***Documentație Tema 1***

**Realizator: Stroia Lucian Dorin**

*● Cerința temei*

**Cuprins**

*● Scopul realizării aplicației*

*● Abordarea problemei*

*● Structura interna a programului*

*● Diagrame*

*● Concluzii*

***Cerința temei***

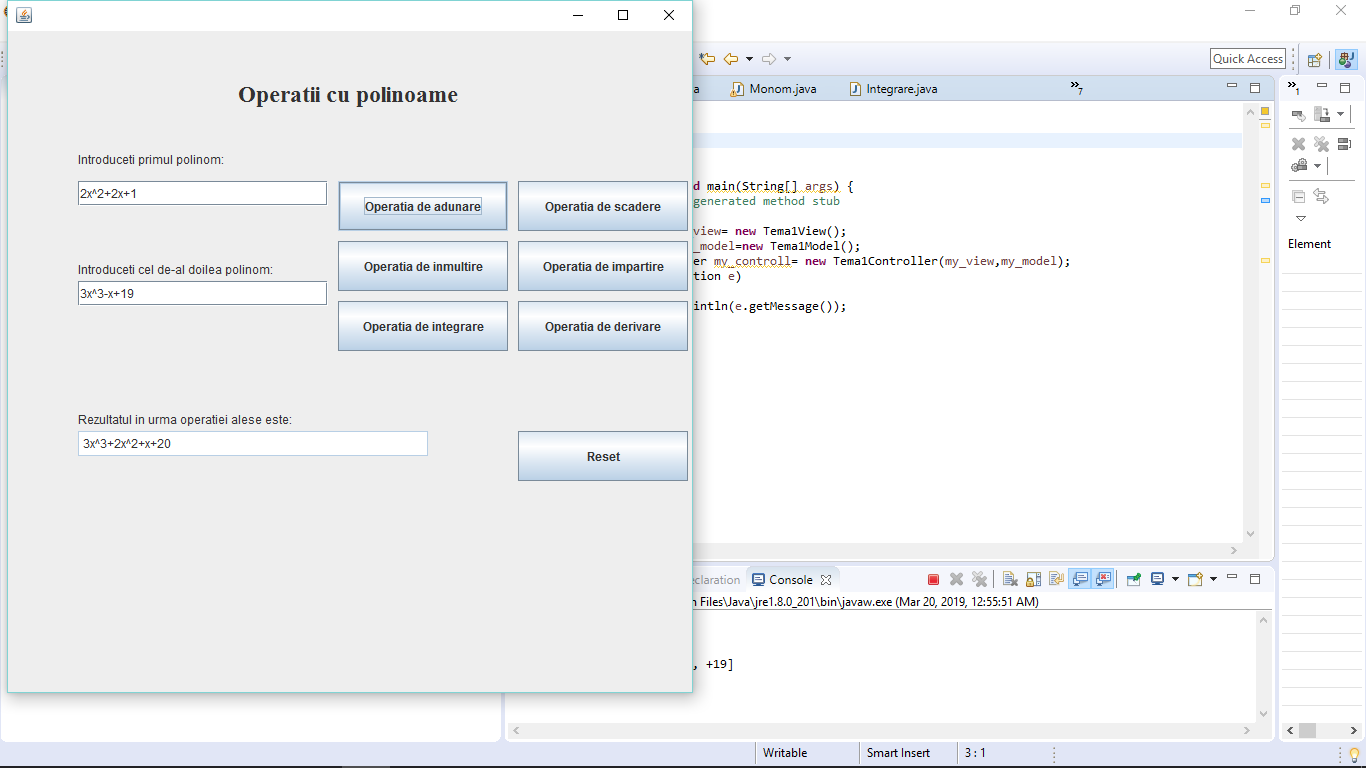
*Se cere dezvoltarea unei aplicații JAVA care are ca și scop efectuarea de operații de bază între 2 polinoame, de o singură variabilă, având coeficienți întregi, rezultatul fiind reprezentat tot sub forma unui polinom de aceeași structură.*

*Operațiile dorite pentru implementare sunt:*

* 1. *Adunare*
  2. *Scădere*
  3. *Înmulțire*
  4. *Împărțire*
  5. *Integrare( se realizează integrarea primului polinom introdus)*
  6. *Derivare( se realizează derivarea primului polinom introdus)*

***Scopul realizării aplicației***

*Această aplicație permite utilizatorului introducerea a doua polinoame sub forma anX^bn+an-1X^bn-1+…+a1X^b1+a0X^b0*, *mulțimea valorilor întregi (an…a0) fiind mulțimea coeficienților monumului corespunzător iar mulțimea valorilor întregi (bn…b0) fiind mulțimea gradelor monumului corespunzător. După introducerea celor doua polinoame se dorește efectuarea de operații aritmetice între cele doua expresii, utilizatorul având posibilitatea de a alege ce operație dorește să efectueze prin intermediul interfeței grafice.*



***Abordarea problemei***

*Pentru înțelegerea datelor introduse de către utilizator, stringul introdus de către acesta este organizat in mai multe monoame, care au ca si caracteristici coeficientul monomului dar si puterea acestuia. Astfel, putem defini un monom ca fiind o structură de doi întregi, esențiali în realizarea operațiilor intre 2 polinoame.*

***Structura internă a programului***

*Programul este structurat in 3 pachete, fiecărui pachet fiindu-i atribuit mai multe clase care în care sunt descrise metodele aplicate asupra datelor introduse de către utilizator astfel încât să se afișeze rezultatul corect din punct de vedere matematic.*

*♦Primul pachet, numit ”polinoame”, are ca și componente următoarele clase:*

1. *Clasa Monom*

*În această clasă, sunt definite 2 variabile de instanță, coeficientul monomului si gradul*

*acestuia, ambele fiind definite ca și variabile întregi. Constructorul specific acestei clase primește ca și parametri cele 2 valori întregi care reprezintă coeficientul si gradul monomului, astfel fiind posibilă crearea unui astfel de obiect care urmează să fie stocat în polinom.*

*Metoda toString a fost realizată astfel încât sa returneze reprezentarea monomului1, fiind luate în considerare mai multe cazuri particulare, când 2 monoame pot să difere atunci când sunt afișate. De exemplu, reprezentarea monomului ”x” diferă față de reprezentarea monomului ”2x^3” deoarece, chiar dacă monomul ”x” este stocat în aplicație având coeficientul 1 si gradul 1, se dorește afișarea acestuia cu o simplă variabilă , și anume ”x”.*

*Această clasă implementează interfața ”Comparable” deoarece se dorește ordonarea crescătoare a monoamelor care formează polinomul în funcție de gradul monoamelor. Din acest motiv, este descrisă și metoda specifică interfeței ”Comparable”, metoda compareTo(Object obj) care ordonează crescător monoamele în funcție de gradul acestora.*

1. *Clasa Polinom*

*Această clasă este responsabilă pentru transformarea Stringului primit ca și parametru,*

*fiind preluat din JTextField-ul din interfața grafică, și trecut prin mai multe metode care au ca și rol aflarea de monoame cu coeficientul și gradul specific.*

*Ca si variabile de instanță, această clasă are definite un ArrayList de Monoame, un String care face referire la Stringul trimis prin parametru la apelul constructorului, un ArrayList de Stringuri, un String al acestui ArrayList reprezentând forma inițiala a Monomului, înainte să fie prelucrat, și un ”numărposibil” care are ca si rol determinarea numărului maxim de monoame rezultat în urmă secționării Stringului principal.*

*Această clasă are definit 2 constructori, unul care primește ca paramtru Stringul introdus de către utilizator, și un constructor care poate primi direct un ArrayList de Monoame.*

*◊ Metoda ”transformare( )” are ca și scop extragerea posibililor Stringuri care pot fi un monom, din Stringul principal si adăugarea acestora într-un ArrayList. Numărul posibil de monoame este determinat prin intermediul calculării numărului de ”+” respectiv ”-” din Stringul trimis ca parametru însă este luam în considerare si posibilitatea de începere a acestuia fără vreun semn matematic, în cazul în care inițial se începe cu un caracter care reprezintă o cifră sau o parte dintr-un număr pozitiv( ex: ”2x^2”).*

*În cadrul acestei metode, este apelată o altă metodă definită în cadrul clasei Polinom, și anume metoda ”taiere( )” care, în funcție de parametri primiți și anume ”int deUnde, char caracterul, String aux” salvează în Stringul aux caracterele care încep de la poziția ”deUnde” până la întâlnirea primului ”+” sau ”-”, moment în care Stringul aux este adăugat în ArrayList-ul de Stringuri, care urmează să fie supus altor metode pentru a afla coeficientul respectiv gradul fiecărui monom.*

*◊ Metoda ”sectionarePolinom( )” are ca și scop transformarea dintr-un ArrayList de Stringuri, ArrayList obținut în urma apelului metodei explicate anterior, într-un ArrayList de Monoame, prin crearea fiecărui monom specific polinomului în urma aplicării a doua metode care au ca și scop aflarea coeficientului respectiv aflarea gradului pe baza unui element din ArrayList-ul de Stringuri.*

*◊ Metoda ”aflareCoeficient(String a)” are ca și scop aflarea primului număr din Stringul trimis ca și parametru, prin formarea acestui întreg din mai multe caractere identificate, fiind luate în considerare fiecare caracter si definit natura acestuia. Adică daca primul caracter citit din Stringul trimis ca și parametru este ”–” atunci următoarele caractere descoperite până la întâlnirea lui ”x” vor forma un coeficient negativ. Sunt luate în considerare toate cazurile de reprezentare a unui monom, adică doar sub forma unui număr întreg, x-ul si puterea lipsind, dar și cazurile în care Stringul transmis ca și parametru este reprezentat de ”-x” respectiv ”+x”, caz în care coeficientul este salvat ca fiind 1 respectiv -1.*

*◊Metoda ”aflareGraf(String a)” are ca și scop aflarea gradului corespunzător monomului pe baza Stringului trimis ca și parametru, fiind luate în considerare mai multe cazuri, precum lipsa vreunui grup de caractere după apariția caracterului ”x”, caz în care gradul monomului este 1, cazul în care nu apare variabila ”x”, caz în care gradul monomului respectiv este 0. Procedeul de formare a numărului din caracterele descoperite după ”^” este asemănător cu cel aplicat pentru aflarea coeficientului.*

1. *Clasa Adunare*

*Această clasă are ca și scop realizarea unui polinom a cărui monoame se formează prin*

*adunarea corespunzătoare a monoamelor alte doua polinoame.*

*◊În această clasă, este definită o metoda ”asemenea(Polinom poli1,Polinom poli2)” ca*

*returnează o variabilă booleană in funcție de care ne putem da seama dacă cele 2 polinoame între care dorim să realizăm operația de adunare sunt opuse, adică dacă le adunăm, rezultatul va fi 0. Metoda ”asemenea” returnează false daca cele 2 polinoame sunt complementare, adică adunate dau rezultatul 0, si true în caz contrar.*

*◊Metoda ”add(Polinom p1, Polinom p2) ” primește ca și parametri cele 2 polinoame si returnează alt polinom care rezultă prin adunarea celor 2 argumente. În primă fază, sunt examinate cele 2 polinoame prin intermediul metodei ”asemenea” iar daca cele 2 polinoame sunt complementare, având rezultatul 0 în urma adunării, in ArrayListul de monoame al polinomului rezultat adăugăm un monom fals, care are coeficientul 0 si gradul 0. În caz contrar, in ArrayList-ul unui polinom auxiliar sunt puse pe rând elementele celor 2 polinoame, monom cu monom, iar apoi, se realizează o parcurgere a acestui ArrayList de monoame (al auxiliarului) și în cazul în care găsim 2 monoame care au același grad, setăm coeficientul unuia dintre acesta ca fiind suma dintre propriul coeficient si coeficientul celui de-al doilea monom cu care corespunde din punctul de vedere al puterii, iar celălalt monom, este setat sa conțină coeficientul 0 respectiv gradul 0.*

*La final, este parcurs ArrayList-ul polinomului auxiliar și în cazul în care nu se găsește un monom care nu are coeficientul ȘI gradul egale cu 0, acel monom se adaugă în ArrayList-ul polinomului rezultat care urmează să fie returnat.*

1. *Clasa Scădere*

*Această clasă este destul de asemănătoare cu Clasa Adunare, diferența principală fiind*

*reprezentată de procedeul de adăugarea în ArrayList-ul polinomului auxiliar, a monoamelor celui de-al doilea polinom, fiind pusă în valoare operația de scădere, i-am adăugat cu coeficientul schimbat, adică cei pozitivi i-am transformat în negativi iar cei negativi în pozitivi. Astfel, când se realizează identificarea monoamelor din auxiliar care au gradul identic, coeficienții acestora se ADUNĂ, simulând scăderea. Se obține astfel, un polinom care are ca și monoame rezultatul scăderii dintre monoamele celor 2 polinoame transmise ca și paramtri.*

1. *Clasa Înmulțire*

*Înmulțirea dintre două polinoame se realizează prin înmulțirea propriu-zisă a fiecărui*

*monom din primul polinom cu fiecare monom din cel de-al doilea polinom.*

*Acest procedeu a fost aplicat și în cadrul metodei ”mul” definite în această clasă, metodă care primește ca și parametru 2 polinoame, urmând să returneze un alt polinom care este rezultatul înmulțirii celor 2 polinoame argumente.*

*Au fost luate în considerare mai multe aspecte, precum posibilitatea de a înmulții un monom cu alt monom, unul din ei având coeficientul 0, sau gradul 0, motiv pentru care au fost tratate fiecare dintre aceste cazuri în parte în vederea obținerii unui rezultat corect și cat mai ușor de afișat.*

1. *Clasa Integrare*

*În această clasă, am definit o metodă care primește ca și parametru primul polinom*

*trimis de către utilizator și returnează un String, din cauza faptului că în urma integrării, apar coeficienți care nu sunt întregi, alegerea mea fiind de reprezenta acești coeficienți prin introducerea caracterului ”/” pentru a da de înțeles că există un raport.*

*Pentru a face Integrarea, am utilizat un ArrayList de Stringuri suplimentar pentru a memora gradul fiecărui monom sub forma unui String, iar după, am parcurs polinomul transmis prin parametru și am realizat o afișare pe mai multe cazuri, în funcție de natura fiecărui coeficient al monomului din ArrayList-ul polinomului transmis prin parametru. Gradul monoamelor s-a modificat fiind crescut cu o unitate, iar la coeficient, a apărut caracterul ”/” urmat de Stringul salvat în ArrayListul suplimentar, String care reprezintă puterea înaintea majorării cu o unitate.*

*Astfel, avem ca și output un String care este particularizat în funcție de modificările pe care polinomul le suferă în urma procesului de integrare.*

1. *Clasa Derivare*

*Pentru derivarea unui polinom, am ales să implementez o metodă care primește ca și parametri un polinom, urmând să fie parcurs. În cazul în care gradul monomului este egal cu 0, atunci coeficientul este setat la 0. Dacă nu, coeficientul este setat astfel încât să fie produsul dintre valoarea actuală a coeficientului si gradul monomului. Ultima modificare constă în diminuarea gradului cu o unitate, fiind urmată returnarea polinomului rezultat.*

*♦ Cel de-al doilea pachet ”interfața” cuprinde următoarele clase:*

1. *Clasa Tema1View*

*Această clasă extinde clasa JFrame, clasa specifică interfețelor grafice. Ca și variabile de*

*instanță, sunt definite mai multe variabile de tipuri specifice clasei JFrame, precum butoane(JButto), textfield-uri(JTextField), frame (JFrame) etc.*

*Am creat constructorul clasei Tema1View, iar in momentul in care acesta a fost apelat, se*

*execută toate atribuirile din interiorul său, făcând posibilă deschiderea frame-ului cu care interacționează utilizatorul, unde are dreptul să introducă cele 2 polinoame și să selecteze operațiile pe care dorește să le efectueze prin actionarea cu un simplu click asupra unui buton care are un mesaj semnificativ, în funcție de acțiunea pe care o declanșează.*

*Este definită și o metodă suplimentară care are ca și rol detectarea corectitudinii de introducere a polinomului de către utilizator, dându-i posibilitatea să își corecteze erorile pe care le-a comis. Metoda este numita ”Eroare”.*

1. *Clasa Tema1Model*

*Această clasă este cumva main-ul ascuns al programului, aici fiind urmați pașii pentru*

*obținerea unui rezultat dorit.*

*Sunt definite metode precum adunarePolinoame, scaderePolinoame etc. care primesc ca și parametru Stringul introdus de către utilizator in Vederea Grafică.*

*Astfel, aici se apelează construcția polinoamelor cu String-urile preluate Tema1View, și sunt apelate metodele specifice claselor care poartă numele operației pe care o fac. Ex: Adunare, Scadere, etc.*

1. *Clasa Tema1Controller*

*Această clasă unește Tema1Model si Tem1View, definind listenere pentru fiecare buton*

*la apăsarea butonului ”Operația de adunare”, se preiau informații din Tema1View si se transmit in Tema1Model. Se obține astfel rezultatul dorit care urmează să fie afișat in JTextField-ul care nu este editabil de către utilizator.*

*♦ Ultimul pachet ”teste” cuprinde doar clasa test:*

1. *Clasa Test*

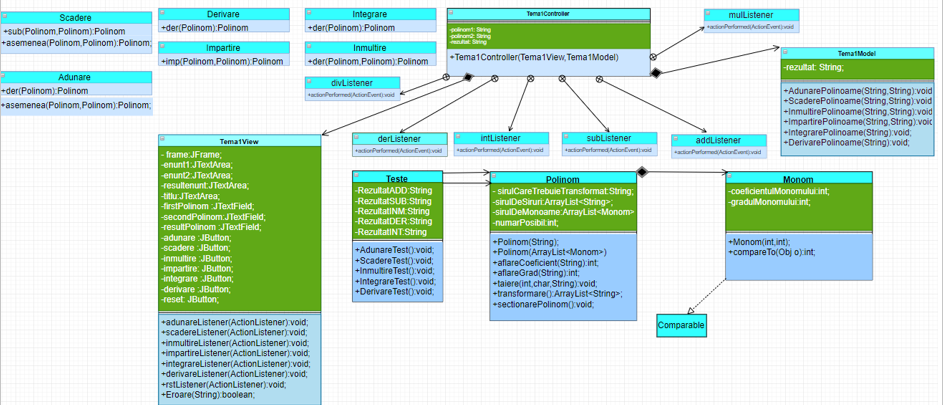
*Această clasă a fost creată special pentru testarea funcționalităților definite in cele*

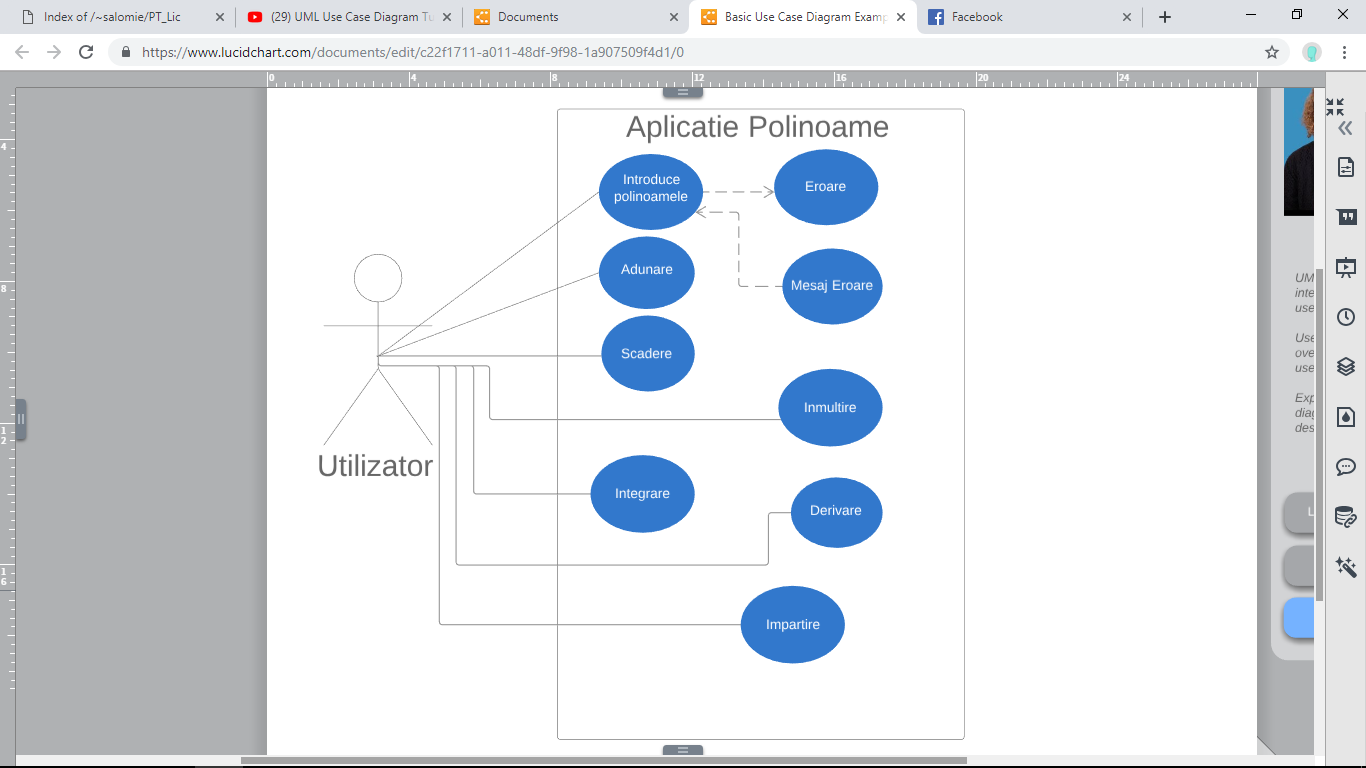
*pachete descrise, fiind definite câteva variabile de instanță care reprezintă niște rezultate corecte si sigure în urma operațiilor aritmetice aplicate asupra 2 polinoame date ca și exemplu.*

*Astfel s-a creat cate o metoda @Test pentru fiecare funcționalitate unde prin intermediul sintaxei ”assertequals” sunt verificate daca rezultatul obținut în urma apelului unei metode care are ca și scop efectuarea unei operații aritmetice coincide cu rezultatul pe care l-am definit ca si variabila de instanță, fiind convinși că acest rezultat definit de noi este corect. În urma rulării metodelor test, se poate observa faptul ca se verifică coincidența dintre ce ne așteptăm să obținem si ceea ce obținem.*

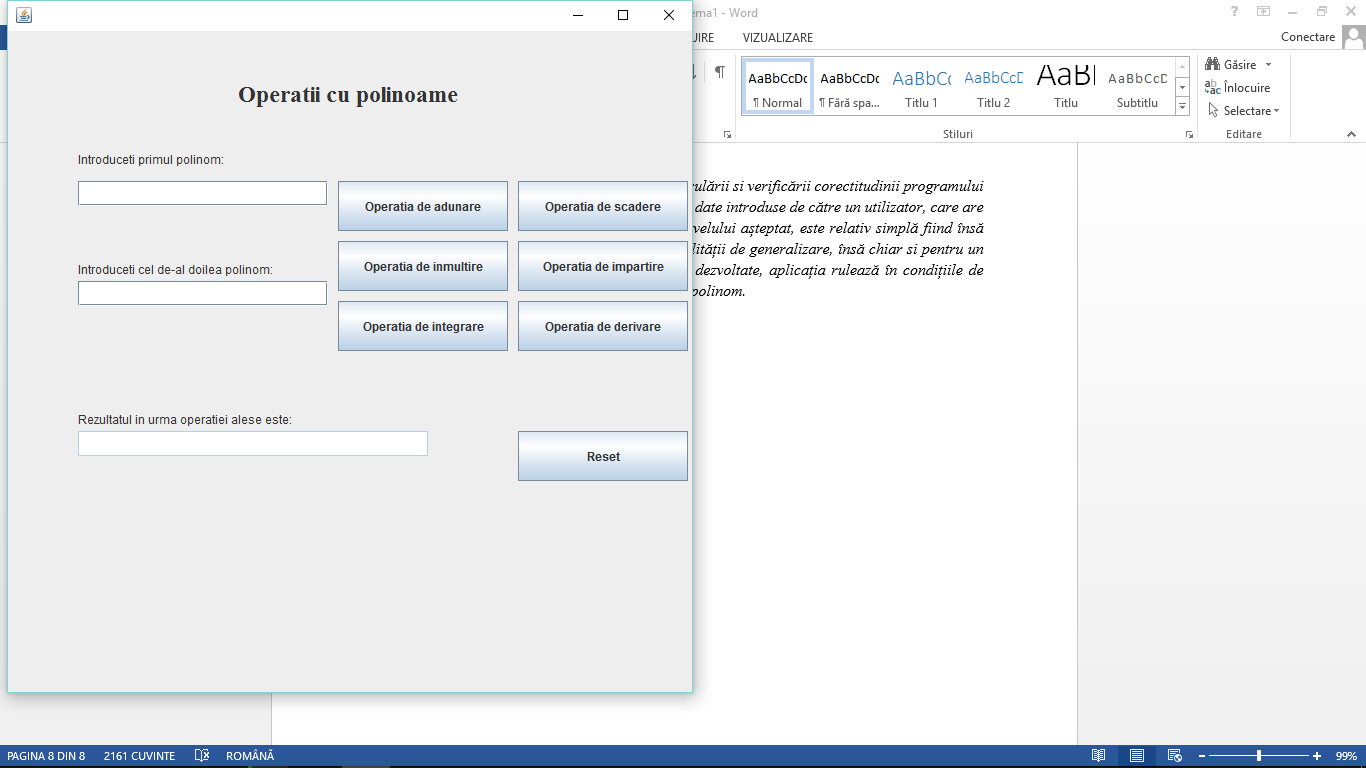
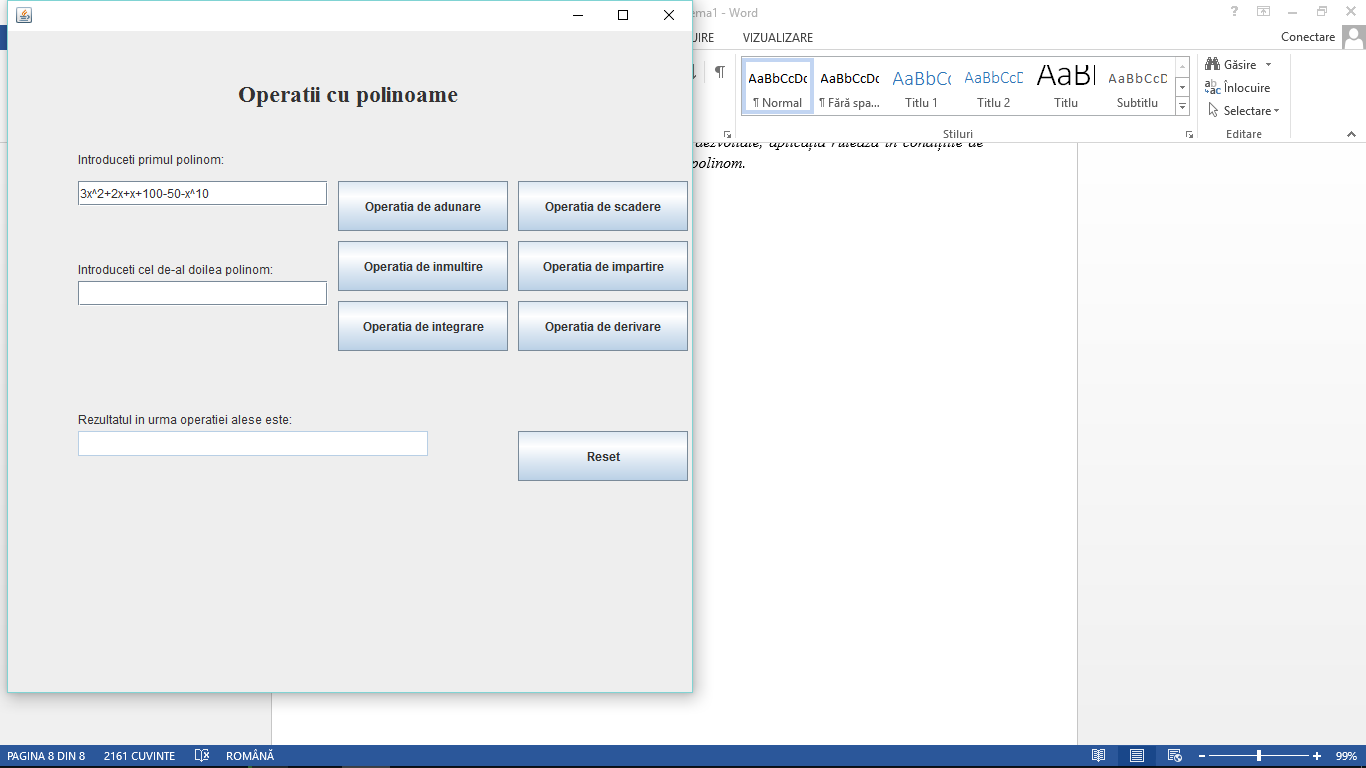
***Diagrame UML***

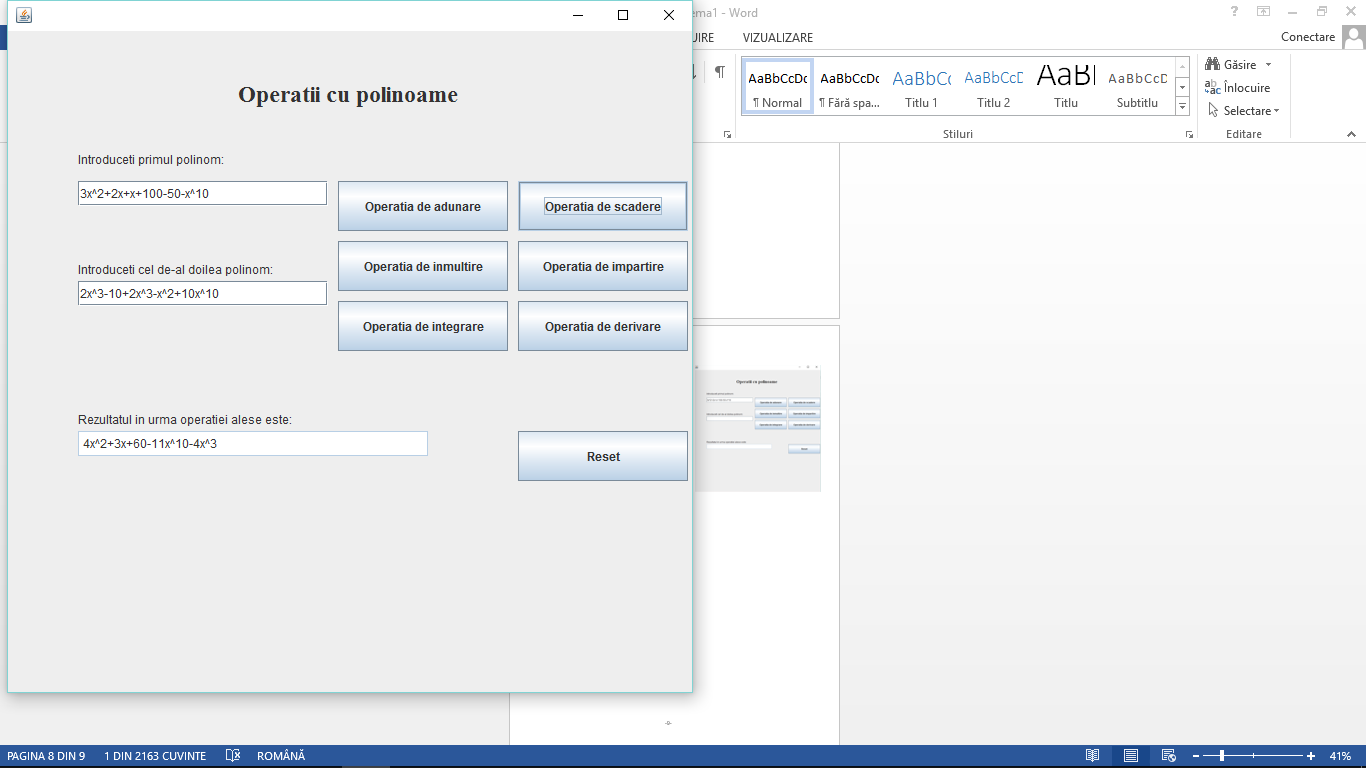
***●Diagrama de clase***

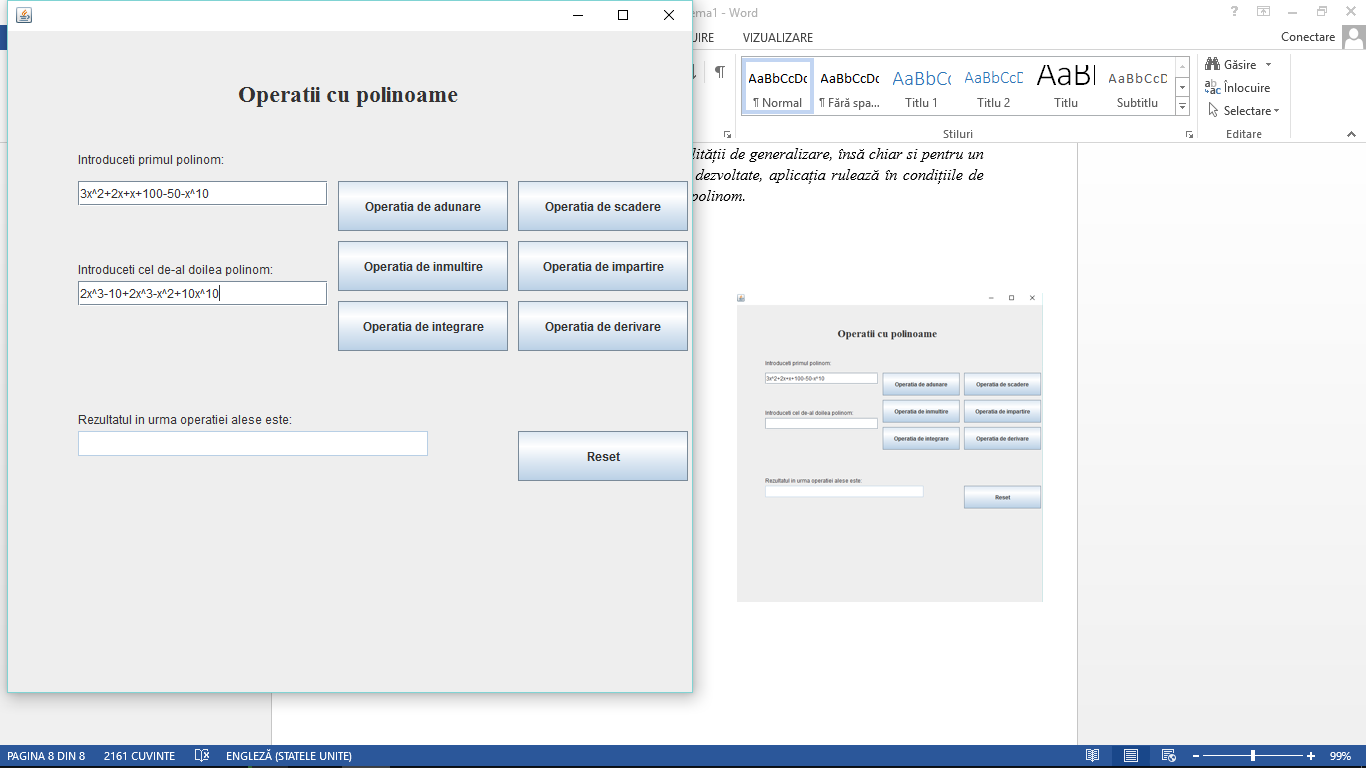
******

***●Diagrama USE CASE***

***Pașii de execuție a programului:***







***Concluzii***

*Concluziile pe care le putem trage în urma rulării si verificării corectitudinii programului creat se rezuma la faptul că pentru a prelucra niște date introduse de către un utilizator, care are niște cunoștințe de matematică corespunzătoare nivelului așteptat, este relativ simplă fiind însă destul de complexă din punctul de vedere al posibilității de generalizare, însă chiar si pentru un utilizator care nu are cunoștințe matematice bine dezvoltate, aplicația rulează în condițiile de respectare a unor standarde de introducere a unui polinom.*