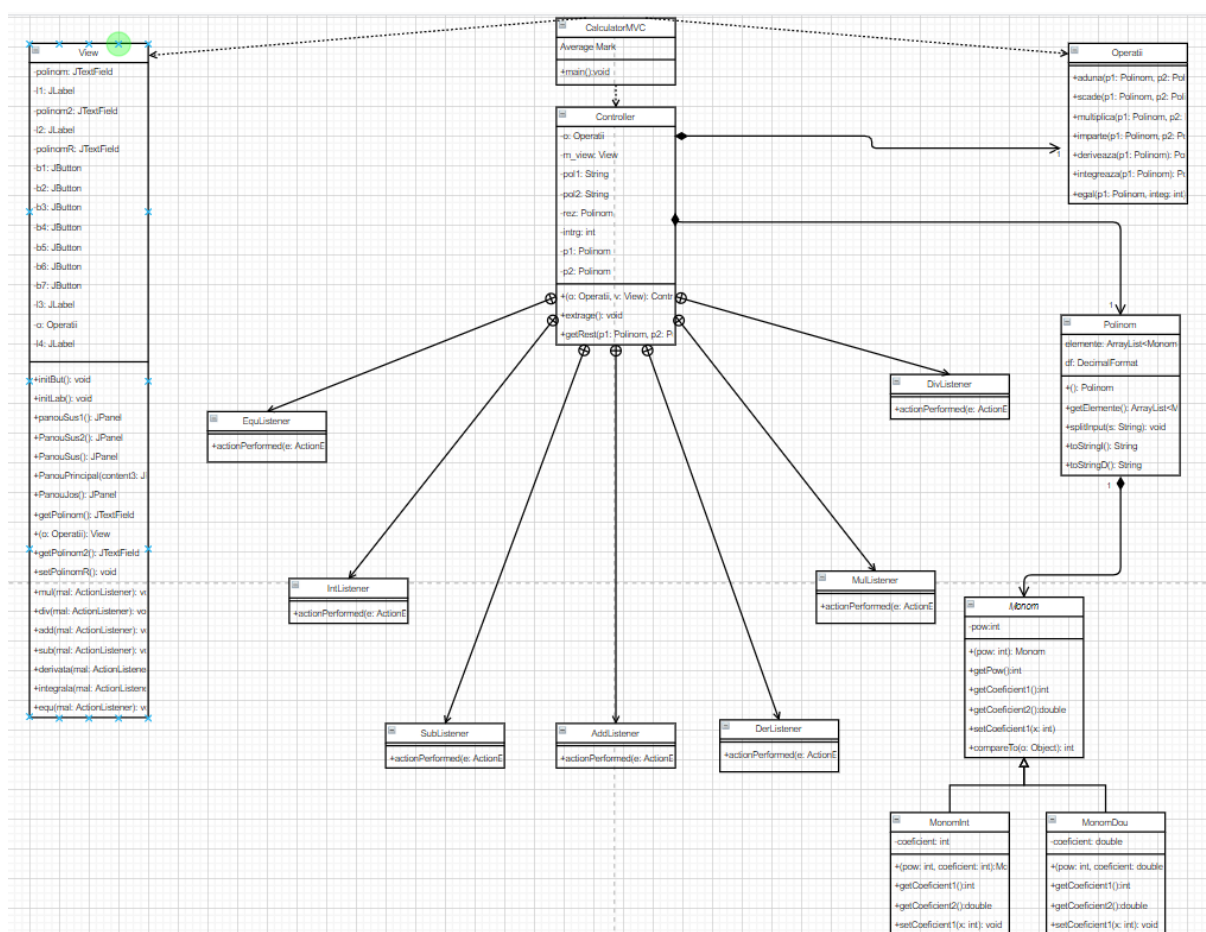
Am luat în calcul eventualele erori de introducere a unor date care nu sunt în conformitate cu intrările dorite, iar ca urmare utilizatorul va primi un mesaj de eroare în acest caz.



3. Proiectare

Pentru reprezentarea temei am ales un model MVC și am folosit 5 pachete: Model (conține operațiile propriu-zise), View (conține interfața), Controller (reprezintă legătura dintre Model și View), Teste (pentru testele în Junit) și Calculator (pentru clasa de baza cu metoda main de unde se apelează proiectul).

Diagrama UML este reprezentată mai jos, iar în ea putem observa clasele proiectului cu legăturile specifice (moștenire, legătură pentru clase interne...), dar și metodele și structurile de date folosite.



4. Implementare

Voi începe descrierea claselor referindu-mă la diagrama UML de clase de mai sus, într-o manieră Top-Down.

- Clasa CalculatorMVC

Este clasa de bază a proiectului, aceasta conținând metoda statică main unde sunt apelați constructorii claselor Operații, View și Controller.

- Clasa View

Extinde JFrame, aparține pachetului cu același nume și reprezintă clasa pentru interfața programului. Constructorul acesteia apelează metodele pentru inițializarea fiecărei componente a interfeței cu utilizatorul, de la butoane, la label-uri și până la panel-uri. În principal aceste metode instanțiază panel-urile și le stabilește un anumit pattern. Pe lângă aceste metode mai avem și metode ca add, sub, mul, div care au rolul de a lega operațiile propriu-zise de interfața programului pentru a putea realiza operațiile dorite în momentul interacțiunii unui utilizator cu interfața calculatorului.

- Clasa Controller

În constructorul acestei clase ne vom folosi de metodele specificate mai sus din clasa View și de clasele interne ale clasei Controller pentru a realiza legătura dintre aceste trei pachete Model, View și Controller și implicit, pentru a respecta modelul MVC specificat în cerințele proiectului.

Metoda extrage realizează un apel al metodei splitInput din clasa polinom pentru a realiza trecerea de la un String oarecare introdus de utilizator în câmpurile speciale pentru introducerea polinoamelor la o reprezentare în polinom ce conține șiruri de monoame. Această metodă este folosită de fiecare dată când avem nevoie să prelucrăm polinoamele introduse de utilizator.

Metoda getRest va returna restul operației de împărțire în funcție de două cazuri:

- a) Primul polinom are gradul mai mic decât al doilea
-în această situație restul va fi chiar primul polinom introdus, iar rezultatul va fi 0.
- b) Cazul contrar primului caz
-în acest caz, am scăzut din primul polinom produsul celui de al doilea polinom cu câtul împărțirii (teorema împărțirii cu rest).

Dupa stabilirea acestor două cazuri, am eliminat monoamele ce au coeficientul 0.

Clasele interne AddListener, SubListener, DerListener, IntListener, MulListener și DivListener implementează ActionListener și, după cum am menționat mai sus, vor apela metodele operațiilor corespunzătoare (adună. Scade, derivează, integrează, multiplică și împarte) după ce s-a realizat convertirea de la String la Polinom prin intermediul metodei extrage.

Clasa internă EquListener implementează ActionListener și dacă aceasta este instanțiată după realizarea unei împărțiri se va modifica și câmpul alocat restului, altfel se va modifica doar câmpul alocat rezultatului în conformitate cu rezultatul obținut din realizarea operației precedente apăsării butonului =.

- Clasa Operatii

Această clasă, după cum indică și numele său, conține metodele reprezentative pentru fiecare operație pe polinom plus metoda egal pentru momentul apăsării butonului =.

Metoda aduna are ca parametrii două polinoame și va returna rezultatul operației de adunare a acestora, rezultat care va fi tot un polinom. Principiul acestei metode constă în parcurgerea ambelor polinoame și adunarea coeficienților corespunzători acelorași puteri, ținând cont ca acești coeficienți să nu se reducă, iar dacă se reduc nu se vor adăuga la polinomul corespunzător rezultatului. După terminarea monoamelor unui polinom se vor parcurge restul elementelor rămase și se vor adăuga, pe rând la rezultat.

Metoda scade are ca parametrii două polinoame și va returna rezultatul operației de scădere a acestora, rezultatul fiind tot un polinom. Principiul acestei metode este acela de a parcurge, pe rând, ținând cont de putere, monoamelor ambelor polinoame și scăderea celor cu puteri

egale. Dacă coeficienții se reduc nu se va adăuga acel monom la polinomul rezultat. După terminarea monoamelor unui polinom se vor parcurge monoamele celui rămas, iar dacă acesta e primul monoamele rămase se vor adăuga la rezultat fără nicio schimbare, dar dacă este al doilea, monoamele rămase se vor adăuga la rezultat cu semn opus.

Metoda `multiply` are ca parametru două polinoame și va returna rezultatul operației de înmulțire a acestora tot ca un polinom. Această operație are ca principiu de funcționare parcurgerea monoamelor ambelor polinoame, înmulțirea coeficienților și adunarea puterilor, iar dacă polinomul corespunzător rezultatului are elemente se va ține cont și de acestea: dacă avem monoame cu puterea egală cu noua putere obținută, elementul nu se va adăuga la lista de monoame a polinomului rezultat, ci i se va adăuga la coeficientul corespunzător puterii menționate noul coeficient obținut.

Metoda `derive` are ca parametru un polinom și va realiza operația de derivare, după cum indică și numele, urmând să returneze rezultatul sub forma unui polinom. Principiul de funcționare este parcurgerea monoamelor polinomului dat ca parametru, înmulțirea puterii cu coeficientul pentru a obține un nou coeficient și decrementarea puterii, obținându-se, astfel, un nou monom ce se va adăuga la rezultat.

Metoda `integrate` are ca parametru un polinom, realizează integrarea acestuia și va returna rezultatul sub forma unui polinom de monoame cu coeficienți reali. Principiul de funcționare se bazează pe parcurgerea monoamelor polinomului și, folosindu-ne de formula integrării unui polinom, puterea fiecărui monom va fi incrementată, iar coeficientul va fi rezultatul împărțirii dintre coeficientul curent și puterea incrementată. Dacă noul coeficient nu va fi 0, se va adăuga la polinomul rezultat, altfel se va trece la următorul monom.

Metoda `divide` are ca parametru două polinoame, realizează operația de împărțire și va returna rezultatul sub forma unui polinom de monoame cu coeficienți reali. Principiul de funcționare este bazat pe metoda matematică folosită în schema lui Horner, restul fiind calculat în Controller după cum am menționat mai sus. Monoamele rezultate cu coeficienți egali cu 0 nu vor fi adăugate la rezultat. Această metodă se va folosi și de metodele `subtract` și `multiply` care realizează operația de scădere și înmulțire specificate mai sus, doar că lucrează pe polinoame de monoame cu coeficienți reali și returnează un polinom de monoame cu coeficienți reali.

Metoda `toString` are ca parametru un polinom și un număr întreg pentru a putea converti un polinom într-un String folosindu-se de metodele `toStringI` și `toStringD`. Dacă această metodă a fost apelată în urma unei împărțiri sau a unei integrări se va folosi `toStringD` deoarece vom lucra cu coeficienți reali, altfel se va folosi metoda `toStringI`.

- **Clasa Polinom**

Clasa polinom are scopul de a reprezenta input-urile ca polinoame ce conțin monoame, în cazul acesta fiecare polinom are câte un ArrayList de monoame.

Metoda `splitInput` are ca parametru un String și are rolul de a transforma acest String într-un polinom. În cazul în care datele introduse nu se pot converti se semnalează o eroare. Pentru o conversie corectă polinomul trebuie să aibă forma: $3x^2 + 1x^1 + 7x^0$. Pentru a realiza această conversie m-am folosit de metoda `split` și am separat șirul în funcție de `+`, `-`, `*` și `^`.

Metodele `toStringI` și `toStringD` transformă, după cum am spus mai sus, un String într-un polinom de monoame cu coeficienți întregi, respectiv reali. Se vor elimina monoamele cu coeficienți nuli.

Tot în această clasă am suprascris metoda `equals` și `hashCode` pentru a le folosi în ClasaTest din pachetul Teste.

- Clasa Monom

Această clasă abstractă implementează Comparable pentru a putea ordona șirul de monoame după putere, în ordine descrescătoare. Am suprascris metodele compareTo pentru ordonare, dar și metodele equals și hashCode pentru a le folosi în clasa ClasaTest.

Pe lângă metodele menționate mai sus, avem constructorul clasei și metodele getPow, getCoefficient1, getCoefficient2, setCoefficient1, setCoefficient2, ultimele 4 fiind suprascrise în clasele ce moștenesc clasa Monom. Câmpul pow reprezintă putere întreagă a unui monom.

- Clasa MonomInt

Această clasă moștenește clasa de mai sus și reprezintă un monom cu coeficienți întregi. De aceea, metodele abstracte ale sale sunt suprascrise aici. Avem getter-ul getCoefficient1 și setter-ul setCoefficient1 pentru coeficientul întreg al acestui monom. Am suprascris metodele equals și hashCode pentru a mă folosi de ele în testele Junit pe care o să le explic mai jos.

- Clasa MonomDou

Această clasă moștenește clasa Monom și reprezintă un monom cu coeficienți reali. De aceea, metodele abstracte ale sale sunt suprascrise aici. Avem getter-ul getCoefficient2 și setter-ul setCoefficient2 pentru coeficientul real al acestui monom. Am suprascris metodele equals și hashCode pentru a mă folosi de ele în testele Junit pe care o să le explic mai jos.

- Interfața utilizator

Această interfață este realizată în clasa View despre care am scris mai sus. Ea constă din 4 Jpanel-uri, 7 butoane pentru operațiile pe polinoame, 4 JTextField-uri pentru polinoamele introduse de utilizator și două pentru rezultat și rest. Aceasta mai conține și 5 JLabel-uri pentru a le indica utilizatorilor ce reprezintă fiecare JTextField, dar și pentru a indica cum ar trebui să arate datele introduse pentru a se evita o posibilă eroare. Aceasta se poate observa în imaginea de mai jos.

Tema1 - Calculator de polinoame

Polinoamele introduse trebuie sa aiba forma: 1*x^0+6*x^2+...

Dati polinomul 1

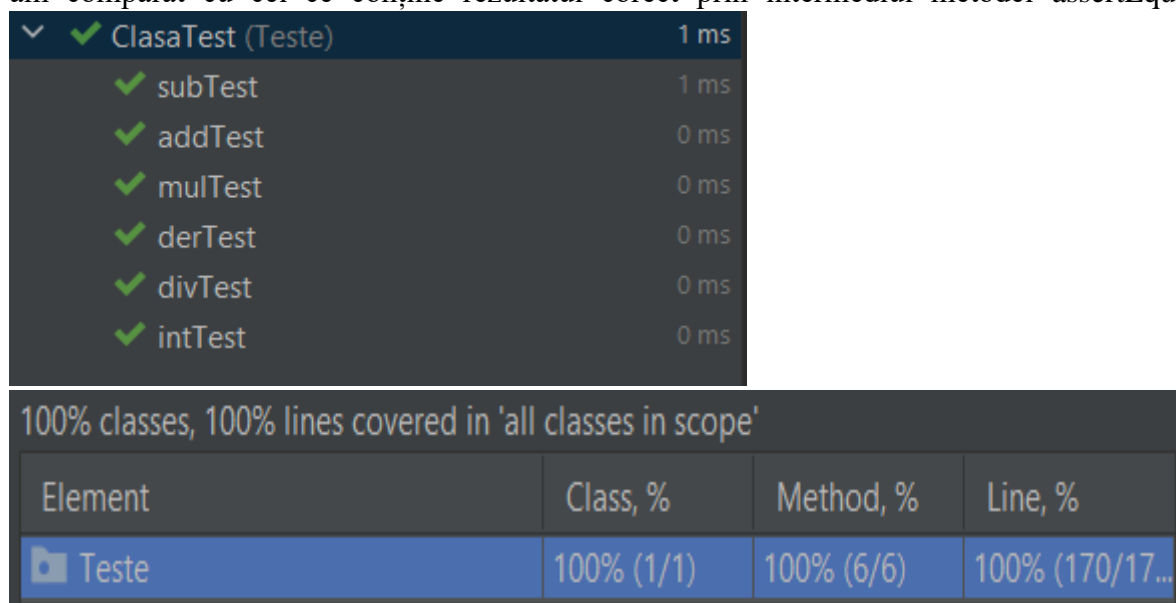
Dati polinomul 2

*	/	+
-	^	∫
=	Polinomul rezultat: <input type="text"/>	
	Restul: <input type="text"/>	

5. Rezultate

Testele le-am realizat în JUnit, unul pentru fiecare operație și au fost realizate astfel: au fost date două sau un polinom de intrare și unul cu rezultatul corect. Apoi am realizat operația dorită,

apelând metoda corespunzătoare operației dorite pentru a obține un polinom. Acest polinom l-am comparat cu cel ce conține rezultatul corect prin intermediul metodei `assertEquals`.



✓ ClasaTest (Teste)	1 ms
✓ subTest	1 ms
✓ addTest	0 ms
✓ mulTest	0 ms
✓ derTest	0 ms
✓ divTest	0 ms
✓ intTest	0 ms

100% classes, 100% lines covered in 'all classes in scope'			
Element	Class, %	Method, %	Line, %
Teste	100% (1/1)	100% (6/6)	100% (170/17...)

După cum se vede în imaginile de mai sus, testele au fost efectuate cu succes.

6. Concluzii

În concluzie, consider că această temă a fost o provocare căreia i-am făcut față cu succes, fiind mulțumită de ceea ce am implementat. Punctele forte ale acestei teme constau în aprofundarea limbajului de programare Java, conceptele programării orientate pe obiect și manipularea interfețelor grafice. Faptul că am folosit un model MVC m-a ajutat foarte mult, proiectul fiind mai ușor de manipulat și de înțeles pentru orice alt programator ce s-ar uita peste el.

7. Bibliografie

- Youtube- pentru a-mi aminti împărțirea polinoamelor
- Cursurile de POO din semestrul trecut