

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE DEPARTAMENTUL CALCULATOARE**

DOCUMENTAŢIE TEMA 1

SISTEM DE PROCESARE A POLINOAMELOR

la disciplina

TEHNICI DE PROGRAMARE

CĂRBUNE IOANA-ELENA

Grupa 30221

An academic:2018-2019

Cuprins

[1. Obiectivul temei 3](#_Toc3762772)

[1.1. Obiectivul principal 3](#_Toc3762773)

[1.2. Obiective secundare 3](#_Toc3762774)

[2. Analiza problemei 3](#_Toc3762775)

[2.1. Use-case 3](#_Toc3762776)

[2.2. Descrierea paşilor 4](#_Toc3762777)

[3. Proiectare 4](#_Toc3762778)

[3.1. Decizii de proiectare 4](#_Toc3762779)

[3.2. Diagramă pachete 4](#_Toc3762780)

[3.3. Diagrama de clase ( nivel conceptual ) 5](#_Toc3762781)

[3.4. Structuri de date 5](#_Toc3762782)

[3.5. Proiectare clase 5](#_Toc3762783)

[3.6. Relaţii 6](#_Toc3762784)

[3.6. Algoritmi 6](#_Toc3762785)

[3.7. Interfaţa utilizator 6](#_Toc3762786)

[4. Implementare 6](#_Toc3762787)

[4.1. Implementare clase 6](#_Toc3762788)

[4.1.1. Monom 6](#_Toc3762789)

[4.1.2. Polinom 7](#_Toc3762790)

[4.1.3. MainView 7](#_Toc3762791)

[4.1.4. MainController 8](#_Toc3762792)

[4.1.5. MessageController 8](#_Toc3762793)

[4.1.6. App 8](#_Toc3762794)

[4.2. Implementarea interfeţei utilizator 8](#_Toc3762795)

[5. Rezultate 9](#_Toc3762796)

[6. Concluzii 10](#_Toc3762797)

[7. Bibliografie 10](#_Toc3762798)

# 1. Obiectivul temei

## 1.1. Obiectivul principal

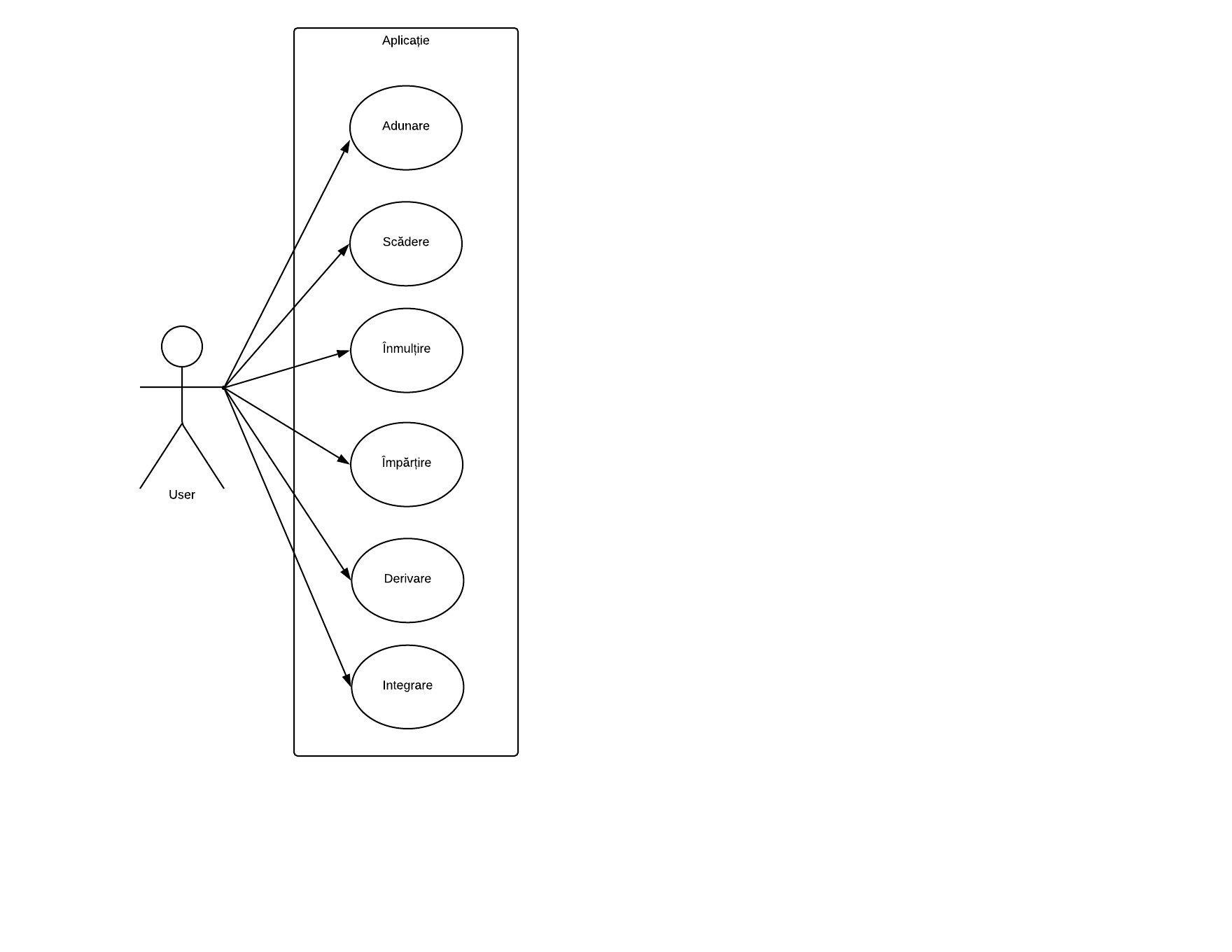
Obiectivul principal al temei este de a dezvolta un sistem de procesare al polinoamelor cu o singură variabilă şi cu coeficienţi întregi cu scopul de a realiza operaţii cu acestea. Se urmăreşte realizarea operaţiilor de adunare, scădere, înmulţire, împărţire, intregrare şi derivare.

## 1.2. Obiective secundare

|  |  |  |
| --- | --- | --- |
| Obiective secundare | Descriere | Capitol |
| Descriere use-case | Constă în descrierea sumară a funcţionalităţii aplicaţiei. | 2 |
| Structuri de date | Descrierea şi argumentarea structurilor de date alese. | 3 |
| Proiectarea claselor | Se prezintă clasele realizate. | 3 |
| Dezvoltare de algoritmi | Se prezintă algoritmii implementaţi în scopul realizării cerinţelor impuse. | 3 |
| Implementare interfaţă utilizator | Constă în prezentarea interfeţei GUI. | 4 |
| Testare | Reprezintă o serie de teste JUnit ce constau în posibile scenarii. | 5 |

2. Analiza problemei

2.1. Use-case

Se observă posibilele modalităţi de utilizare ale aplicaţiei.

## 2.2. Descrierea paşilor

Un prim pas primordial în realizarea operaţiilor este introducerea de către utilizator a polinoamelor în TextField-uri.

După alegerea operaţiei se realizează următorii paşi comuni:

* Preluarea textului din TextField-uri;
* Validarea textului cu scopul de a-l transpune într-o structură de tip polinom. Acest pas presupune la rândul său o serie de subpaşi.

1. Separarea textului în posibile monoame;
2. Verificarea corectitudinii formatului fiecărui monom în parte;

( semn coefficient x ^ exponent );

1. În cazul validării totale a monoamelor va rezulta un polinom, în caz contrar se va observa un mesaj de eroare.

* Transmiterea polinoamelor sau polinomului, după caz, în metodele aferente operaţiilor alese;
* Calcularea rezultatului;
* Afişarea rezultatului;

În cazul operaţiilor de derivare şi integrare este necesară selecţia de către utilizator a unuia dintre cele două polinoame. Ca urmare field-ul rămas neutilizat se va bloca pentru a preveni posibile confuzii sau erori.

În cazul operaţiei de împărţire se fac unele verificări suplimentare pentru a atenţiona utilizatorul cu privinţă la împărţirea unui număr la 0 ( operaţie ilegală ).

3. Proiectare

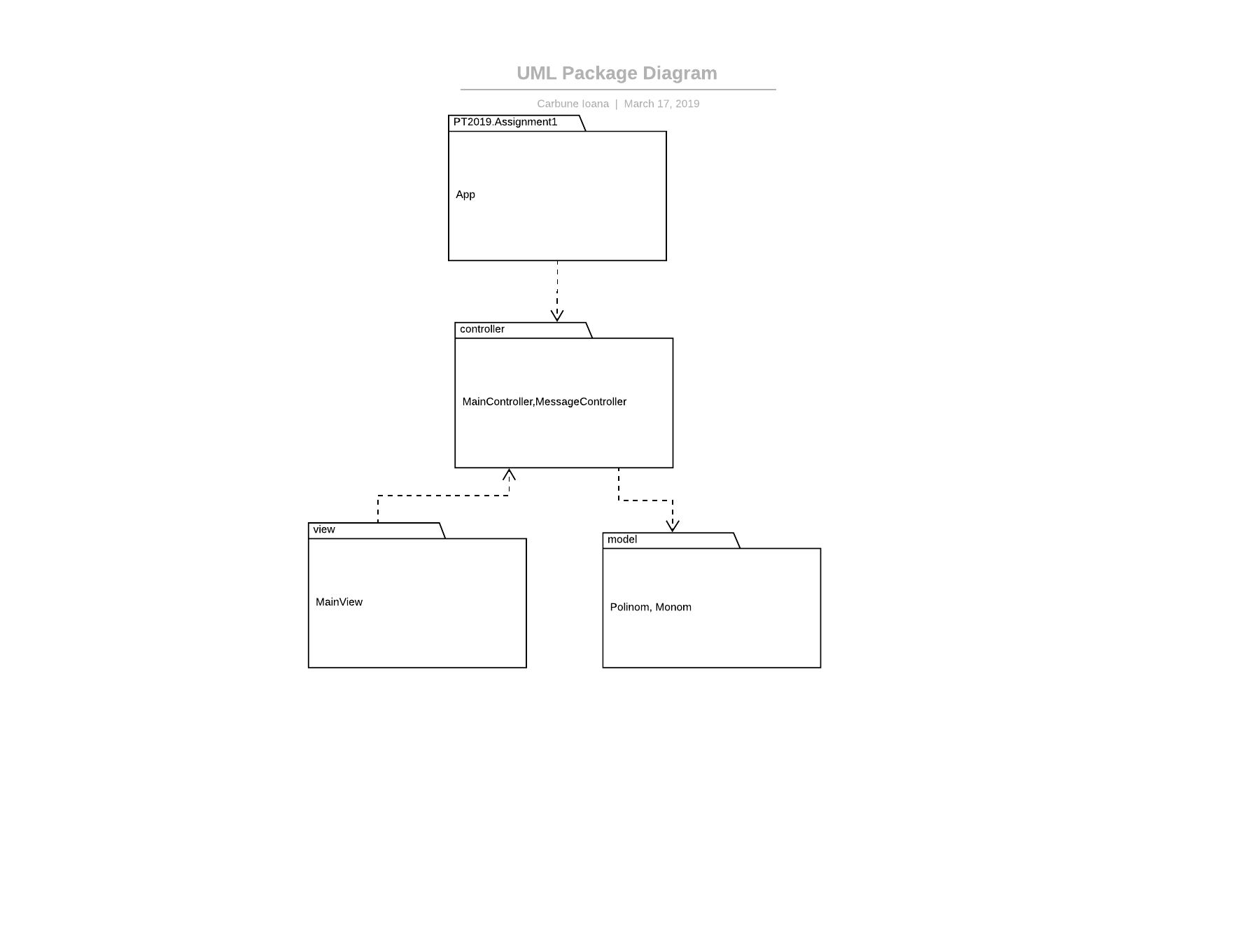
## 3.1. Decizii de proiectare

O decizie majoră de proiectare luată în realizarea acestei aplicaţii este cea de a urma arhitectura MVC ( Model-View-Controller) cu scopul de a fi totul bine organizat şi structurat. Prin urmare, proiectul este structurat pe mai multe pachete în funcţie de rolul fiecărei clase. Din punct de vedere al relevanţei pentru această aplicaţie, cele mai importante clase sunt Monom şi Polinom.

## 

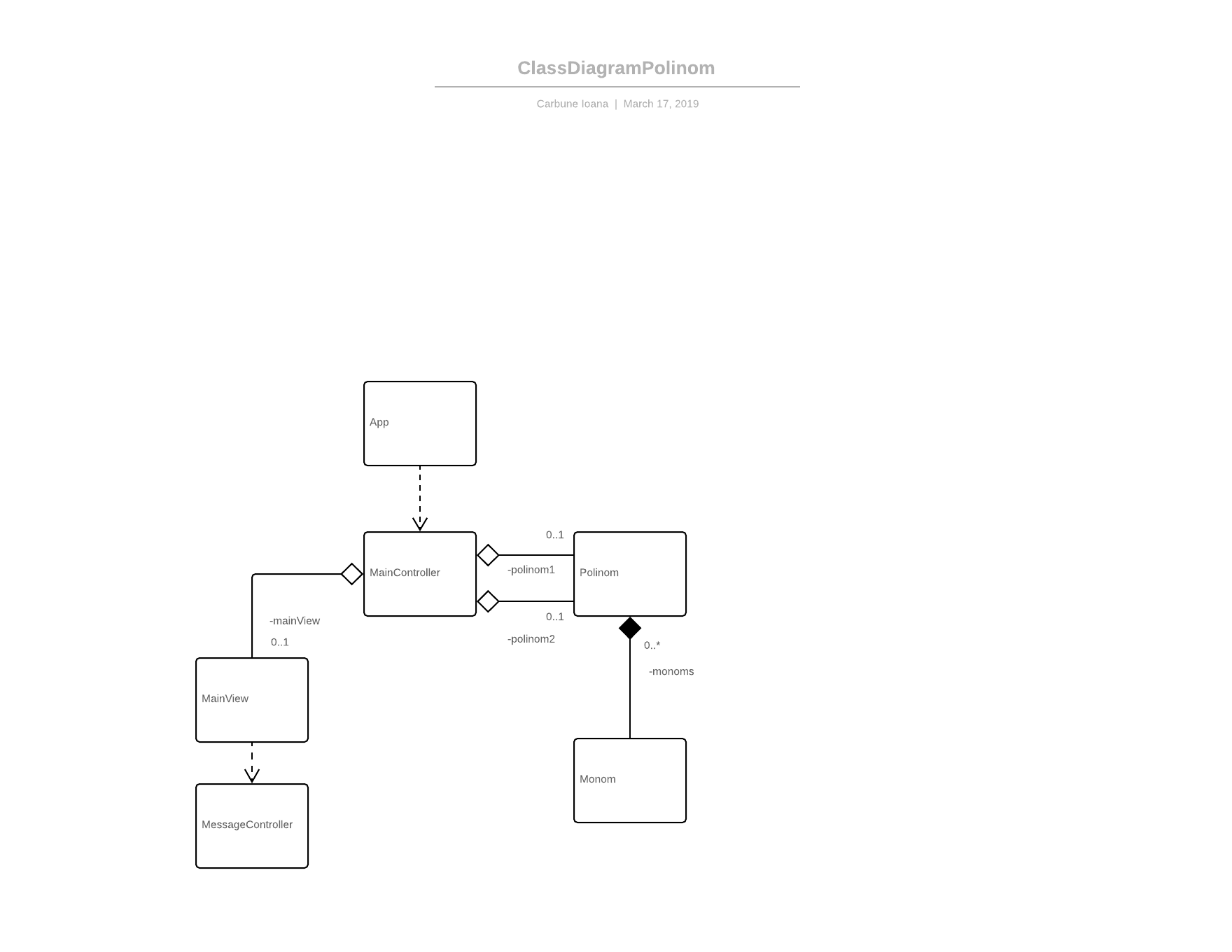
## 3.2. Diagramă pachete

În diagrama de mai jos se poate observa modul în care aplicaţia este structurată pe pachete, relaţiile dintre acestea şi clasele componente.



## 3.3. Diagrama de clase ( nivel conceptual )

În această diagramă se pot observa clasele şi relaţiile dintre acestea, care urmează a fi descrise într-un capitol următor.



## 3.4. Structuri de date

În mod firesc un polinom este format din mai multe monoame, mai exact o listă. Pentru a uşura calculele şi transpunerea textului furnizat de utilizator în polinoame, am ales ca lista de monoame să fie de tipul ArrayList <Monom>. Structura de acest tip pune la dispoziţie manipularea mai uşoară a memoriei. De exemplu, utilizatorul poate introduce un polinom cu 1 sau 100 de monoame, dezvoltatorului aplicaţiei îi este imposibil de dedus în stadiul implementării ce dimensiune o să aibă un anumit polinom. Un array de monoame, de exemplu, ar fi fost extrem de ineficient în această situaţie.

## 3.5. Proiectare clase

Aplicaţia este formată din următoarele clase:

* Monom;
* Polinom;
* MainView;
* MainController;
* MessageController;
* App;

În proiectarea claselor am apelat la încapsulare. Acestea conţin atribute private care pot fi accesate din alte clase prin metodele aferente fiecăreia, acestea fiind publice.

## 3.6. Relaţii

După cum s-a putut observa în diagrama de mai sus între clasele aplicaţiei există mai multe relaţii. Între Polinom şi Monom este o relaţie de compoziţie, datorită faptului ca un polinom este format din 1 sau mai multe monoame. Între MainController şi Polinom,respectiv MainView există o relaţie de agregare, ultimele doua fiind instanţiate în MainController. Celelalte relaţii sunt de dependenţă între App şi MainController, respective MainView şi MessageController.

## 3.7. Algoritmi

Algortimii utilizaţi sunt cei ştiuţi din matematică pentru calculele cu polinoame: adunarea, scăderea, înmulţirea sau împărţirea a două polinome şi integrarea sau derivarea unui singur polinom. Implementarea lor va fi detaliată într-un capitol următor.

## 3.8. Interfaţa utilizator

Interfaţa utilizator este proiectată în felul următor: utilizatorul dispune de posibiliatea de a introduce două texte corespunzătoare celor două polinoame, sub acestea se află câte un buton pentru fiecare operaţie, mai exact 6 butoane. Datorită faptului că unele operaţii se execută pe un singur polinom, există şi opţiunea de a selecta polinomul dorit, în partea dreaptă a butoanelor pentru operaţii. După introducerea textului şi alegerea operaţiei dorite există mai mult posibilităţi. În cazul în care textul introdus respectă formatul cerut, deasupra fiecăruia va apărea un mesaj de succes şi în josul ferestrei rezultatul sub formă de text needitabil, în caz contrar se va afişa un mesaj de eroare. Interfaţa dispune şi de un buton de reset care acţionează asupra textelor introduse.

# 4. Implementare

### 4.1. Implementare clase

### 4.1.1. Monom

* Atributele clasei Monom sunt coeficientul întreg şi exponentul double.
* O metodă importantă este validateMonom ( String ): Boolean. Aceasta este apelată de către validatePolynomial din clasa Polinom pentru fiecare monom în parte. Dacă un singur monom nu respect formatul << semn coefficient x ^ exponent >> va returna valoarea false şi atunci se consideră că întregul polinom are un format incorect.
* De asemenea importantă este şi supraîncarcarea constructorului. Am ales să folosesc doi constructori care se diferenţiază prin parametrii lor. Primul este folosit la crearea monoamelor în momentul citirii lor de la tastatură şi are ca paramatru un string. Acesta este separat în două substring-uri care vor reprezenta mai apoi exponentul şi coeficientul. Al doilea constructor folosit este cel generat automat care are ca şi parametri un întreg şi un double, acesta fiind folosit în metodele ce reprezintă operaţiile pe polinoame.

### 4.1.2. Polinom

* Precum s-a specificat mai sus, clasa Polinom conţine o listă de monoame.
* Cele mai importante metode sunt cele care reprezintă operaţiile, scrise prin transpunerea algoritmilior matematici în cod, acestea returnând mai apoi rezultatul sub formă de string.
  + Adunarea-> addition ( Polinom ): String la primul polinom se adaugă rând pe rând câte un monom din cel de-al doilea polinom, adunându-se coeficienţii monoamelor cu acelaşi exponent.
  + Scăderea -> subtraction ( Polinom ): String se urmează aceiaşi paşi ca în cazul adunării cu excepţia faptului că se adaugă monoamele cu -coeficient.
  + Înmulţirea -> multiplication ( Polinom ): String se ia fiecare monom din primu polinom şi se înmulţeşte cu fiecare monom din al doilea polinom.
  + Împărţirea -> division ( Polinom ): String dacă primul polinom e 0 rezultatul va fi 0, dacă al doilea polinom e 0 se va returna un mesaj de eroare datorită faptului că împărţirea la 0 nu este posibilă, în celelalte cazuri împărţirea se face normal. Atâta timp cât gradul primului polinom nu e mai mic decât al celui de-al doilea se ia primul termen din ambele, se face împărţirea, rezultatul se înmulţeşte cu împărţitorul, se scade din deîmpărţitor şi se repetă paşii.
  + Derivarea -> derivative (): String conform algortimului de derivare din cunoscut de la matematică se parcurge monom cu monom, se înmulţeşte coeficientul curent cu exponentul iar exponentul scade cu 1.
  + Integrarea -> integration(): String se parcurge monom cu monom, coeficientul current se împarte la exponent +1, iar noul exponent va fi vechiul exponent +1.
* Metoda validatePolynomial ( String ): boolean testeaza stringul primit ca paramentru dacă respectă formatul de scriere a unui polinom. În caz afirmativ construieşte polinomul şi returnează true, în caz negativ va returna valoarea false fără a reţine stringul primit în polinom.
* De asemenea o importanţă mare are şi polynomialToString (): String, care stabileşte în ce mod se va afişa un polinom ( în scopul evitării rezultatelor de genul 0x+1 ). Dacă polinomul începe cu un monom cu coeficient pozitiv nu se va mai afişa semnul. Dacă polinomul, va avea mai multe monoame cu coeficientul 0 nu se vor afişa decât în cazul în care întregul polinom=0;
* Metoda addMonoms ( Monom ) : void are rolul de a adăuga monomul primit ca parametru la lista polinomului apelant. În cazul în care există deja monoame cu acelaşi exponent nu se va adăuga un monom nou, în schimb se va modifica coeficientul celui vechi. În caz contrar se va adăuga monom nou, cu condiţia ca acesta să nu fie 0.
* Nu în ultimul rând, sortPolynomial(): void are rolul de a aduce polinomul cu care este apelată la forma sa canonică şi anume în cazul unui polinom de genul 1-x+2x^2 acesta va deveni 2x^2-x+1 după apelarea metodei.

### 4.1.3. MainView

* Clasa MainView conţine tot ce ţine de interfaţa grafică: 6 x Button pentru selectarea operaţiilor, 2 TextField-uri pentru introducerea polinoamelor, 2 label-uri pentru mesajele de eroare, 2 RadioButtons pentru selectarea unui singur polinom, un TextField needitabil pentru afişarea rezultatului şi nu în ultimul rând un Button pentru reset. Acestea sunt structurate în funcţie de rolul lor în VBox-uri şi HBox-uri pentru o aliniere egală.
* Ca şi metode importante se pot aminti error1(String, Color ): void , respectiv error2(String, Color ): void - care afişează un mesaj de eroare (sau de succes) atunci când este cazul, cu culoarea solicitată (roşu sau verde) .
* Metodele de verficare a butoanelor radio radio1IsSelected() : boolean şi radio2IsSelected() : boolean ajută la identificarea TextField-ului din care se preia polinomul pentru derivare sau integrare.
* resetFields() resetează TextField-urile şi butoanele radio.

### 4.1.4. MainController

* start(Stage): void apelează metoda show din MainView care are rolul de a arăta pe ecran interfaţa grafică.
* initializeButtonListeners(): void apelează pentru fiecare buton în parte metodele de ActionListener din MainView cu scopul de a le atribui câte un rol.

### 4.1.5. MessageController

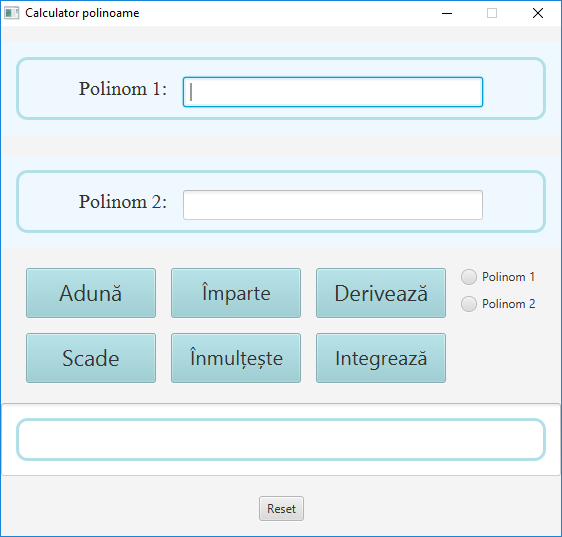
* Constă într-o singură metoda şi anume displayMessage(String, Label , Color ) : void care are rolul de a crea un fir nou de lucru cu scopul de a afişa în label-ul primit ca paramentru un mesaj timp de 3 secunde după care acesta va dispărea.

### 4.1.6. App

* Această clasă are rolul de a lansa aplicaţia.

## 4.2. Implementarea interfeţei utilizator

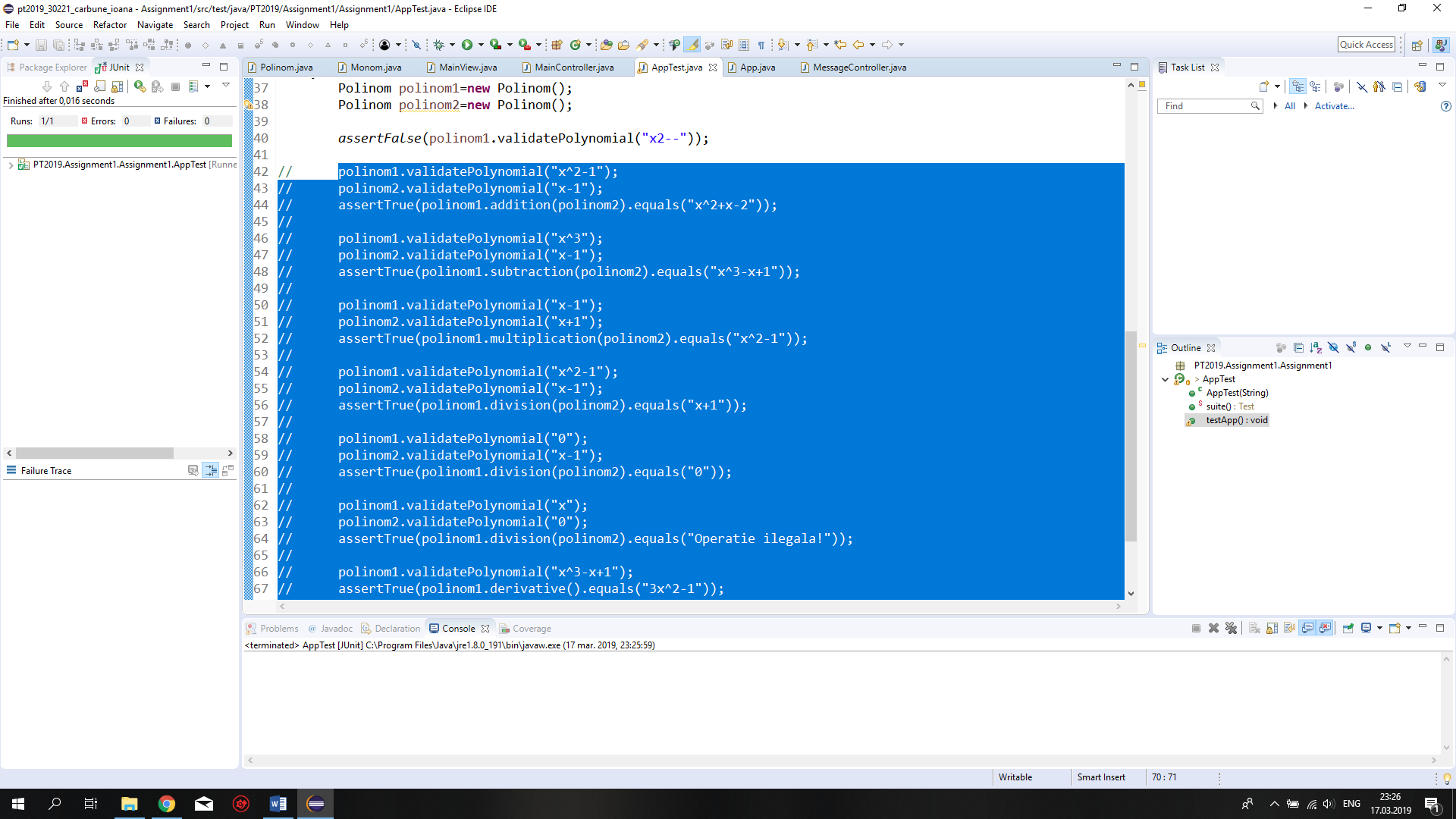
Pentru implementarea intefeţei utilizator am ales să utilizez JavaFx deoarce în opinia mea este mai accesibil decât alte opţiuni, de exemplu Swing. Cum a fost precizat mai sus ca şi elemente au fost folosite butoane, butoane radio, TextField-uri, lable-uri ,dar şi VBox-uri şi HBox-uri. Toate acestea au fost poziţionate într-un BorderPane care are rol de layout. De asemenea, fiecărui buton îi corespunde o metodă addXButtonActionListener(EventHandler<ActionEvent>) care are rolul de a le atribui o acţiune.



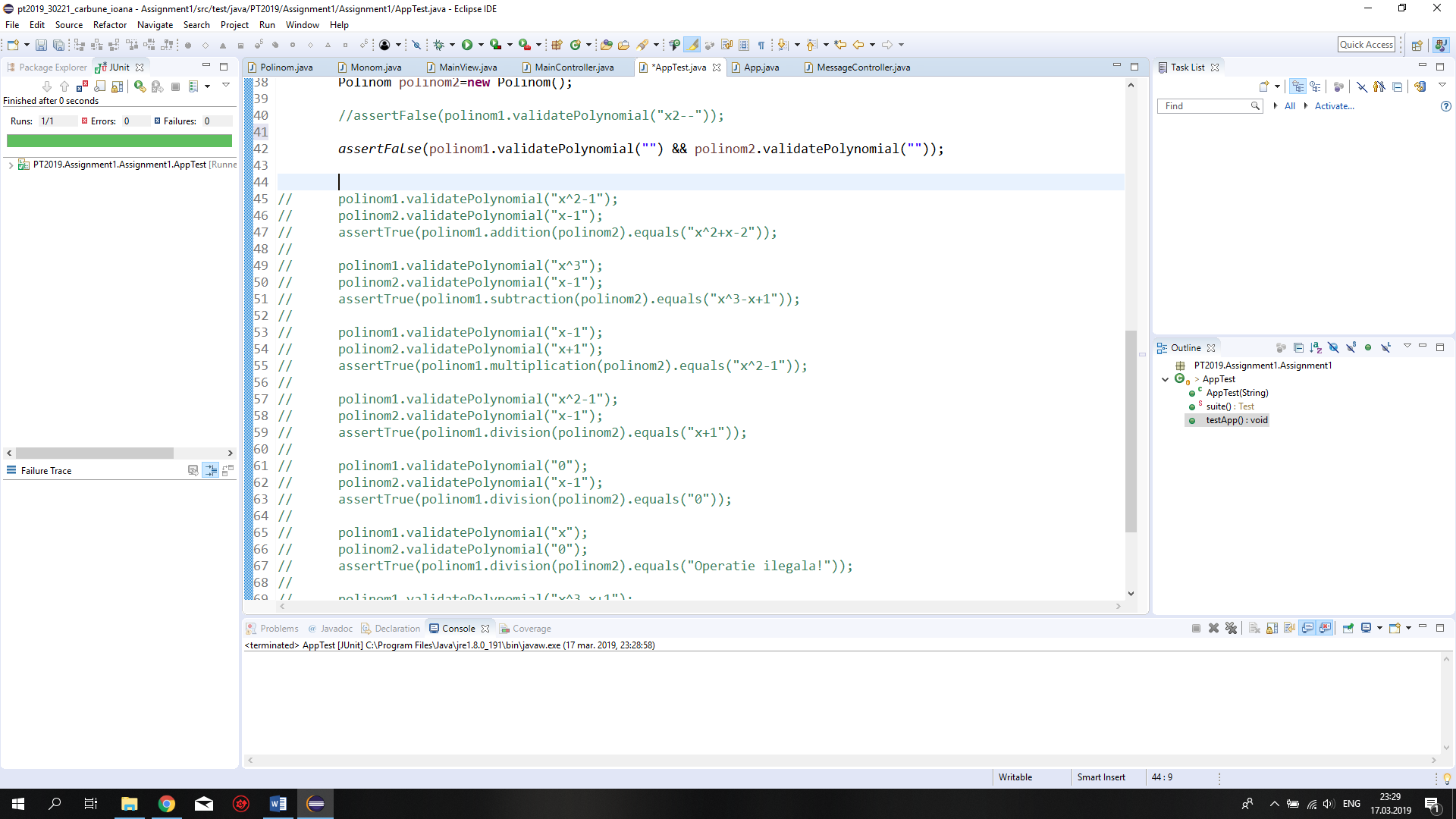
5. Rezultate

Cazurile care se pot întâlni în utilizarea aplicaţiei:

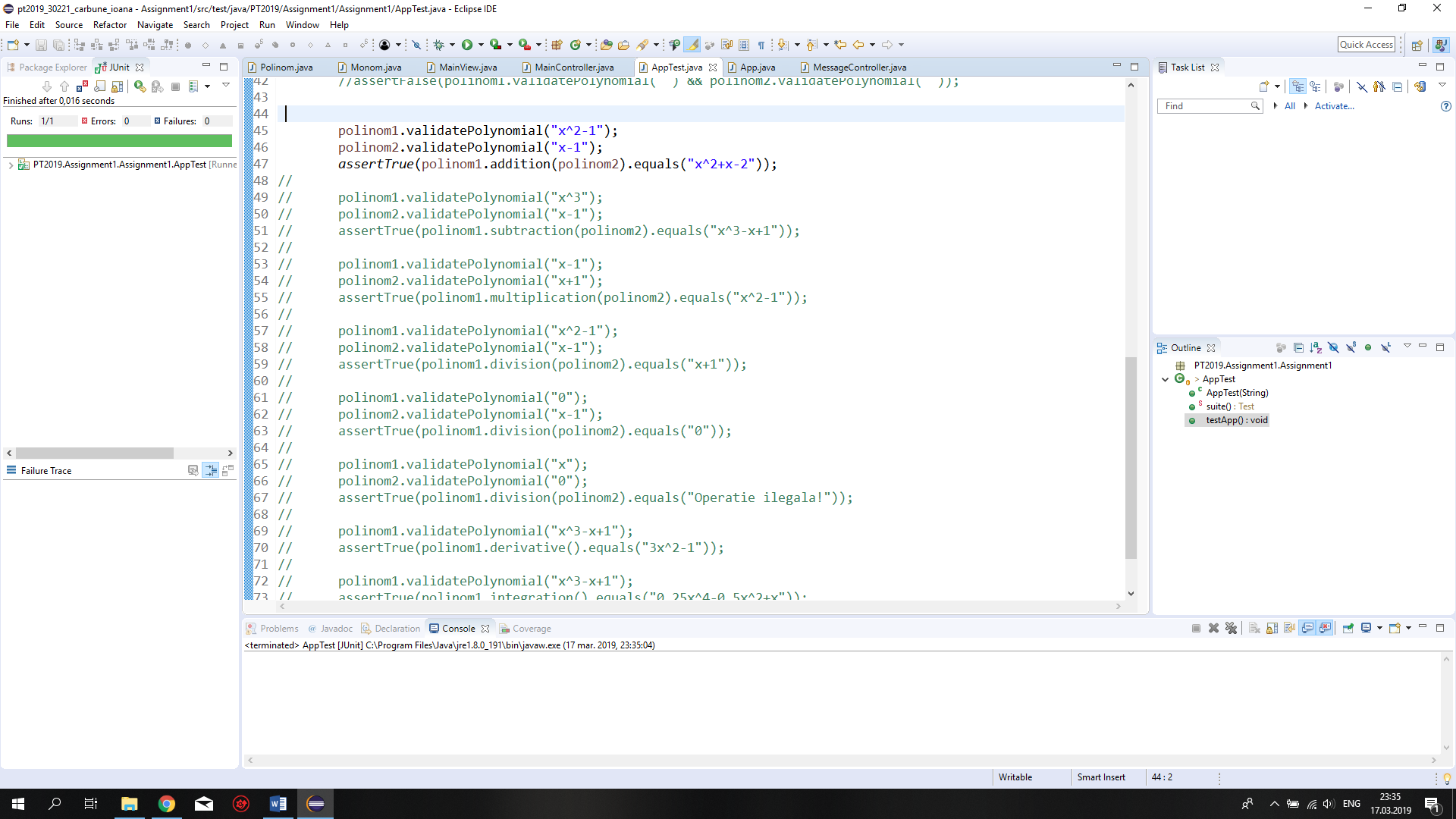
* Utilizatorul introduce un format gresit;



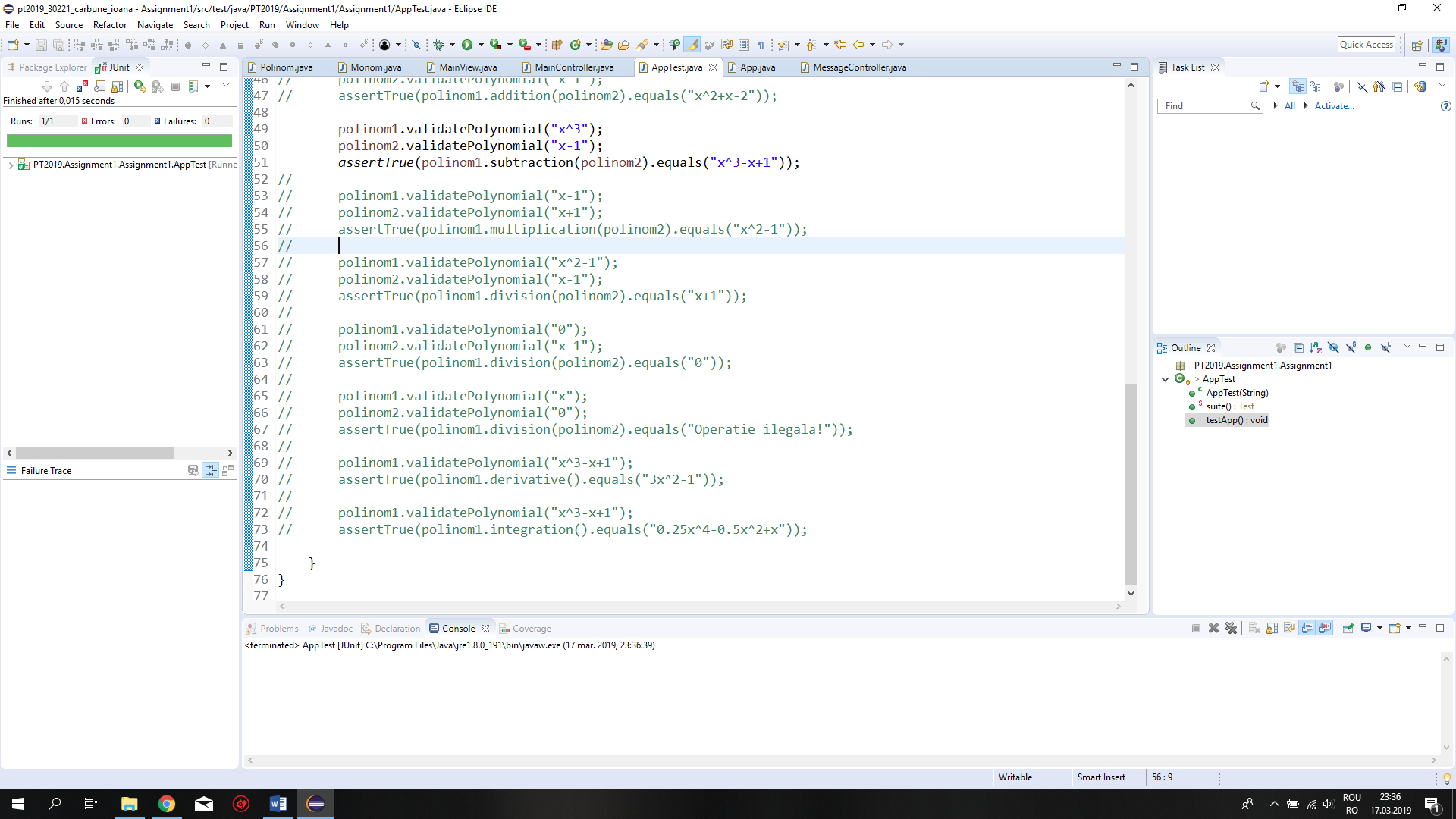
* Utilizatorul nu introduce nimic şi solicită o operaţie;



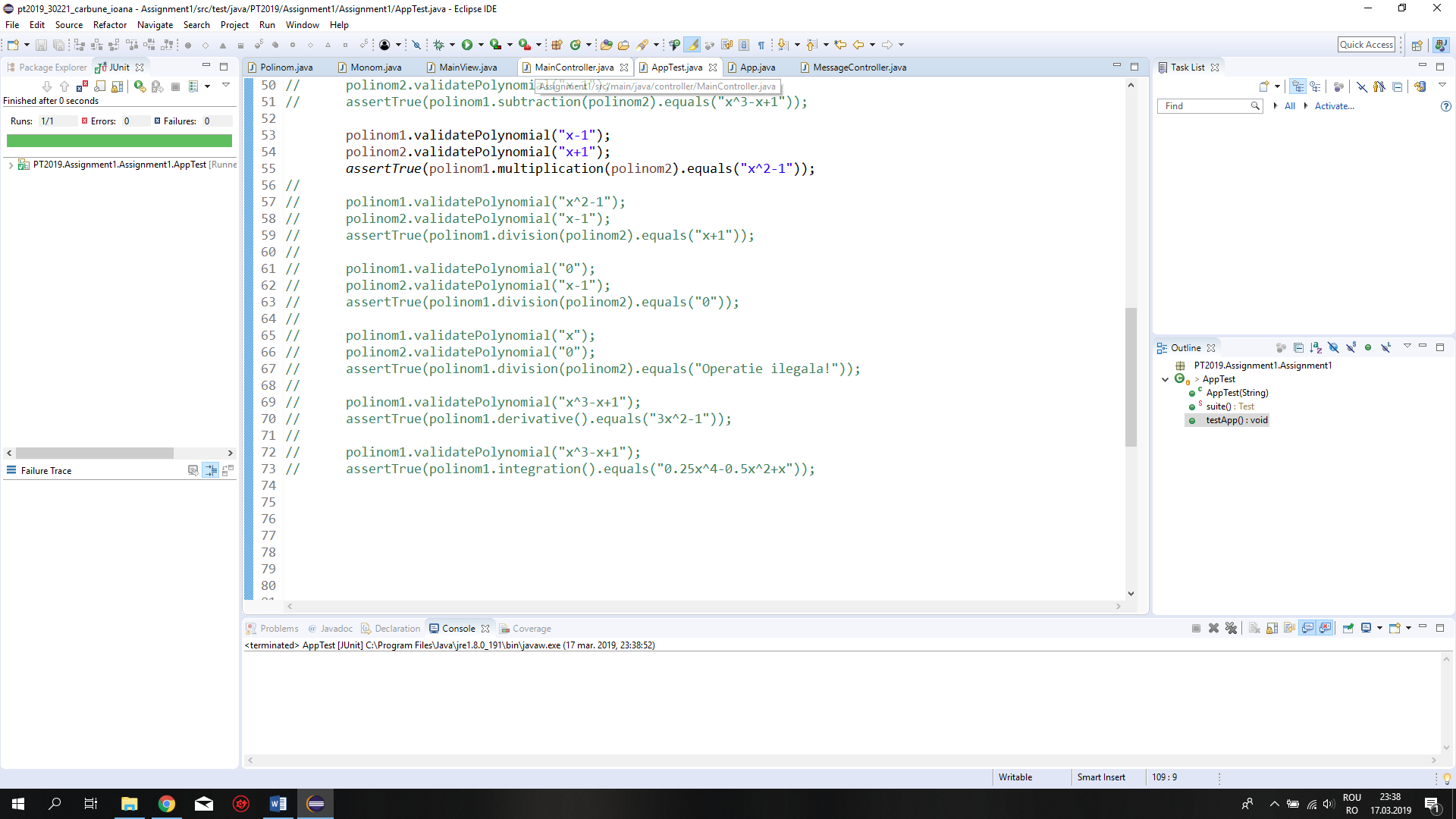
* Formatul este corect şi utilizatorul vrea să efectueze o adunare;



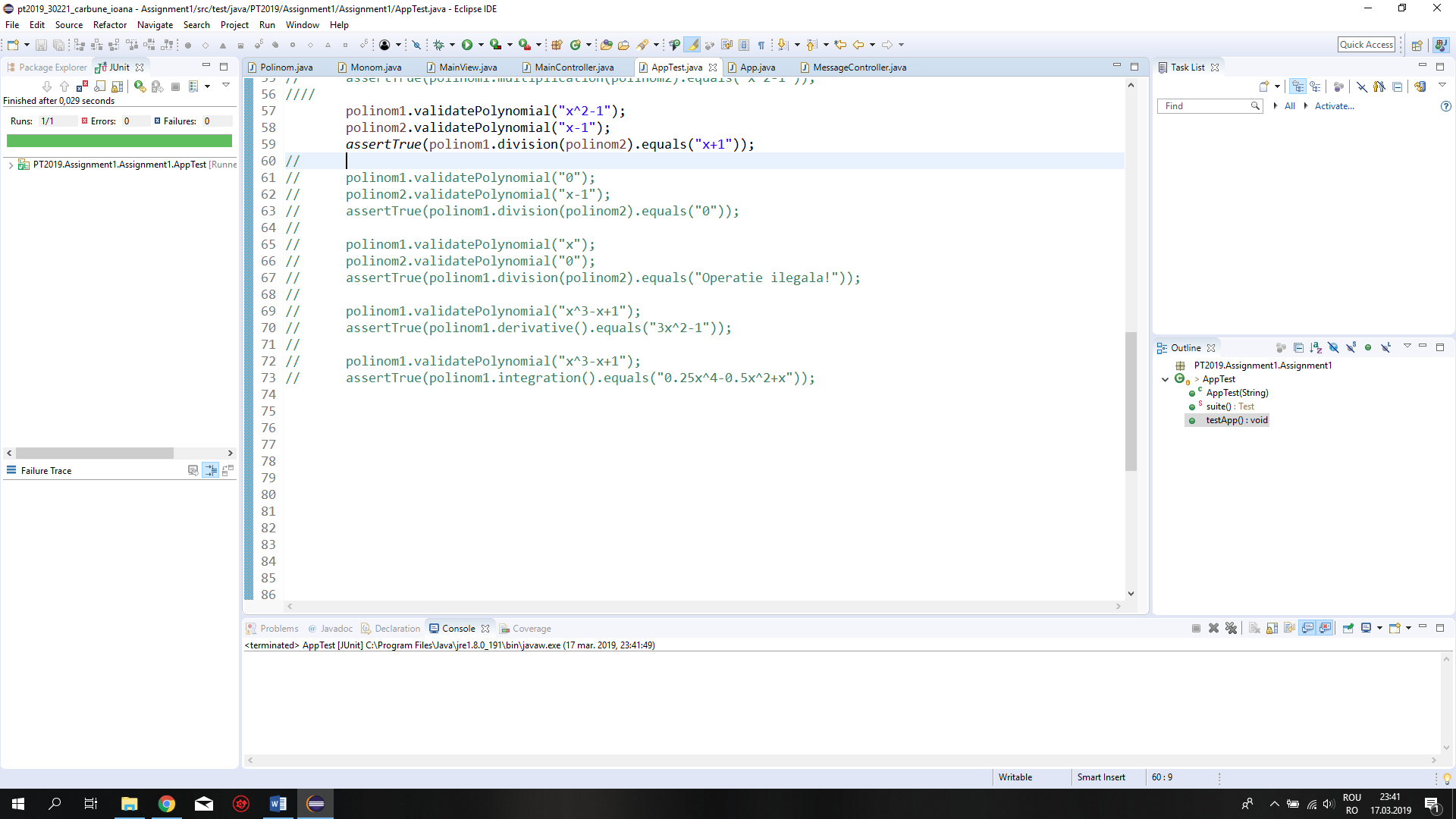
* Formatul este corect şi utilizatorul vrea să efectueze o scădere;



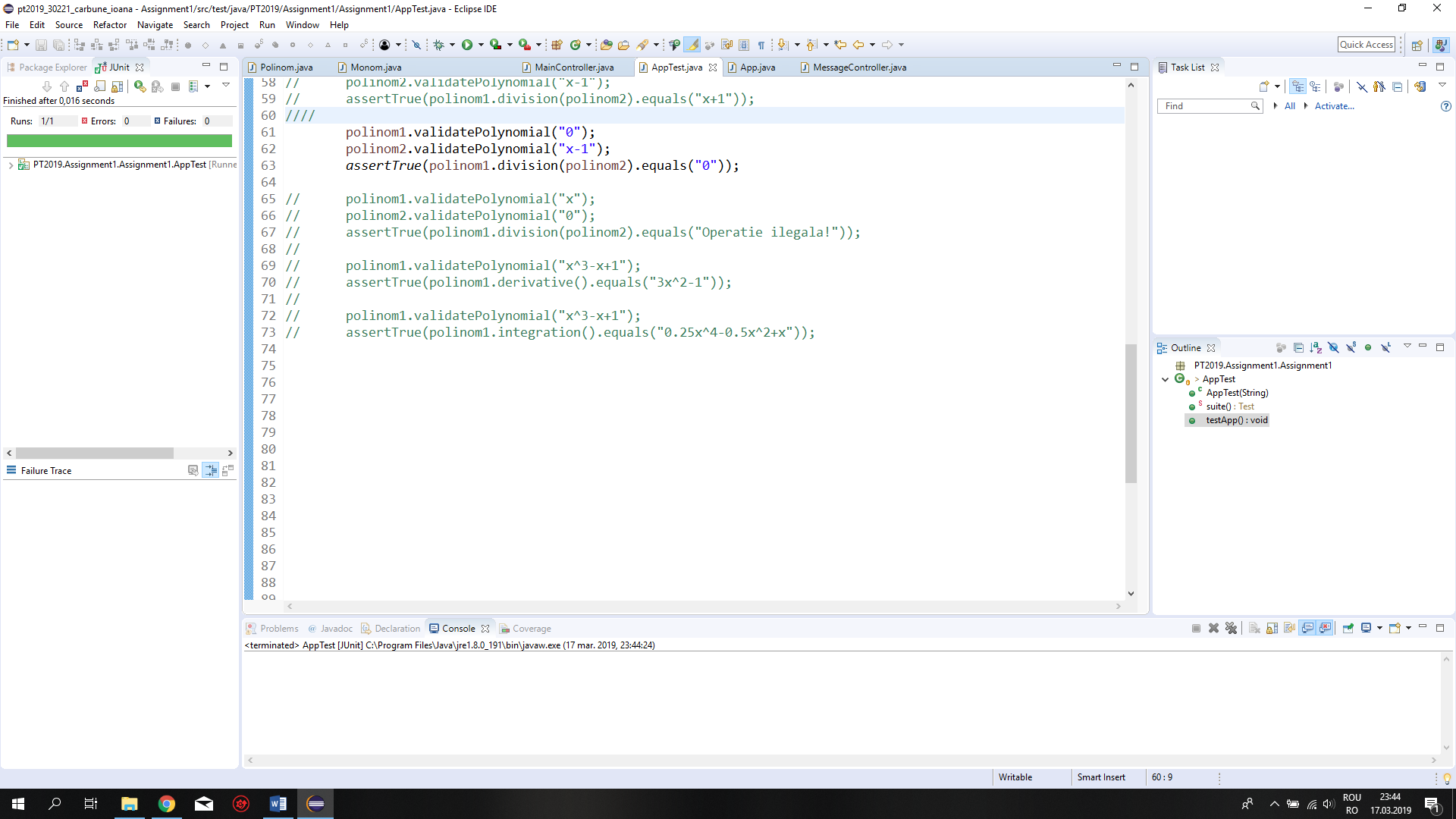
* Formatul este corect şi utilizatorul vrea să efectueze o înmulţire;



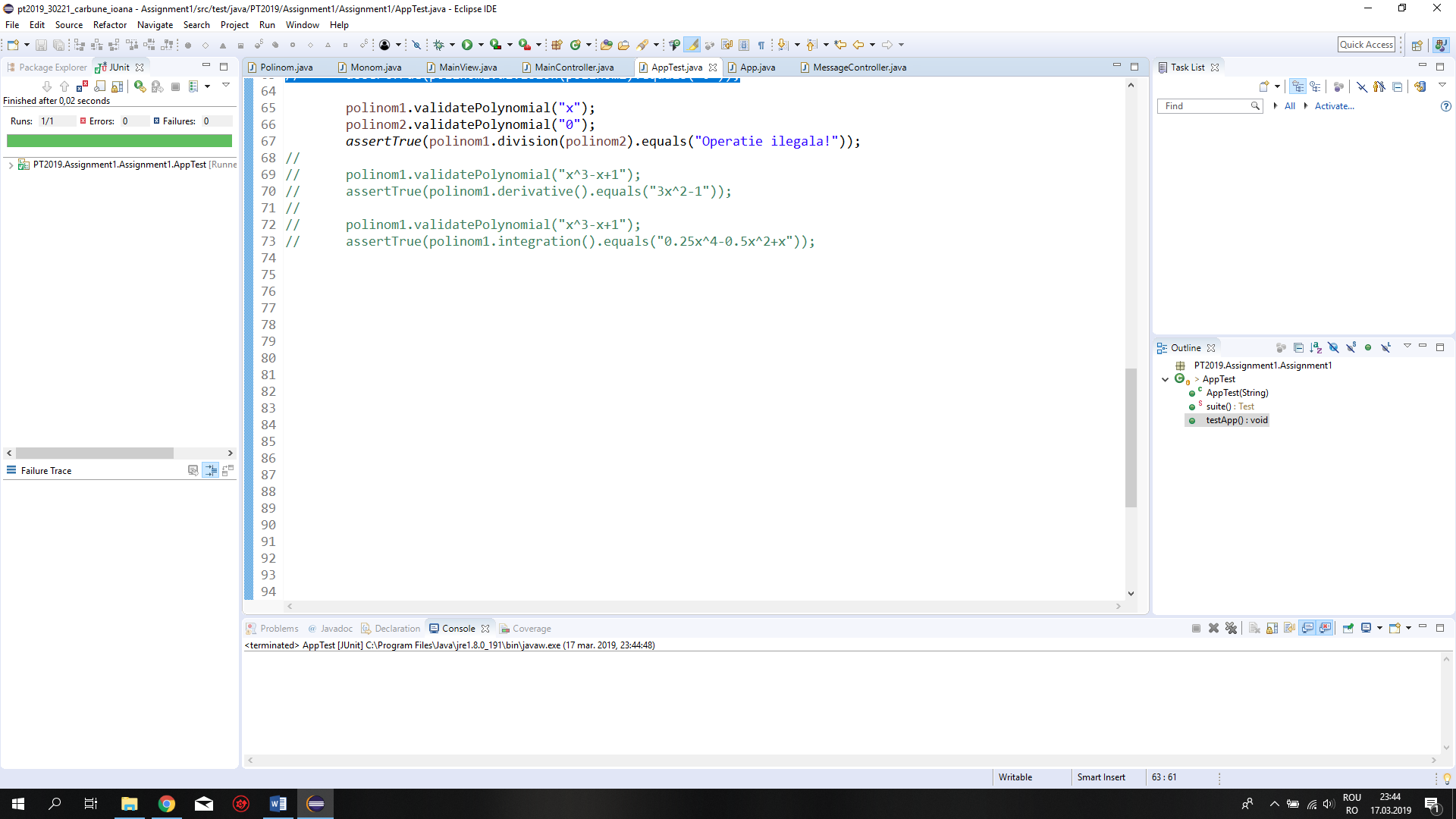
* Formatul este corect şi utilizatorul vrea să efectueze o împărţire;



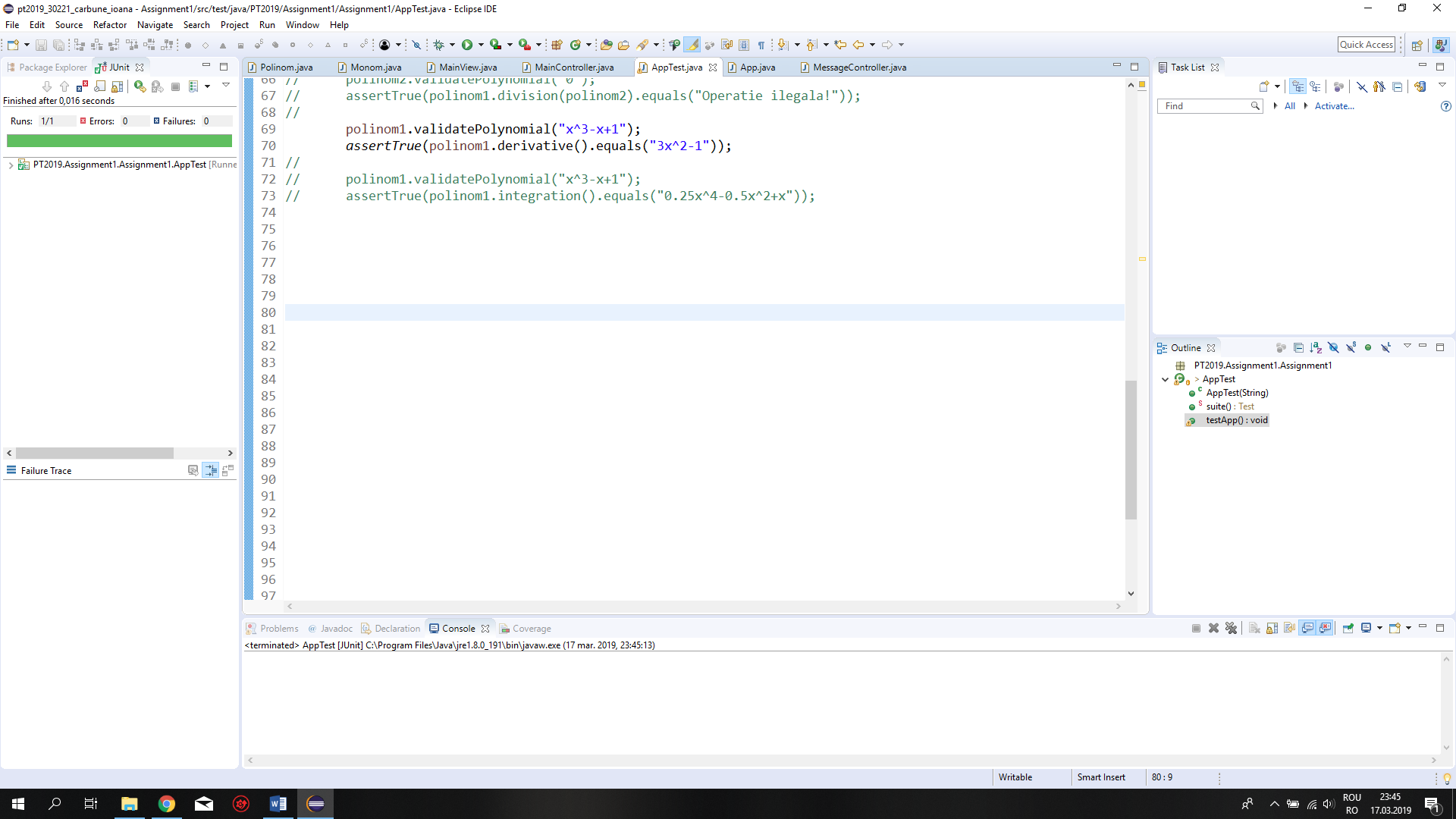
* Formatul este corect şi utilizatorul vrea să efectueze o împărţire a lui 0;



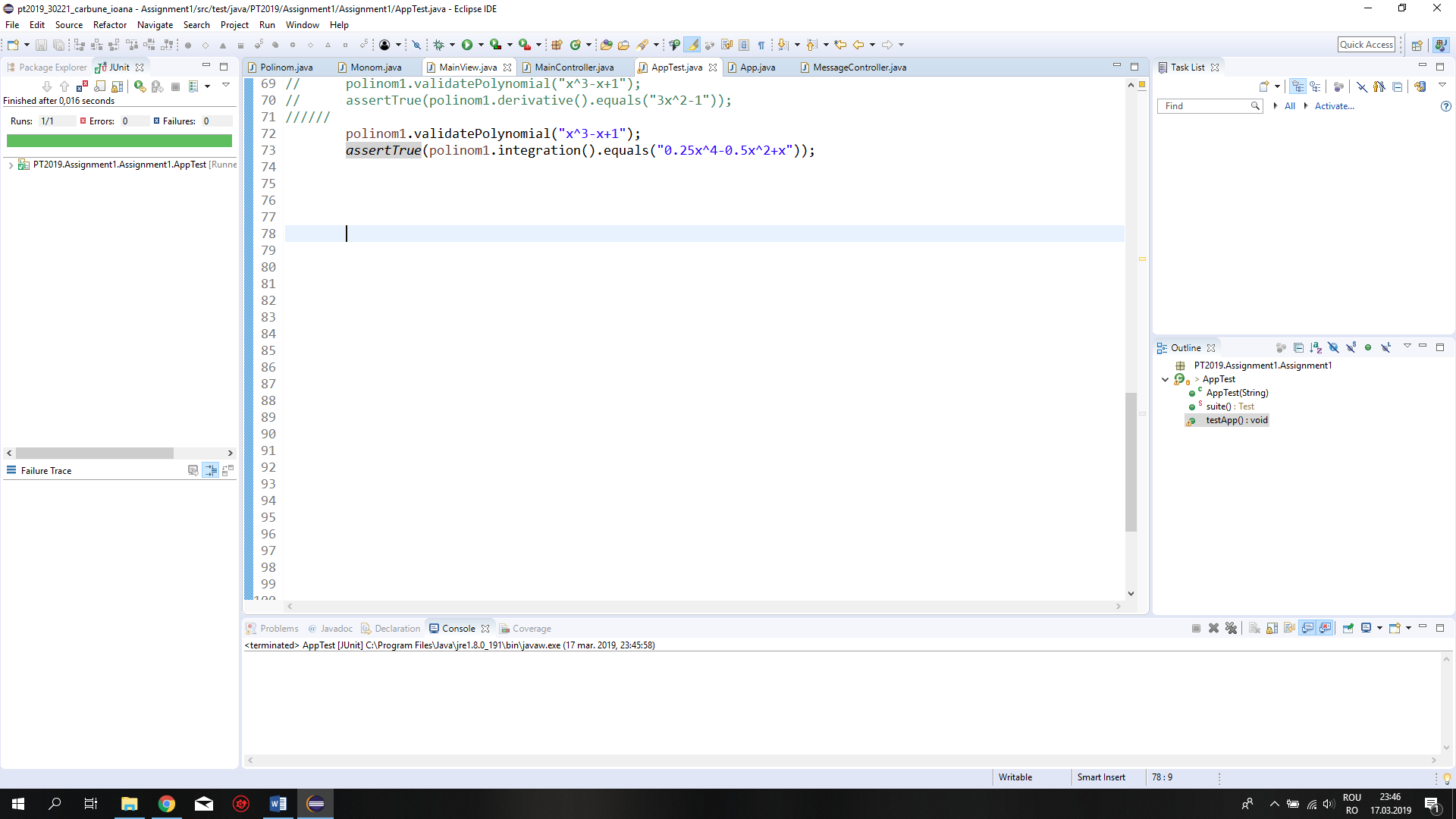
* Formatul este corect şi utilizatorul vrea să efectueze o împărţire la 0;



* Formatul este corect şi utilizatorul vrea să efectueze o derivare;



* Formatul este corect şi utilizatorul vrea să efectueze o integrare;



# 6. Concluzii

Datorită acestei teme am reuşit o mai bună familiarizare cu ideea de arhitectură MVC şi cu munca individuală.

Consider că o posibilă dezvoltare a temei poate fi adăugarea unei funcţii suplimentare de calculare a valorii unui polinom. Ca exemplu pentru un polinom x^2-1 şi un x=3 să se returneze valoarea 8.

O altă posibilă dezvoltare poate fi aflarea rădăcinilor reale ale unui polinom. Pentru acelaşi polinom de mai sus s-ar returna valorile 1 şi -1.

Totodată, se poate considera o posibilă dezvoltare şi adăugarea unui algoritm de aflare a celui mai mare divizor comun a două polinoame.

# 7. Bibliografie

* Pentru afişarea mesajelor de eroare:

<https://stackoverflow.com/questions/19968012/javafx-update-ui-label-asynchronously-with-messages-while-application-different>

* Pentru algoritmii operaţiilor pe polinoame:

<https://www.youtube.com/watch?v=FsotIB0Usvw&t=66s>

<https://en.wikipedia.org/wiki/Polynomial#Arithmetic>

* Pentru metode JavaFX:

<https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm>