UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTAȚIE

CALCULATOR DE POLINOAME

BUMBU RĂZVAN-VASILE

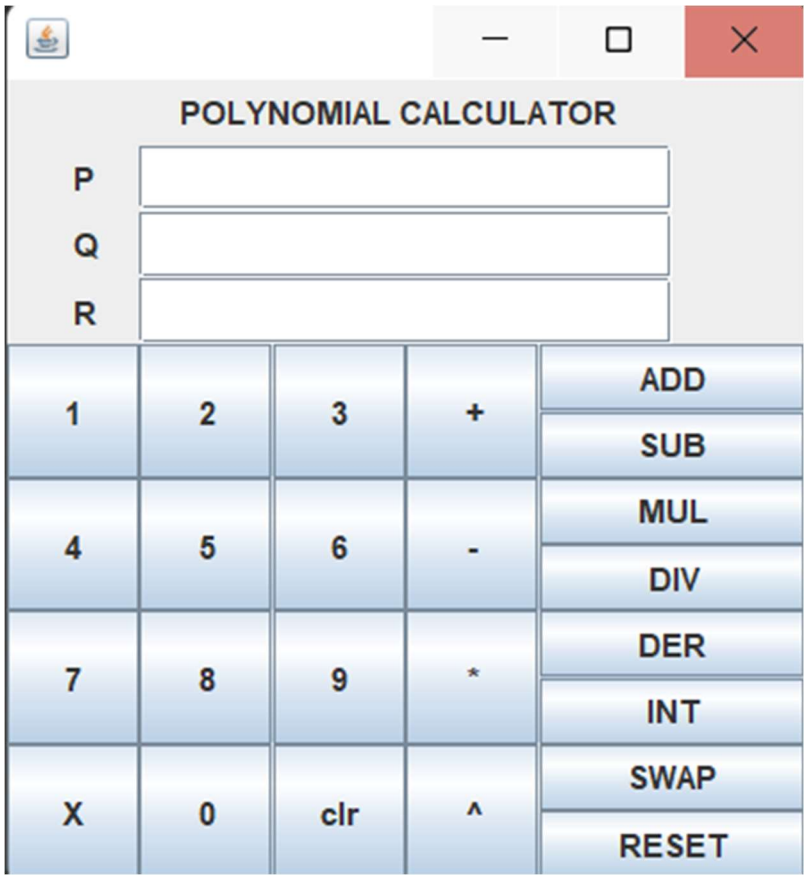
GRUPA 30223

CUPRINS

1. OBIECTIVUL TEMEI

Obiectivul temei este acela de a implementa o aplicație care simulează un calculator capabil să efectueze operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare asupra polinoamelor.

Calculator beneficiază de o interfață grafică prin intermediul căreia utilizatorul poate să introducă polinoamele dorite, după care prin intermediul butoanelor de selecție poate să realizeze operația dorită. Calculatorul beneficiază totodată de o interfață user-friendly, fiind ușor de folosit datorită denumirii sugestive a butoanelor.



2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

Calculatorul este capabil să implementeze operațiile de adunare, scădere, înmulțire, împărțire, derivare și integrare asupra polinoamelor. Pentru o mai ușoară manipulare, polinoamele sunt reprezentate de o listă de monoame, fiecare monom având un coeficient și un grad. Astfel, forma generală a unui monom va fi **A*X^B ($A \cdot X^B$).**

Interfața grafică (GUI) este o interfață cu utilizatorul bazată pe un sistem de afișaj ce utilizează elemente grafice. Interfață grafică este numit sistemul de afișaj grafic-vizual de pe un ecran, situat funcțional între utilizator și dispozitive electronice cum ar fi computere. Pentru a prezenta toate informațiile și acțiunile posibile, GUI oferă pictograme și indicatori vizuali, în contrast cu interfețele bazate pe text, care oferă doar nume de comenzi (care trebuie tastate) sau navigația text.

Interfața grafică a calculatorului este una foarte ușor de folosit, butoanele având denumiri sugestive. Astfel, utilizatorul poate să introducă polinomul dorit atât de la tastatură, cât și prin intermediul butoanelor din cadrul interfeței. Un lucru esențial pentru funcționarea corectă a calculatorului este forma polinomului introdus. Mai precis, fiecare monom trebuie să fie de forma **A*X^B** și să fie separat de unul dintre operatorii + sau **-.** Dacă de exemplu un monom nu este introdus corect (exemplu de greșeli: **AX^B, A*XB, AX** etc), acesta nu va fi recunoscut de calculator, operațiile neputând fi realizate.

În cazul în care s-a efectuat o greșeală în scrierea polinomului, se poate reveni la pasul anterior prin simpla apăsare a butonului **CLR.**

În cazul în care se dorește ștergerea totală a polinoamelor, se poate apăsa butonul **RESET.**

Pentru navigarea între polinoame (în momentul în care se dorește introducerea acestora) se va utiliza butonul **SWAP** (inițial este selectat primul polinom pentru introducere).

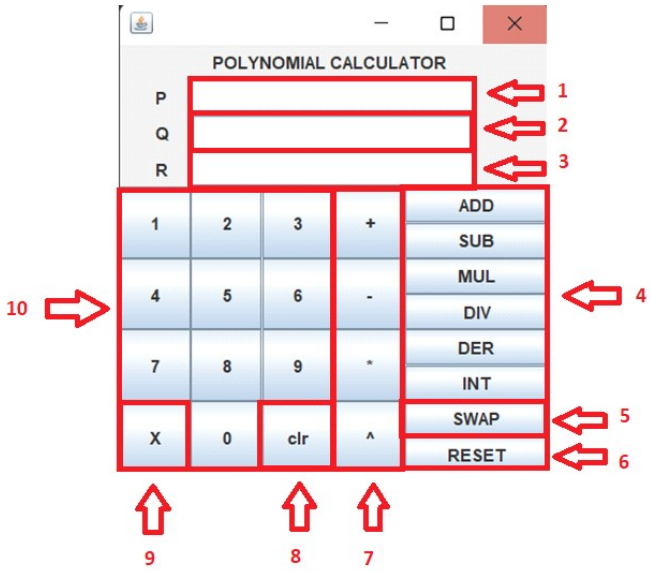
Operația dorită se realizează prin simpla apăsare a butonului corespunzător ei (butoanele au denumiri sugestive).

O diagramă use-case este o reprezentare a interacțiunii unui utilizator cu sistemul, ea arătând relația dintre utilizator și diferitele cazuri de utilizare în care este implicat acesta.

Exemplu de use-case:

1. Utilizatorul introduce primul polinom
2. Utilizatorul introduce cel de-al doilea polinom
3. Utilizatorul apasă butonul specific operației dorite
4. Utilizatorul primește rezultatul operației

În cazul în care formatul unuia dintre polinoamele introduse nu corespunde cu formatul cerut, atunci nu se va returna niciun rezultat.



LEGENDĂ:

1 – *caseta primului polinom* – în aceasta se va introduce primul polinom

2 – *caseta celui de-al doilea polinom* – în aceasta se va introduce cel de-al doilea polinom

3 – *caseta polinomului rezultat în urma operației* – în aceasta va fi afișat polinomul rezultat în urma operației

4 – *butoane operații* – prin apăsarea lor se va realiza operația dorită

5 – *buton SWAP* – folosit pentru a face selecția între polinoame

6 – *buton RESET* – folosit pentru a șterge toate datele introduse în casetele polinoamelor

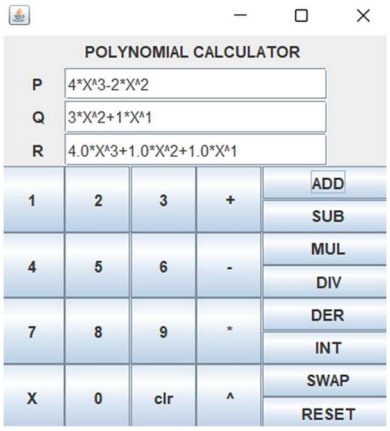
7 – *butoane introducere semne* – prin apăsarea lor vor fi introduși operatorii +-*^ în casetele polinoamelor

8 – *buton CLR* – prin apăsarea lui se va șterge ultimul caracter introdus în casetele polinoamelor

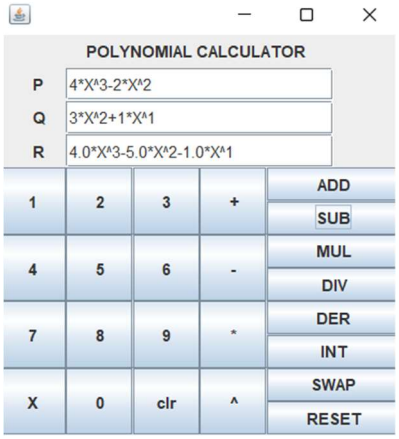
9 – *buton X* – prin apăsarea lui se va introduce caracterul X în casetele polinoamelor

10 – *butoane cifre* – prin apăsarea lor se vor introduce cifrele în casetele polinoamelor

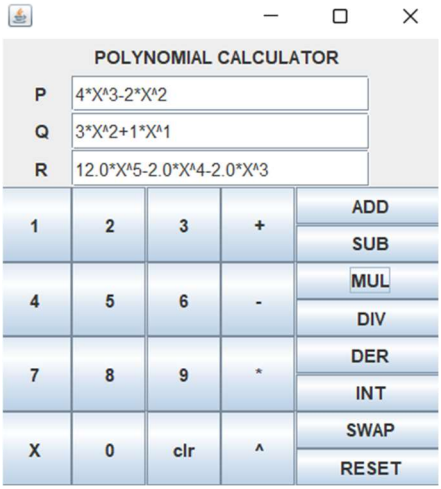
ADUNAREA A DOUĂ POLINOAME



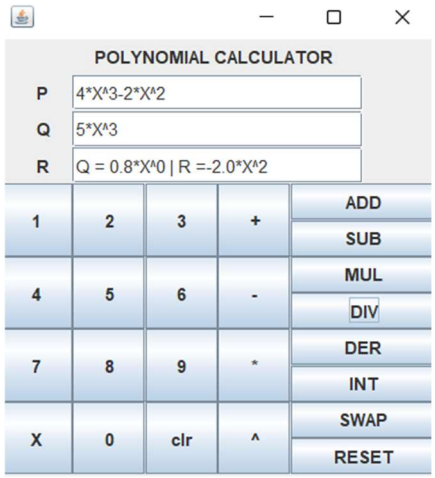
SCĂDEREA A DOUĂ POLINOAME



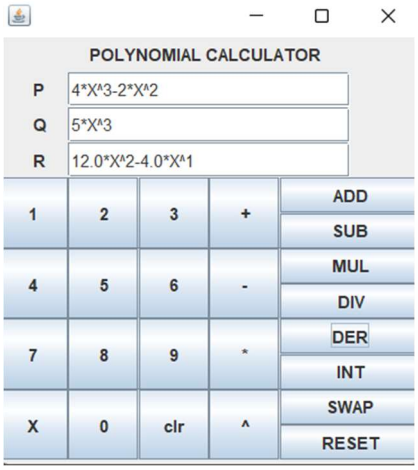
ÎNMULȚIREA A DOUĂ POLINOAME



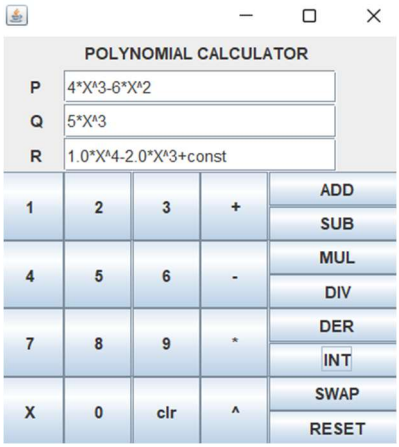
ÎMPĂRȚIREA A DOUĂ POLINOAME



DERIVAREA UNUI POLINOM



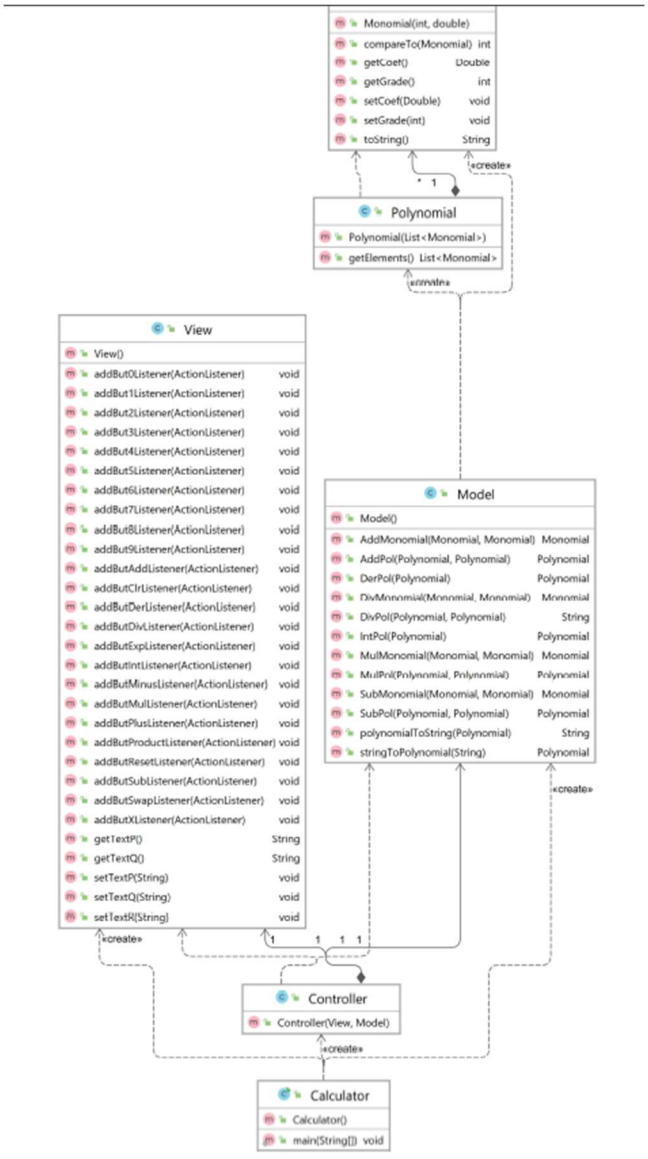
INTEGRAREA UNUI POLINOM



3. PROIECTARE

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.

Diagrama UML a calculatorului este formată din 6 clase: Monomial, Polynomail, Model, View, Controller și Calculator. Legăturile dintre aceste clase sunt prezentate în imaginea de mai jos.



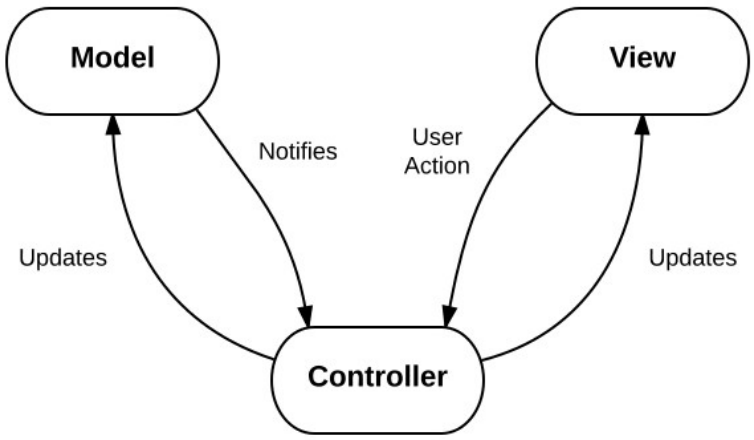
Model-View-Controller (MVC) este un model arhitectural utilizat în ingineria software. Succesul modelului se datorează izolării logicii de business față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual și nivelele inferioare ale regulilor de business sunt mai ușor de modificat, fără a afecta alte nivele.

Calculatorul are la bază o arhitectură de tip MVC. Astfel, clasa Controller devine principala componentă. Aceasta realizează conexiunile dintre clasele Model și View. Modelul este în permanență actualizat prin intermediul controllerului, căruia îi transmite datele. Totodată, view-ul este actualizat în permanență de controller, căruia îi transmite informații despre acțiunile utilizatorului.

Modelul controlează operațiile de utilizare a informației (trimise dinainte de către rangul său superior) pentru a rezulta o formă mai ușoară de înțeles.

View-ul corespunde reprezentării grafice, sau mai bine zis, exprimării formei datelor. Acesta este reprezentat de interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la Controller.

Controllerul are rolul de a controla întreaga aplicație. Se pot controla fișiere, scripturi, programe (în general cam orice tip de informație permisă de interfață). În acest fel putem diversifica conținutul nostru de o formă dinamică și statică, în același timp.



4. IMPLEMENTARE

Calculatorul este constituit din 6 clase: Monomial, Polynomail, Model, View, Controller și Calculator. Fiecare clasă implementează metodele specifice.

- **CLASA MONOMIAL**

Această clasă implementează caracteristicile specifice unui monom: grad și coeficient. Variabilele clasei sunt de tip private (pot fi vizibile doar în interiorul clasei). Metodele implementate în această clasă sunt:

- *public Monomial (int g, double c)* – constructorul clasei (acesta creează monomul).
- *public String toString()* – transformă monomul într-un șir de caractere.
- *public int compareTo(Monomial m)* – realizează compararea gradelor monoamelor (folosită pentru sortarea lor în cadrul polinoamelor).
- *public int getGrade()* – returnează gradul monomului.
- *public void setGrade(int grade)* – setează gradul monomului.
- *public Double getCoef()* – returnează coeficientul monomului.
- *public void setCoef(int coef)* – setează coeficientul monomului.

- **CLASA POLYNOMIAL**

Această clasă implementează caracteristicile specifice unui monom. Astfel, polinomul este alcătuit dintr-o listă de monoame, lista fiind de tip private. Metodele implementate în această clasă sunt:

- *public Polynomial(List<Monomial> e)* – constructorul clasei (acesta creează polinomul).
- *public List<Monomial> getElements()* – returnează lista de monoame ale polinomului.

- **CLASA MODEL**

Această clasă implementează principalele metode necesare pentru implementarea calculatorului. Metodele implementate în această clasă sunt:

- *public Monomial AddMonomial (Monomial m1, Monomial m2)* – realizează adunarea a două monoame. Rezultatul returnat este de tip Monomial.
- *public Monomial SubMonomial (Monomial m1, Monomial m2)* – realizează scăderea a două monoame. Rezultatul returnat este de tip Monomial.
- *public Monomial MulMonomial (Monomial m1, Monomial m2)* – realizează înmulțirea a două monoame. Rezultatul returnat este de tip Monomial.
- *public Monomial DivMonomial (Monomial m1, Monomial m2)* – realizează împărțirea a două monoame. Rezultatul returnat este de tip Monomial.
- *public Polynomial stringToPolynomial (String s)* – realizează transformarea unui șir de caractere în polinom. Rezultatul returnat este de tip Polynomial.
- *public String polynomialToString (Polynomial p)* - realizează transformarea unui polinom în șir de caractere. Rezultatul returnat este de tip String.

- *public Polynomial AddPol (Polynomial p1, Polynomial p2)* – realizeaază adunarea a două polinoame. Rezultatul returnat este de tip Polynomial. Aceasta transformă polinoamele primite ca și parametri în liste, realizeaza operația de adunare conform unui algoritm pe elementele listei, salvează rezultatul într-o nouă listă pe baza căreia se va consrtui rezulatul returnat.
- *public Polynomial SubPol (Polynomial p1, Polynomial p2)* - realizeaază scăderea a două polinoame. Rezultatul returnat este de tip Polynomial. Aceasta transformă polinoamele primite ca și parametri în liste, realizeaza operația de scădere conform unui algoritm pe elementele listei, salvează rezultatul într-o nouă listă pe baza căreia se va consrtui rezulatul returnat.
- *public Polynomial MulPol (Polynomial p1, Polynomial p2)* - realizeaază înmulțirea a două polinoame. Rezultatul returnat este de tip Polynomial. Aceasta transformă polinoamele primite ca și parametri în liste, realizeaza operația de înmulțire conform unui algoritm pe elementele listei, salvează rezultatul într-o nouă listă pe baza căreia se va consrtui rezulatul returnat.
- *public String DivPol (Polynomial p1, Polynomial p2)* – realizeaază împărțirea a două polinoame. Rezultatul returnat este de tip String. Aceasta transformă polinoamele primite ca și parametri în liste, realizeaza operația de împărțire conform unui algoritm pe elementele listei, salvează rezultatul într-o nouă listă pe baza căreia se va consrtui rezulatul returnat.
- *public Polynomial DerPol (Polynomial p)* – realizeaază derivarea unui polinom. Rezultatul returnat este de tip Polynomial.
- *public Polynomial IntPol (Polynomial p)* - – realizeaază derivarea unui polinom. Rezultatul returnat este de tip Polynomial.

- **CLASA VIEW**

Această clasă implementează interfața grafică a calculatorului. Astfel, aceasta are ca și parametri butoanele, textfield-urile, label-urile și panel-ul interfeței grafice. Butoanele au denumiri sugestive, implementând totodată și metode de ActionListener pentru evenimentele de apăsare a butonului. Butoanele sunt de mai multe feluri: butonul X (inserează în caseta polinomului textul X), butoanele de cifre (inserează în caseta polinomului cifrele de la 0 la 9), butoanele de operatori (inserează în caseta polinomului operatorii +-*^), butoanele pentru operații pe polinoame (realizează operația dorită asupra polinoamelor), butonul CLR (șterge ultimul caracter introdus), butonul SWAP (navighează între polinoame pentru introducerea acestora) și butonul RESET (șterge în totalitate textul introdus în casetele polinoamelor). Interfața conține trei zone importante de text: zona în care se introduce primul polinom, zona unde se introduce cel de-al doilea polinom și zona unde se afișează rezultatul. Aceste trei zone sunt reprezentate de TextField-uri care pot fi completate atât de la butoanele interfeței grafice, cât și de la tastatură. Label-urile sunt folosite pentru a eticheta casetele polinoamelor (pentru a identifica mai ușor polinoamele între ele). Panel-ul reunește toate componentele menționate mai sus într-o singură componentă.

- **CLASA CONTROLLER**

Această clasă realizează conexiunea dintre clasele View și Model. Principalele componente ale acestei clase sunt două obiecte de private, unul de tip View și unul de tip Model. Obiectul de tip View este răspunzător pentru interfața grafică, în timp ce obiectul de tip Model este răspunzător pentru operațiile realizate. Obiectul de tip model este în permanență actualizat prin intermediul clasei Controller, căreia îi transmite datele. Obiectul de tip View este actualizat în permanență prin intermediul clasei Controller, căreia îi transmite informații despre acțiunile utilizatorului. În cadrul acestei clase sunt definite răspunsurile la acțiunile de click ale utilizatorului. Astfel, conform butonului apăsat de utilizator se va realiza operația dorită (inserție, calcul, selecție).

- **CLASA CALCULATOR**

Această clasă este implementarea propriu zisă a calculatorului. Astfel, prin intermediul acestei clase calculatorul rulează sau nu. Clasa conține o metodă statică main care creează trei obiecte: un obiect de tip Mode, un obiect de tip View și un obiect de tip Controller. Astfel, această clasă devine "creierul" din spatele calculatorului.

5. REZULTATE

Pentru testare au fost folosite doar date corecte. Cu ajutorul JUnit4, a fost creată o clasă specială destinată testelor. Clasa conține un obiect de tip Model (modelul în cadrul căruia sunt implementate operațiile), două obiecte de tip Monomial (pentru testarea operațiilor pe monoame) și câte trei obicete de tip String, respectiv Polynomial (pentru implementarea operațiilor pe polinoame). Clasa conține o metodă beforeClass în care sunt inițializate obiectele. Apoi, pentru cele 10 metode din cadrul clasei Model a fost realizată câte o metodă de testare cu nume sugestiv. Pentru testare au fost folosite date valide, încercând totodată sa fie generate cazuri cât mai greu de testat (cazuri în care riscul de apariție al unor greșeli este foarte ridicat). Toate cele 10 teste au avut succes.

6. CONCLUZII

În concluzie, această temă m-a ajutat să aprofundez paradigmele Programării Orientate pe Obiect. Astfel, am reușit să îmi dezvolt abilitățile de a lucra cu liste, de a lucra cu interfața grafică și totodată de a mă familiariza cu structura de tip Model View Controller (MVC). De asemenea, consider că am reușit să îmi îmbunătățesc și abilitățile gândirii logice și matematice.

Calculatorul poate fi îmbunătățit. De exemplu, interfața grafică poate primi un aspect mai modern și poate chiar mai ușor de folosit. Tiparul standard de introducere al polinoamelor ar putea fi schimbat (de exemplu, să se accepte și monoame scrise sub forma AX^B). De asemenea, ar putea fi introduse operații noi precum aflarea rădăcinilor polinomului. Un alt punct important care ar putea fi introdus ar putea fi introducerea polinoamelor cu coeficienți reali și exponenți întregi.

7. BIBLIOGRAFIE

- https://www.baeldung.com/
- https://ro.wikipedia.org/wiki/Pagina_principal%C4%83
- https://www.w3schools.com/java/