**DOCUMENTATIE TEMA 1**

**CALCULATOR DE POLINOAME**

**Student: Cibu Mihai**

**Grupa: 30226**

**Profesor Laborator: Marin Oana Andreea**

**Cuprins:**

1. Obiective
   1. Obiectiv principal
   2. Obiective secundare
2. Analiza Problemei, modelare, scenarii, cazuri de utilizare
   1. Analiza problemei, modelare
   2. Scenarii
   3. Cazuri de utilizare

3. Proiectare

4. Implementare

5. Rezultate

6. Concluzii si Dezvoltari Ulterioare

7. Bibliografie

**1.Obiective**

**1.1 Obiectiv principal**

Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficenti intregi.

Aplicatia presupune folosirea unei interfete grafice pentru interactiunea cu utilizatorul, care poate introduce polinoamele (/polinomul) sub forma de String, selecteaza operatia dorita (sunt permise operatiile: adunare, scadere, inmultire, impartire, derivare si integrare), rezultatul fiind afisat tot in interata grafica.

De asemenea, aplicatia trebuie sa respecte paradigmele Programarii Orientate-Obiect, sa fie structurata pe pachete, clasele si metodele trebuie sa respecte numarul de linii impus (maxim 300, respectiv 30) si sa se respecte conventiile Java legate de denumirea claselor, metodelor si variabilelor.

**1.2 Obiective secundare**

|  |  |  |
| --- | --- | --- |
| **Obiectiv secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii | Prezentarea modalitatilor de folosire si functionare a calculatorului de polinoame in cazurile posibile plus situatiile in care utilizatorul greseste la introducerea datelor sau introduce date invalide | **2** |
| Alegerea de structuri de date | Selectarea structurilor de date abstracte folosite pentru stocarea datelor | **3** |
| Impartirea pe clase | Structurarea in clase si pachete | **4** |
| Dezvoltarea algoritmilor | Implementarea propriu-zisa a operatiilor matematice in cod Java | **4** |
| Implementarea solutiei | Prezemtarea pe larg a solutiei si a metodelor folosite | **4** |
| Testare | Testarea metodelor principale folosite in codul sursa pentru verificarea corectitudinii folosind Junit | **5** |

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

**2.1 Analiza problemei, modelare**

Pentru asigurarea corectitudinii datelor introduse se vor face mai multe verificarii pe datele introduse de utilizartor. Pentru ca aplicatia sa functioneze corect, polinoamele trebuie introduse sub forma: ”coeficient x^ putere +/- coeficient x^ putere ...” (de exemplu 3x^2 -4x +2), spatiile albe dintre termeni/coeficienti fiind ignorate la prelucrarea Stringului. Totodata este important ca doar litera „x” sa fie folosita la inttroducerea polinomului (polinom de o singura variabila), altfel se va semnala un mesaj de eroare. Un aspect sesizabil la afisarea rezultatului este faptul ca coefiecientii sunt de tip double. Am facut aceasta alegere deoarece la operatiile de impartire si integrare sunt cazuri in care coeficientii sunt reali si pentru efectuarea corecta a operatiilor s-a facut aceasta conventie, dar aceasta nu influenteaza cu nimic celelalte operatii.

**2.2 Scenarii**

Pe parcursul implementarii am verificat mai multe scenarii posibile, atat favorabile cat si nefavorabile. Scenariile favorabile se refera la introducerea corecta a polinoamelor (dupa formatul prezentat mai sus) si observarea rezultatului in campul corespunzator. In ceea ce priveste scenariile nefavorabile s-au facut verificari legat de introducerea corecta a polinoamelor dupa formatul corespunzator, introducerea de date in ambele campuri daca se doreste efectuarea unei operatii ce necesita doi operanzi, introducerea polinomului in primul camp pentru operatiile de derivare si integrare si introducerea unui polinom nenul in campul al doilea pentru impartire.

**2. 3Cazuri de utilizare:**

Use case: Adunarea a doua polinoame.

Actor: Utilizatorul care doreste sa efectueze operatii pe polinoame.

Preconditii: Utilizatorul introduce in campurile specificate din interfata grafica cele doua polinoame pentru care doreste sa efectueze operatia de adunare si selecteaza operatia de adunare prin click asupra butonului „Adunare”.

Postconditii: Utilizatorul o sa observe in campul rezultat al interfetei grafice rezultatul operatiei.

Flow: Datele introduse de utilizator vor fi preluate, verificare, procesate si in cazul in care se indeplinesc conditiile respective se va calcula suma celor doua polinoame.

(Acelasi procedeu se va efectua si pentru operatiile de scadere, inmultire si impartire).

Use case: Derivarea unui polinom.

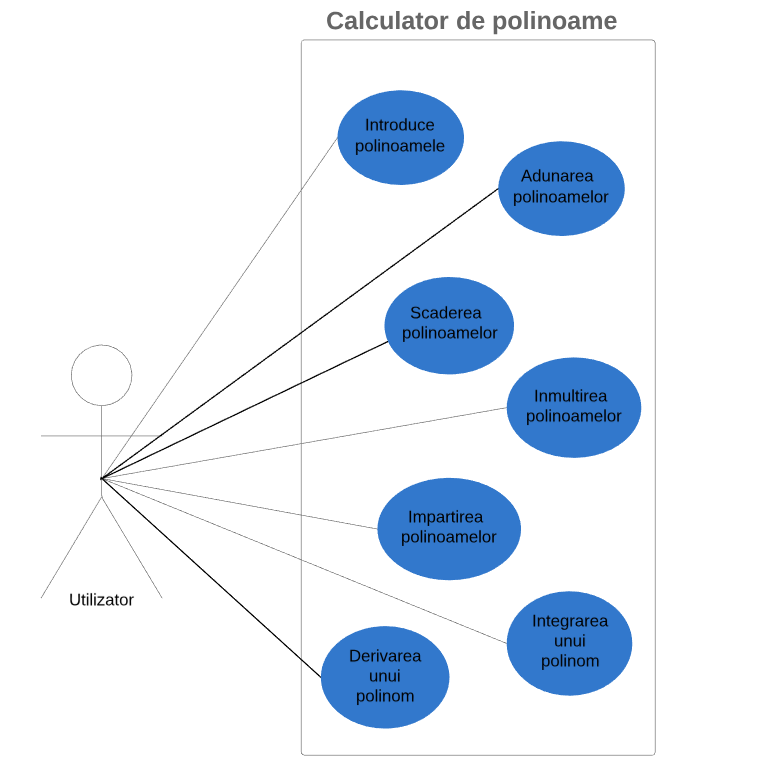
Actor: Utilizatorul care doreste sa efectueze operatii pe polinoame

Preconditii: Utilizatorul introduce in campul specificat din interfata grafica polinomul pentru care doreste sa efectueze operatia de derivare si selecteaza operatia de derivare prin click asupra butonului „Derivare”.

Postconditii: Utilizatorul o sa observe in campul rezultat al interfetei grafice rezultatul operatiei.

Flow: Datele introduse de utilizator vor fi preluate, verificare, procesate si in cazul in care se indeplinesc conditiile respective se va calcula suma celor doua polinoame. Pentru aceasta operatie, cat si pentru cea de integrare utilizatorul este nevoit sa introduca polinomul intr-un singur camp (daca se introduce in celalalt camp si primul este gol, utilizatorul va primi mesaj de eroare).

(Acelasi procedeu se va efectua si pentru operatia de integrare).



**3. Proiectare**

In ceea ce priveste proiectarea propriu zisa a proiectului, am structurat aplicatia in pachete, fiecarui pachet fiindu-i intrebuintate responsabilitati comune/ inrudite. De asemenea, aplicatia este structurata pe principiul Model-View-Controller. Modelul este reprezentarea obiectuala a polinoamelor, View este reprezentata de interfata grafica, care faciliteaza interactiunea utilizatorului cu aplicatia, iar Controllerul face legatura intre cerintele utilizatorului transmise prin intermediul interfetei si modelul(polinomul). Tot in Controller se implementeaza ascultatorii pentru butoanele din interfata. Pachetele create sunt:

* Stocare: reprezinta modelul propriu-zis, compus din clasele Monom si Polinom.
* Operatie: compus din interfata Operatie si clasele Adunare, Scadere, Inmultire, Impartire, Derivare, Integrare, clase care implementeaza interfata Operatie si care descriu operatiile care se pot efectua.
* Controller: cu clasa reprezentatica ControllerCalcPolinoame. Acest pachet se ocupa cu legarea interfetei grafice (preluarea datelor introduse de utilizator, analiza lor) de model (polinom).
* GUI: in cadrul acestui pachet se realizeaza implementarea interfetei grafice.
* PT2019.A1.A1: contine clasa App (cu metoda main), si clasa JunitTest, in cadrul careia se realizeaza testele.

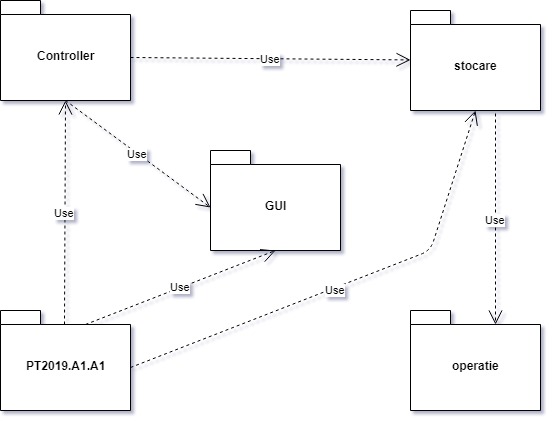
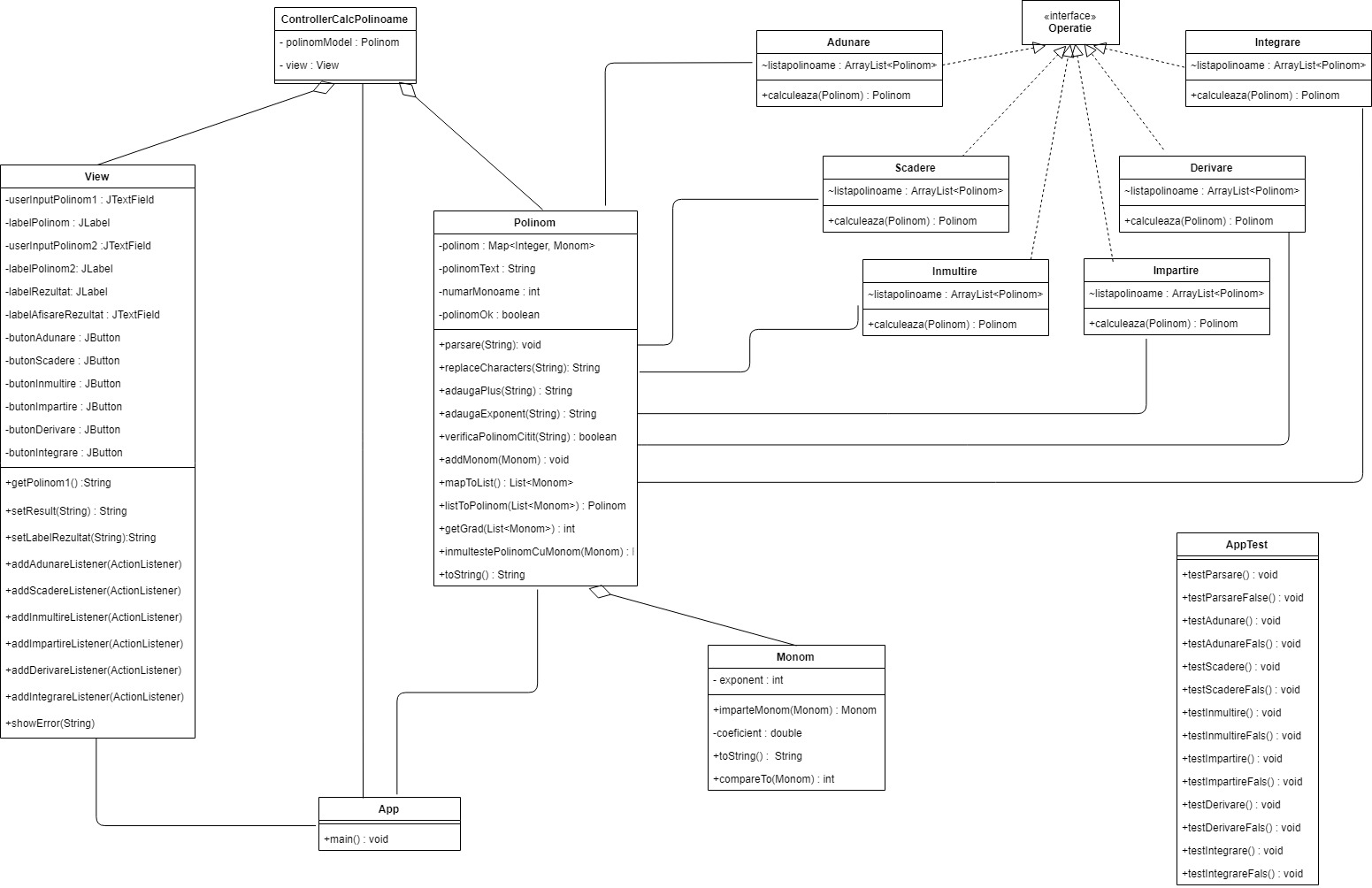


Fig2. Diagrama de pachet



In ceea ce priveste Structurile de Date, pentru stocarea polinomului am folosit HashMap<Integer, Monom>, cheia fiind reprezentata de exponentul monomului, iar valoarea de Monom. Am ales sa folosesc aceasta structura deoarece am considerat ca operatiile se pot efectua mult mai usor, tinand cont ca doua monoame cu acelasi exponent vor fi mapate la aceeasi adresa, este mult mai usor de a reduce numarul de termeni asemenea.

De asemenea am folosit si ArrayList, pentru a efectua anumite parcurgeri pe polinoame, am implementat interfata Comparable pentru a putea face sortarea in functie de exponentul monomului in ordine descrescatoare. Totodata am definit propria interfata „Operatie” folosita pentru implementarea mai usoara a operatiilor pe polinoame. Legat de algoritmii folositi, pentru implementarea operatiei de impartire a polinoamelor am folosit pseudocodul Polynomial long division (vezi Bibliografie).

**4. Proiectare**

In continuare voi descrie fiecare clasa, impreuna cu campurile si metodele importante, inclusiv implementarea interfetei utilizator.

* **Monom**: Un monom reprezinta structura de baza a unui polinom, astfel este necesar ca in

aceasta clasa sa stocam informatii legate de coeficientul monomului (de tip double) si exponentul acestuia (de tip intreg). Legat de metodele acestei clase, pe langa cele accesaore si mutatore avem metoda pentru:

* imparteMonoame, primeste un monom ca si argument si returneaza un monom nou reprezentand impartirea dintre this si argumentul metodei
* compareTo, folosita pentru sortarea listelor de monoame
* toString, care afiseaza monomul, verificand printre altele daca exponentul este 1, daca coeficietul este 1, evitand afisarea „1x” sau „x^1”;
* **Polinom**: Aceasta clasa contine reprezentarea propriu zisa a polinomului. Stocarea polinomului se face sub forma de HashMap, unde cheia este reprezentata de exponent, iar valoarea de Monom. Campul acestei clase este HashMap-ul polinom. Principalele metode ale acestei clase sunt:
* verificaPolinomCitit: se asigura ca utilizatorul a introdus doar caractere valide (cifre, litera „x”, caracterele „+, -, ^, .”).
* replaceCharacters: in cadrul acestei metode Stringul introdus este prelucrat, se elimina spatiile albe, se adauga coeficentii 1 in cazul in care utilizatorul a introdus doar „x”, si se adauga exponentul 0, respectiv 1 pentru cazurile „x+8”
* parsare: in cadrul acestei metode se apeleaza metodele prezentate mai sus, si in cadrul ei se construieste Hashmap-ul , in care se stocheza polinomul. Parsarea se realizeaza cu ajutorul metodei split, specifica Stringurilor.
* addMonom: metoda primeste ca argument un monom care se adauga in HashMap-ul corespunzator. Daca la adresa la care se doreste maparea exista deja un monom, atunci se creeaza un nou monom cu coeficientul egal cu suma coefientilor monoamelor care se doresc mapate la aceeasi cheie.
* inmultesteMonomCuPolinom: metoda primeste ca argument un monom si returneaza un polinom egal cu inmultirea polinomului this si monomul trimis ca si argument
* mapToList: se creaza un ArrayList de Monoame din polinomul existent, lista care va fi ulterior folosita.
* listToPolinom: creeaza un polinom echivalent cu lista trimisa ca si argument.
* getGrad: returneaza cel mai mare exponent al unui monom cu coeficient nenul din polinm
* toString: metoda folosita pentru afisarea frumoasa a polinomului. In cadru ei se parcurge ArrayList-ul sortat descrescator si se apeleaza metoda toString pentru fiecare monom.
* **Operatie**: (reprezinta o interfata cu o singura metoda „calculeaza care primeste ca si argumente un numar variabil de polinoame si returneaza o lista de polinoame, corespunzator fiecarei operatii in parte”). Aceasta interfata este implementata de 6 clase, reprezentand fiecare operatie.
* **Adunare:** clasa care implementeaza interfata Operatie, continand cod pentru metoda calculeaza. Ca si camp avem o lista de polinoame, iar metoda calculeaza este cea responsabila de operatia de adunare. Aceasta metoda se bazeaza pe metoda addMonom din clasa polinom: se parcurg pe rand cele doua polinoame trimise ca si parametru si se apeleaza metoda addMonom pentru fiecare monom.
* **Scaderea**: la fel ca si clasa Adunare, aceasta clasa contine doar metoda claculeaza, operatia de scadere bazandu-se fix pe aceeasi logica ca si la adunare, cu exceptia faptului ca la parcurgerea polinomul 2, metoda addMonom se apeleaza cu –coeficient.
* **Inmultire**: aceasta clasa are ca si campuri un ArrayList de polinoame, iar ca si metode, doar metoda calculeaza. In cadrul metodei calculeaza, se parcurg cele doua polinoame monom cu monom, si la fiecare pas se creeaza un monom nou cu coeficientul egal cu produsul coeficientilor celor doua monoame si exponentul egal cu suma exponentilor celor doua monoame. Noul monom este adaugat in polinomul rezultat apelandu-se metoda addMonom, descrisa in clasa polinom.
* **Impartire**: aceasta clasa se ocupa de operatia de impartire a doua polinoame, in special metoda calculeaza, care primeste ca si argumente un numar variabil de polinoame (se va verifica daca numarul de argumente este 2, reprezentand numarul corect de argumente pentru care se poate efectua operatia) . Algoritmul de impartire este transpunerea in cod Java a algoritmului care se efectueaza pentru impartirea polinoamelor. Aceasta metoda returneaza o lista de polinoame, adica atat catul impartirii cat si restul.
* **Derivare** + **Integrare:** clasele implementeaza interfata Operatie, si implementeaza metoda calculeaza,care este similara pentru cele doua operatii: se parcurge polinomul trimis ca argument si se creeaza un polinom nou, crespunzator derivarii sau integrarii prin modificarea coeficientilor si exponentilor.
* **ControllerCalcPolinoame:**  este clasa care face legatura intre interfata grafica si model( clasa Polinom in cazul de fata), si este responsabila de implementarea ascultatorilor pentru butoanele interfetei si trimiterea de comenzi in functie de ce butoane s-au apasat pentru efectuarea operatiei corespunzatoare. Tot in cadrul acestei clase se seteaza si textul pentru campul de rezultat.
* **View:** Aceasta clasa este responsabila cu interfata grafica, iar ca variabile instanta avem trei Jlabel-uri, doua TextField-uri, in care utilizatorul va introduce polinoamele, un TextField in car ese va afisa rezultatul si sase JButton, cate unul pentru fiecare operatie posibila. Metodele principale ale acestei clase sunt legate de citirea String-urilor din TextField-uri, setarea de mesaje in TextField-ul pentru rezultat, si cate o metoda pentru adaugarea de ascultator pentru fiecare JButton. In constructorul acestei clase se seteaza aspectul ferestrei. Fereastra este construita din doua JPanel-uri, in care se adauga pe rand elementele decrise mai sus.

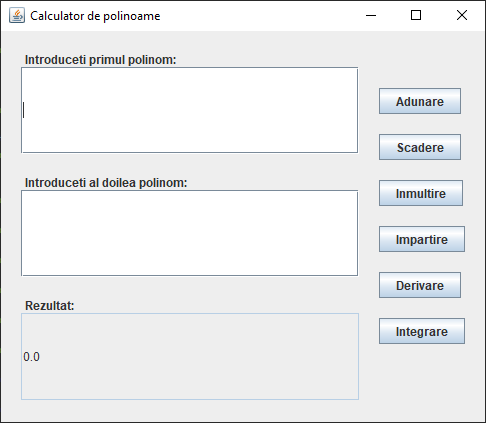


Fig.4 Interfata Grafica

Se observa in Fig. 4 campurile in care utilizatorul trebuie sa introduca polinaomele, sugerate de etichetele(„Introduceti primul polinom”, „Introduceti al doilea polinom”), campul de afisare al rezultatului si cele 6 butoane reprezentand operatiile posibile. In fig. 5 se observa transmiterea de mesaj de eroare in cazul in care utilizatorul introduce date invalide, iar in fig.6 se observa afisarea in campul „Rezultat”, rezultatul corect al operatiei de adunare efectuate pe datele introduse corect.

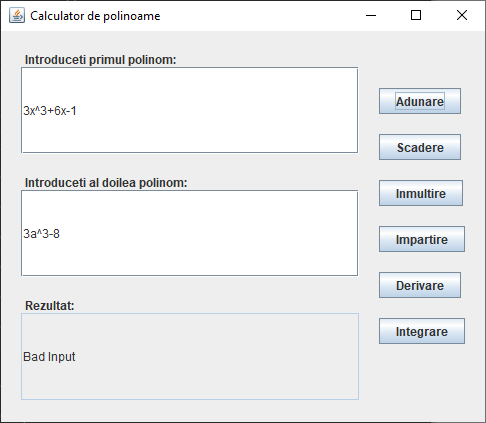


Fig. 5 Date de intare invalide in campul 2

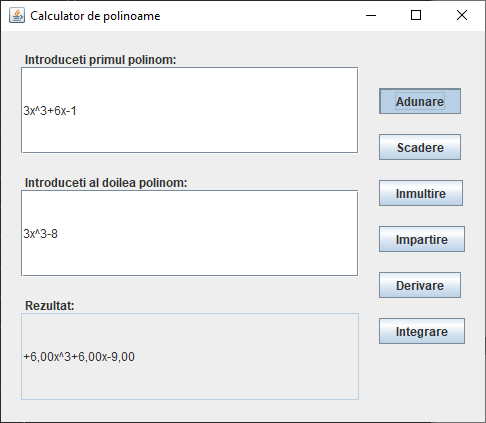
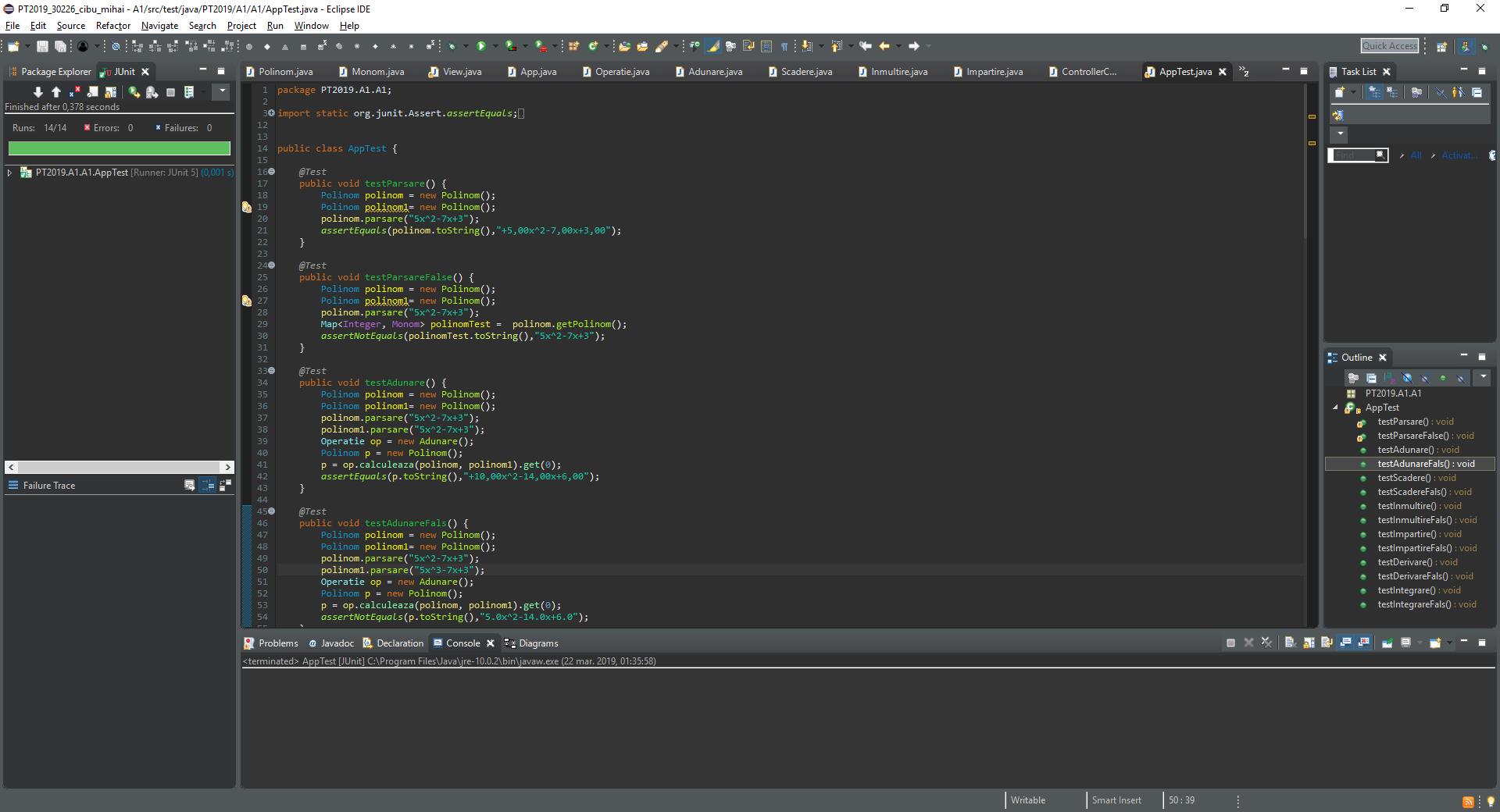


Fig.6 Date de intrare valide plus afisarea rezultatului operatiei selectate

**5. Rezultate**

Pentru verificare corectitudinii programului am folosit Libraria JUnit, implementand astfel teste, doua pentru fiecare metoda care efectueaza operatiile de adunare, scadere, inmultire, impartire, derivare, integrare si pentru metoda de parsare a Stringului citit de la tastatura.

Fig.7 Rezultatele testarii

**6. Concluzii**

Pe parcursul proiectarii, dezvoltarii si implementarii acestei teme mi-am exersat abilitatile de a scrie cod in Java, am descoperit lucruri noi legate de implementarea anumitor metode, am invatat despre Varargs (metode care pot primi ca argument numar variabil de variabile), ceea ce m-a ajutat la implementarea metodelor pentru operatiile pe polinoame.

In ceea ce priveste posibilitatile de dezvoltare ulterioara, aplicatia s-ar putea dezvolta prin adaugarea de polinaome intr-o „mini baza de date”, utilizatorul putand introduce mai multe polinoame, apoi sa selecteze din ele si sa se efectueze operatia dorita.

**7. Bibliografie**

<https://stackoverflow.com>

draw.io/ - pentru generarea diagramelor UML

https://en.wikipedia.org/wiki/Polynomial\_long\_division