UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Tehnici de programare fundamentale
Tema 1

Calculator de polinoame

- Documentație -

Realizat de:
Ducai David Alex – grupa 30223

Cuprins

1. Obiectivele temei

Obiectivul principal al temei este realizarea unui calculator care lucrează cu polinoame. Calculatorul trebuie să aibă o interfață grafică, care permite introducerea de polinoame, selectarea operației matematice dorite și afișarea rezultatului operației selectate.

Obiectivele secundare ale temei sunt următoarele:

- Analizarea problemei și identificarea resurselor necesare realizării temei propuse – *capitolul 2*
- Realizarea design-ului calculatorului de polinoame – *capitolele 3 și 4*
- Implementarea calculatorului – *capitolul 4*
- Testarea calculatorului – *capitolul 5*

2. Analiza problemei

2.1. Cerințe funcționale

Calculatorul de polinoame trebuie să permită utilizatorului să introducă maxim două polinoame și să selecteze operația dorită. În spatele interfeței, calculatorul trebuie să permită realizarea operațiilor de adunare, scădere, înmulțire și împărțire asupra a două polinoame și realizarea operațiilor de integrare și derivare asupra unui polinom.

2.2. Cerințe non-funcționale

Calculatorul de polinoame trebuie să vină însoțit de o interfață intuitivă, ușor de folosit de către utilizator, care să ofere instrucțiuni de utilizare în caz de nevoie și să indice eventualele erori de utilizare a acesteia.

2.3 Cazuri de utilizare

Pentru toate cazurile de utilizare prezentate, *actorul principal este utilizatorul* calculatorului de polinoame. Anumite operații au cazuri de utilizare similare și vor fi prezentate simultan, pentru simplificare.

1. ***Caz de utilizare**: adunarea, scăderea sau înmulțirea polinoamelor*
Scenariu principal (operație realizată cu succes):
   1. Utilizatorul introduce, prin interfață, două polinoame
   2. Utilizatorul selectează butonul operației dorite (“Adunare”, “Scădere” sau “Înmulțire”)
   3. Calculatorul preia polinoamele introduse de utilizator și realizează operația selectată
   4. Calculatorul returnează în interfață, sub formă de text, rezultatul operației selectate

Scenariu secundar (Polinoame neintroduse):
- Utilizatorul nu a introdus un număr suficient de polinoame pentru realizarea operațiilor menționate mai sus
- Se deschide o nouă fereastră de atenționare pentru utilizator
- Se revine în scenariul principal la pasul 1.

2. ***Caz de utilizare**: integrarea sau derivarea unui polinom*
Scenariu principal (operație realizată cu succes):
   1. Utilizatorul introduce, prin interfață, un polinom în primul câmp disponibil
   2. Utilizatorul selectează butonul operației dorite (“Integrare” sau “Derivare”)
   3. Calculatorul preia polinomul introdus de utilizator și realizează operația selectată
   4. Calculatorul returnează în interfață, sub formă de text, rezultatul operației selectate

Scenariu secundar (Polinom neintrodus):
- Utilizatorul nu a introdus un polinom în primul câmp disponibil în interfață
- Se deschide o nouă fereastră de atenționare pentru utilizator
- Se revine în scenariul principal la pasul 1.

3. ***Caz de utilizare**: împărțirea a două polinoame*
Scenariu principal (operație realizată cu succes):
   1. Utilizatorul introduce, prin interfață, două polinoame
   2. Utilizatorul selectează butonul pentru împărțire (“Împărțire”)
   3. Calculatorul preia polinoamele introduse de utilizator și realizează operația de împărțire
   4. Calculatorul returnează în interfață, sub formă de text, câtul și, daca există, restul împărțirii

Scenariu secundar (Polinoame neintroduse):
- Utilizatorul nu a introdus un număr suficient de polinoame pentru realizarea împărțirii
- Se deschide o nouă fereastră de atenționare pentru utilizator
- Se revine în scenariul principal la pasul 1.

Scenariu secundar (Gradul primului polinom este necorespunzător):
- Utilizatorul a introdus în primul câmp un polinom cu un grad strict mai mic decât gradul polinomului introdus în al doilea câmp
- Se deschide o nouă fereastră de atenționare pentru utilizator

- Se revine în scenariul principal la pasul 1.

Scenariu secundar (Al doilea polinom este polinomul nul (zero)):

- Utilizatorul a introdus în al doilea câmp polinomul 0
- Se deschide o nouă fereastră de atenționare pentru utilizator
- Se revine în scenariul principal la pasul 1.

4. ***Caz de utilizare****: ștergerea textelor introduse*

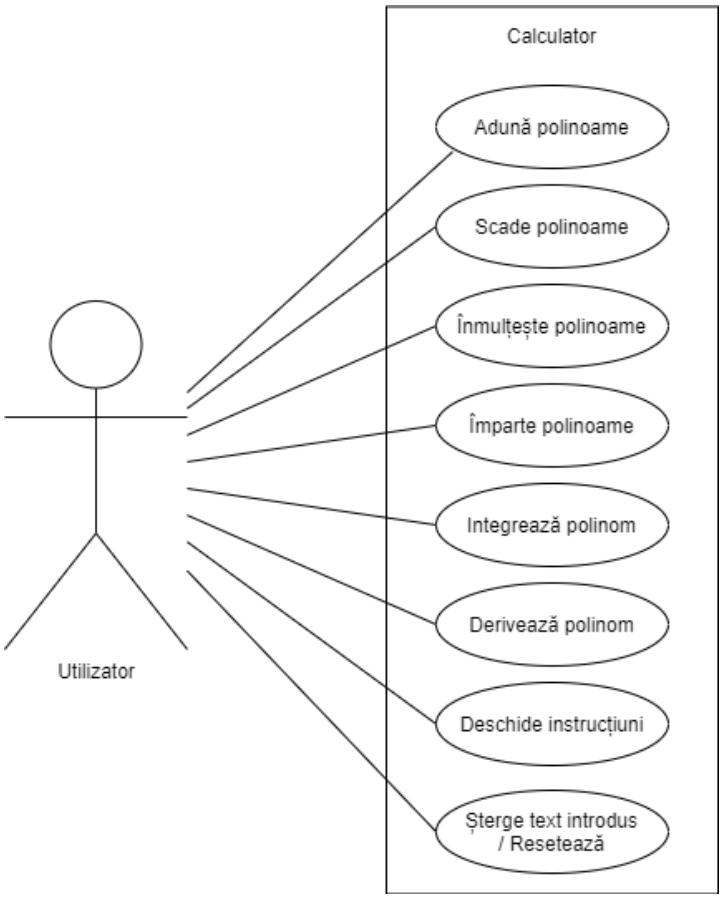
Scenariu principal:

1. Utilizatorul introduce în câmpurile disponibile text de orice tip
2. Utilizatorul apasă butonul de ștergere al câmpurilor (“Ștergere”)
3. Calculatorul șterge textul introdus de utilizator și ascunde câmpul de afișare al rezultatelor

4. ***Caz de utilizare****: citirea instrucțiunilor de utilizare*

Scenariu principal:

1. Utilizatorul apasă butonul de deschidere a instrucțiunilor de utilizare (“Ajutor”)
2. Se deschide o nouă fereastră care conține instrucțiunile de utilizare ale calculatorului de polinoame
3. Utilizatorul poate citi sau nu textul
4. Utilizatorul apasă butonul de închidere din josul ferestrei sau din colțul din dreapta-sus
5. Fereastra cu instrucțiuni se închide



3. Proiectarea temei

Pentru realizarea calculatorului de polinoame am ales modelul arhitectural Model–View–Controller (MVC). Acest model împarte aplicația în trei părți principale: partea de date (Model), partea de ieșiri (View) și partea de intrări (Controller).

Partea de date ("Model") – reține datele utilizate, funcționalitățile și lucrează cu acestea

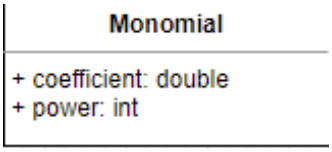
Partea de ieșiri ("View") – afișează rezultate și informații utilizatorului

Partea de intrări ("Controller") – primește imput sub formă de evenimente (mișcări de mouse, apasări de butoane, etc.) și le traduce în cereri de servicii, care sunt adresate fie părții de date, fie părții de ieșire

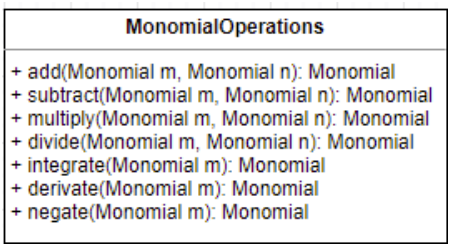
Modelul MVC este avantajos, deoarece permite structurarea ușoară a proiectului, conectarea logică între părți, accesul mai rapid la diferite elemente și reutilizarea acestora.

Pirmul pachet definit va fi pentru *modelul de date*. Prima clasă la care ne gândim logic este clasa care descrie un ***polinom***. Un polinom este o expresie, care conține variabile și constante (numite și coeficienți). Practic, un polinom este o colecție de variabile și constante. Alăturarea unei variabile ridicate la o putere cu un coeficient determină un monom. Deci, un polinom este o *colecție de monoame*.

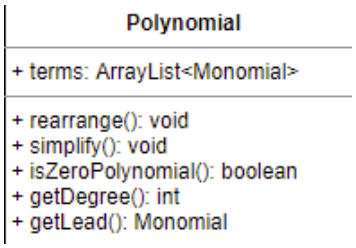
Prima clasă care trebuie implementată este clasa care descrie un ***monom***. Un monom conține un coeficient și o variabilă ridicată la o putere pozitivă întreagă. Acestea vor fi atributele clasei Monomial. Clasa va conține constructori, metode pentru accesarea și modificarea atributelor, pentru afișarea sub formă de text și pentru sortare în funcție de putere.



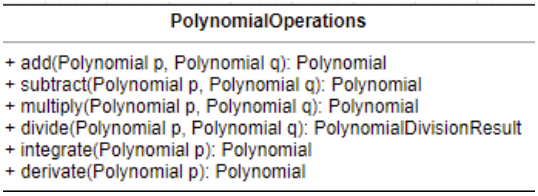
Următoarea clasă este clasa care conține ***operații matematice pe monoame***. Această clasă nu va avea atribute și va conține metode statice de adunare, scădere, înmulțire, împărțire, derivare, integrare și negare a monoamelor. Aceste operații vor fi folositoare la implementarea clasei Polynomial și a clasei cu operații asupra polinoamelor. Operațiile pe monoame și polinoame vor fi adăugate într-un *pachet separat pentru operații*.



Acum, putem implementa ***clasa principală*** *a temei, care descrie un* ***polinom***. Un polinom este o listă de monoame. Această listă va fi singurul atribut al clasei. Clasa va conține constructori, metode pentru accesarea și modificarea listei de monoame, afișarea polinomului sub formă de text și simplificarea polinomului. Clasa mai conține trei metode pentru determinarea gradului polinomului, returnarea primului element din listă și verificarea identității polinomului cu polinomul nul. Aceste metode vor fi utilizate la operațiile pe polinoame și la afișare. Chenarul de mai jos cuprinde doar metodele mai importante.



Urmează clasa care conține ***operațiile matematice ale calculatorului,*** o altă parte importantă a temei. Această clasă va fi asemănătoare clasei cu operațiile pe monoame, care este descrisă mai sus. Conține metode statice de adunare, scădere, înmulțire, împărțire, derivare și integrare. Implementarea operațiilor pe monoame simplifică mult implementarea operațiilor pe polinoame.

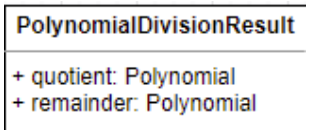


Algoritmul utilizat pentru metoda de împărțire este cunoscut în limba engleză sub numele de *"Polynomial Long Division"*:

Dorim să împărțim polinomul *n* la *d*. În primul rând, este necesar ca polinomul *d* să nu fie 0. Presupunem că termenii polinoamelor sunt sortați descrescător în funcție de puteri: de la cea mai mare la cea mai mică.

```
Fie t un polinom auxiliar
Fie q și r câtul, respectiv restul împărțirii lui n la d
q <- 0
r <- n
cât timp r != 0 și grad(r) >= grad(d) execută {
      t <- lead(r) / lead(d) // lead returnează primul termen al unui
polinom (cel cu puterea egală cu gradul)
      q <- q + t
      r <- r – t * d
}
scrie q,r
```

Observăm ca acest algoritm va returna două polinoame. O metodă nu poate returna două obiecte diferite în același timp. Așadar, vom crea o nouă ***clasă de rezultat al împărțirii de polinoame***, care va avea ca atribute două polinoame: câtul și restul împărțirii. Un obiect de acest tip va putea fi returnat de metodă și astfel vom primi două polinoame dintr-o singură metodă.



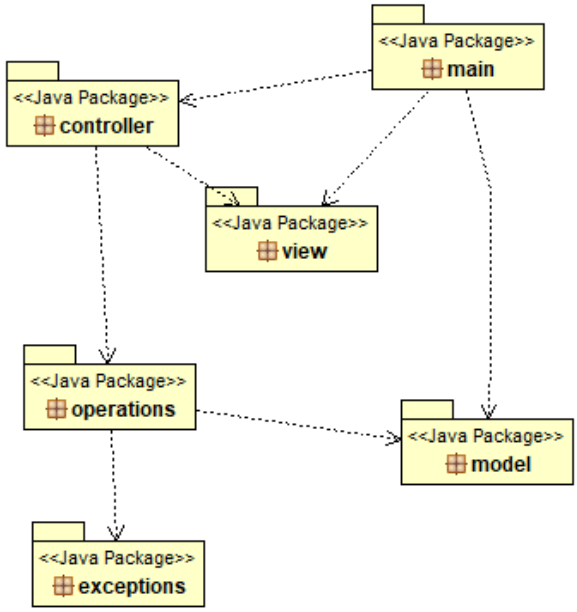
Această clasă are contructor, metode de afișare sub formă de text și de accesare și modificare a atributelor.

Acestea au fost clasele care reține date și operează cu acestea. Următorul pachet îi corespunde părții de ieșire a aplicației ("View"). Pachetul conține view-ul pentru aplicația principală și un view pentru fereastra cu instrucțiuni.

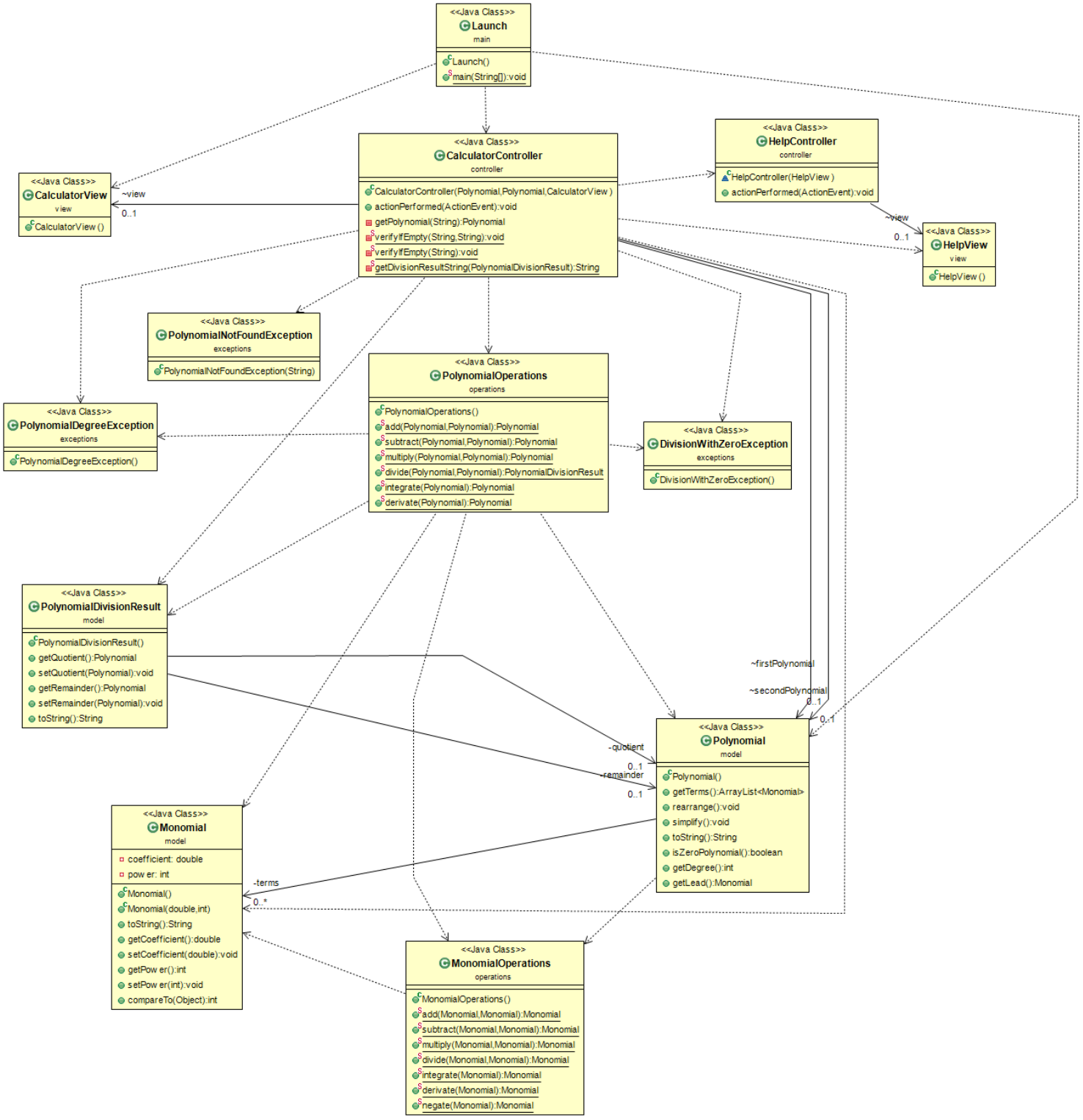
Ultimul pachet principal îi corespunde părții de intrare ("Controller"). Acest pachet conține două clase "Controller" pentru aplicația în sine și pentru fereastra cu instrucțiuni.

Diagrame UML:

- diagrama de pachete (mai există un pachet pentru testarea unitară care nu a fost introdus în diagramă)



- diagrama de clase



4. Implementarea temei

În acest capitol vor fi descrise toate clasele din aplicație împreună cu atributele și metodele cele mai importante.

4.1. Pachetul *model*

Clasa **Monomial –** reține datele unui monom

Atribute

- coefficient – coeficientul monomului
- power – exponentul (puterea) variabilei

Clasa **Polynomial –** reține datele unui polinom

Atribute

- terms – lista de monoame

Metode

- rearrange( ) – sortează termenii unui polinom descrescător de la cea mai mare putere la cea mai mică (în final, prima poziție din listă conține monomul cu puterea cea mai mare)
- simplify ( ) – una dintre cele mai important metode din aplicație. Această metodă simplifică un polinom. De exemplu, după apelul metodei, polinomul $x^2 + x + x + 1$ devine $x^2 +2x +1$. Acestă metodă este utilizată mai ales în metodele operațiilor matematice asupra polinoamelor. Modul de funcționare este următorul:
  1. se sortează polinomul folosind rearrange( )
  2. se adaugă câte un element din polinom într-un listă auxiliară dacă nu a fost adăugat anterior un termen cu aceeași putere pâna la epuizare – după ,sari la 7
  3. se verifică dacă mai există termeni cu aceeași putere după el
  4. dacă da, se adună la termenul adăugat în lista auxiliară și se numără câți termeni au fost adăugați, pentru a putea fi excluși de la adăugare la următoarea iterație
  5. altfel, nu mai există alte monoame cu aceeași putere și se oprește căutarea
  6. se revine la pasul al doilea
  7. lista de monoame inițială va referi lista auxiliară
- isZeroPolynomial( ) – verifică dacă polinomul este nul (Primul termen este monomul 0 cu exponentul 0)
- getDegree( ) – returnează gradul polinomului
- getLead( ) – returnează primul termen din lista de monoame

Clasa **PolynomialDivisionResult –** reține rezultatul unei împărțiri polinomiale

Atribute

- quotient – polinomul cât
- remainder – polinomul rest

4.2. Pachetul *operations*

Clasa **MonomialOperations –** deține metodele pentru operații pe monoame

Metode

- add(Monomial m, Monomial n) – returnează suma monoamelor *m* și *n*
- subtract(Monomial m, Monomial n) – returnează diferența dintre monoamele *m* și *n*
- multiply(Monomial m, Monomial n) – returnează produsul dintre *m* și *n*
- divide(Monomial m, Monomial n) – returnează câtul dintre monoamele *m* și *n*
- integrate(Monomial m) – returnează integrala nedefinită a monomului *m*
- derivate(Monomial m) – returnează derivata monomului *m*
- negate(Monomial m) – returnează opusul monomului *m* (se neagă coeficientul)

Clasa **PolynomialOperations**
Metode

- add(Polynomial p, Polynomial q) – returnează suma polinoamelor p și q
  - Se adaugă termenii ambelor polinoame într-un singur polinom *rezultat*
  - Se apelează metoda simplify( ) pe *rezultat*
  - Se returnează rezultatul
- subtract(Polynomial p, Polynomial q) – returnează diferența dintre *p* și *q*
  - Se adaugă termenii primului polinom într-un singur polinom *rezultat*
  - Se adaugă opușii termenilor din al doilea polinom în *rezultat* (se apelează negate( ) pe fiecare termen din al doilea polinom)
  - Se apelează metoda simplify( ) pe *rezultat*
  - Se returnează rezultatul
- multiply(Polynomial p, Polynomial q) – returnează produsul polinoamelor *p* și *q*
  - Se înmulțește fiecare termen din *p* cu fiecare termen din *q* și se adaugă fiecare produs într-un polinom *rezultat*
  - Se apelează metoda simplify( ) pe *rezultat*
  - Se returnează rezultatul
- divide(Polynomial p, Polynomial q) – returnează câtul și restul împărțirii lui *p* la *q*
  - Algoritmul utilizat este descris la pagina 10
- integrate(Polynomial p) – returnează integrala nedefinită a lui *p*
  - Se adaugă într-un polinom *rezultat* integrala fiecărui termen din *p*
- derivate(Polynomial p) – returnează derivata lui *p*
  - Se adaugă într-un polinom *rezultat* derivata fiecărui termen din *p*

4.3. Pachetul *exceptions*

Conține excepții personalizate pe care le poate arunca programul.

Clasa **PolynomialNotFoundException** – excepție aruncată în controller-ul aplicației dacă nu s-a introdus un polinom într-un câmp

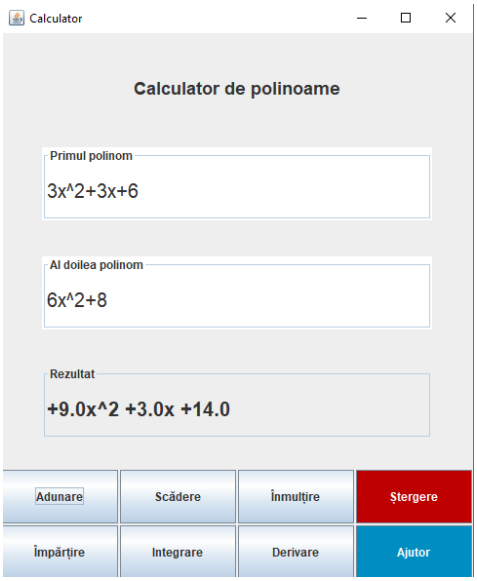
Clasa **PolynomialDegreeException –** excepție aruncată de metoda divide(Polynomial p, Polynomial q) și tratată în controller-ul aplicației dacă gradul primului polinom este mai mic decât gradul celui de-al doilea

Clasa **DivisionWithZeroException -** excepție aruncată de metoda divide(Polynomial p, Polynomial q) și tratată în controller-ul aplicației dacă polinomul *q* este 0

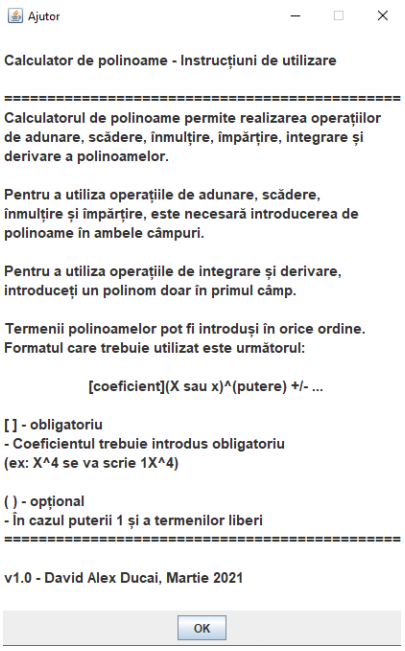
4.4. Pachetul *view*

Conține view-urile pentru aplicație și pentru fereastra cu instrucțiuni de utilizare

Clasa **CalculatorView –** interfața grafică a calculatorului (imagine cu exemplu de utilizare)



Clasa **HelpView -** interfața grafică a ferestrei cu instrucțiuni de utilizare



4.5. Pachetul *controller*

Conține controller-ele asociate fiecărui view din pachetul *view*.
Clasa **CalculatorController**
Metode

- getPolynomial(String imput) – returnează un polinom dintr-un string introdus într-un câmp
- verifiyIfEmpty(String field) – aruncă excepția **PolynomialNotFoundException** dacă un câmp este gol
- getDivisionResultString(PolynomialDivisionResult result) – returnează un text formatat pentru afișarea rezultatului împărțirii

Clasa **HelpController –** conține o singură adăugare de *listener* pentru butonul de închidere

4.6. Pachetul *main*

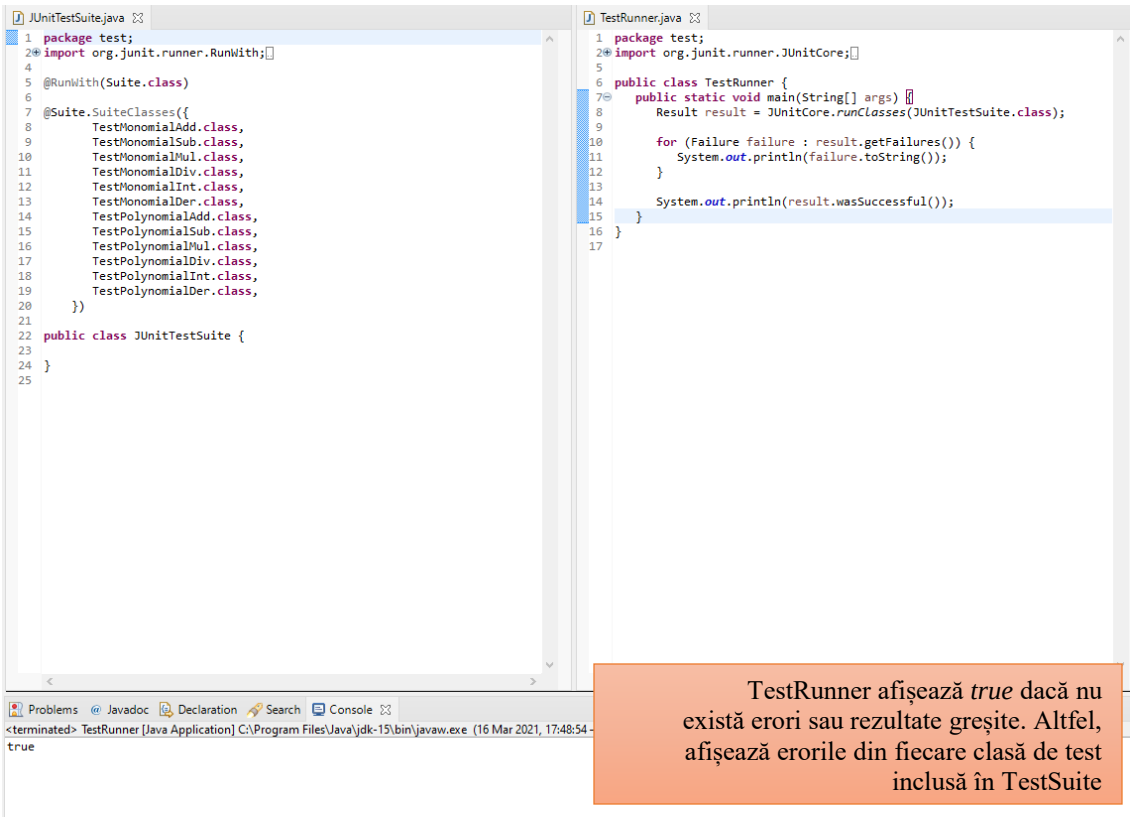
Clasa **Launch –** se execută aplicația

4.7. Pachetul *test*

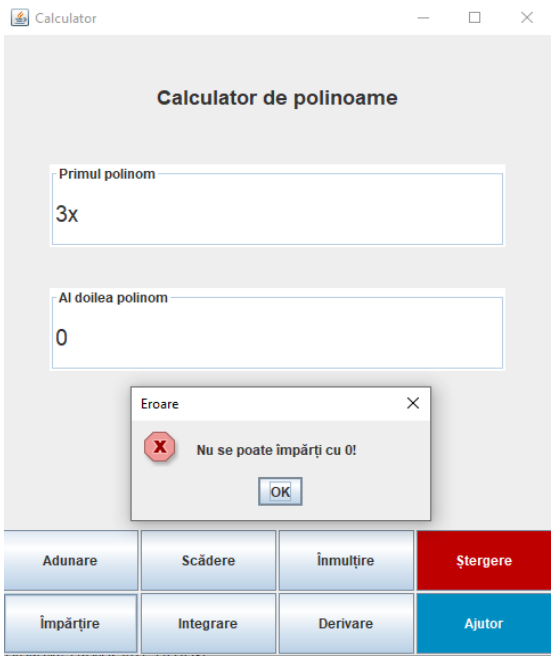
Conține clase de testare în JUnit pentru fiecare metodă din clasele MonomialOperations și PolynomialOperations.

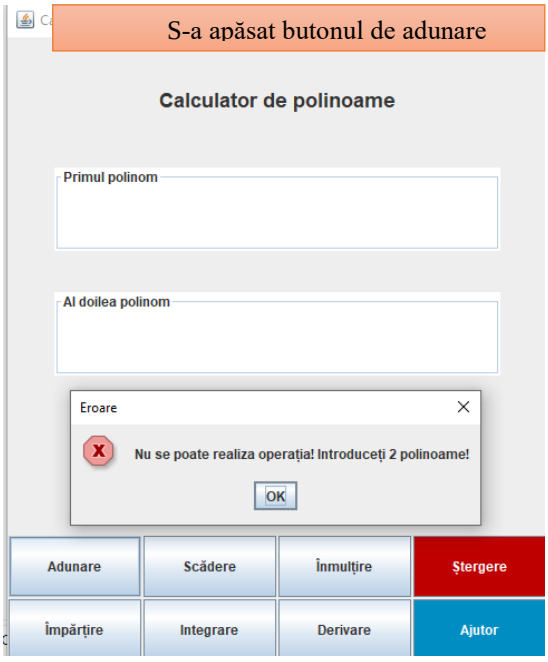
5. Testarea temei și rezultate

Am creat clase de testare în JUnit pentru fiecare metodă din pachetul *operations.* Nu s-au detectat erori sau rezultate eronate.



Testări în aplicație ale excepțiior:





6. Concluzii și dezvoltări ulterioare

Realizând această temă, am învățat:

- să utilizez *regex* pentru extragerea de pattern-uri într-un string
- metode utile pentru *ArrayList,* precum remove(index)
- cum să utilizez numai bucla for-each în lucrul cu listele
- cum sa pun *border* la *JTextField*
- cum să testez toate clasele de test JUnit deodată.

În versiuni ulterioare ale aplicației, aș dori să perfecționez partea de *regex* pentru recunoașterea monoamelor și partea de afișare a rezultatelor operațiilor (de exemplu dacă un coeficient are partea fracționară 0, să se afișeze numai partea întreagă, etc.).

7. Webografie

1. Wikipedia, “Polinom” - https://ro.wikipedia.org/wiki/Polinom;
2. Wikipedia,
“Polynomial Long Division” - https://en.wikipedia.org/wiki/Polynomial_long_division;
3. TutorialsPoint,
“JUnit SuiteTest” - https://www.tutorialspoint.com/junit/junit_suite_test.htm;