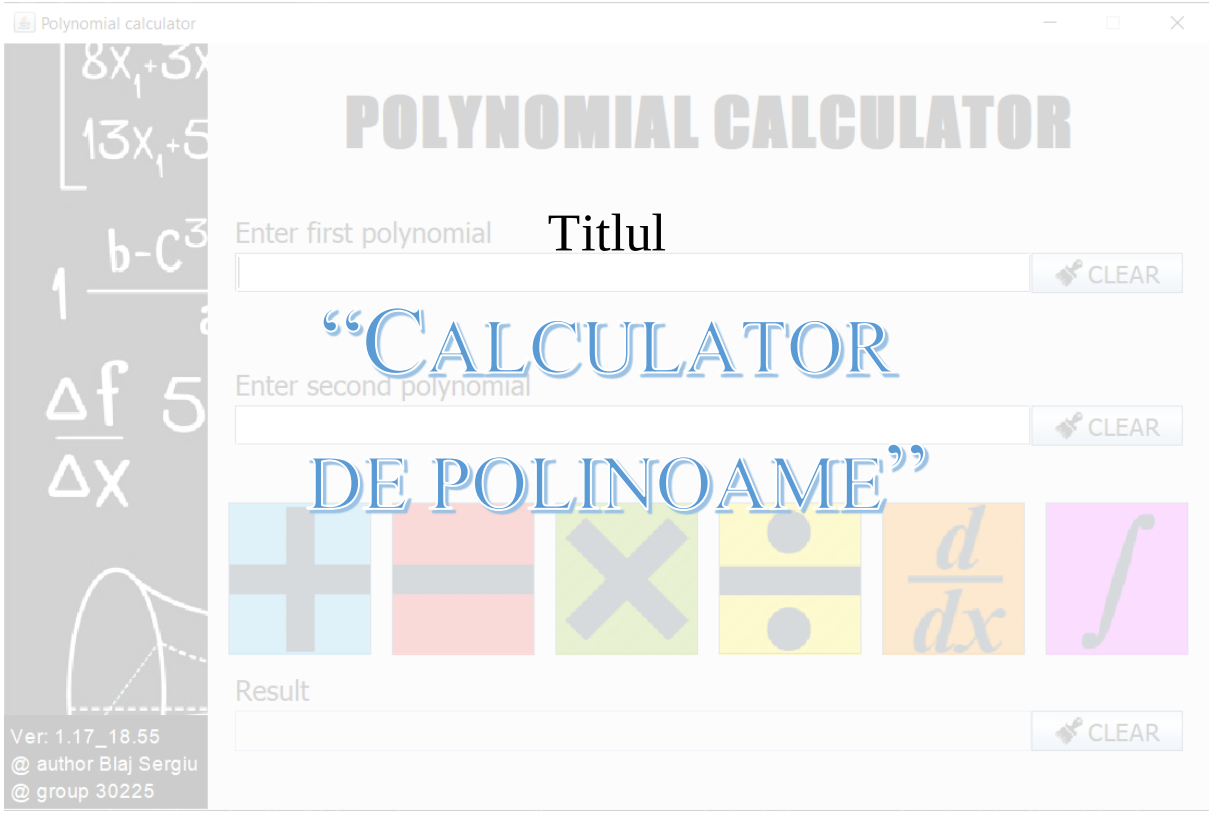
MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Proiect la disciplina

Tehnici de programare fundamentale



Titlul "CALCULATOR DE POLINOAME"

An universitar

2020 – 2021

Blaj Sergiu-Emanuel

An II, grupa 30225

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Cuprins

1. Obiectivul temei

Obiectivul principal al temei îl reprezintă proiectarea și implementarea unui calculator de polinoame de o singură variabilă cu coeficienți întregi. Suplimentar, acest calculator trebuie să asigure:

- utilizarea conceptelor de programare orientată pe obiecte;
- dezvoltarea sa folosind un pattern arhitectural;
- interacțiunea utilizatorului cu calculatorul de polinoame printr-o interfață grafică user-friendly, pentru a facilita navigarea prin aplicație: utilizatorul dispune de două zone de text pentru a introduce cele două polinoame care vor fi supuse operației matematice;
- operații de bază pe polinoame, precum adunarea și scăderea;
- operații avansate pe polinoame, precum înmulțirea, împărțirea, derivarea și integrarea lor, ce necesită procedee mai elaborate;
- posibilitatea de a reseta curăța zona de input, pentru introducerea unor noi scenarii de test;
- testarea și depanarea programului, pentru a verifica funcționalitatea aplicației.

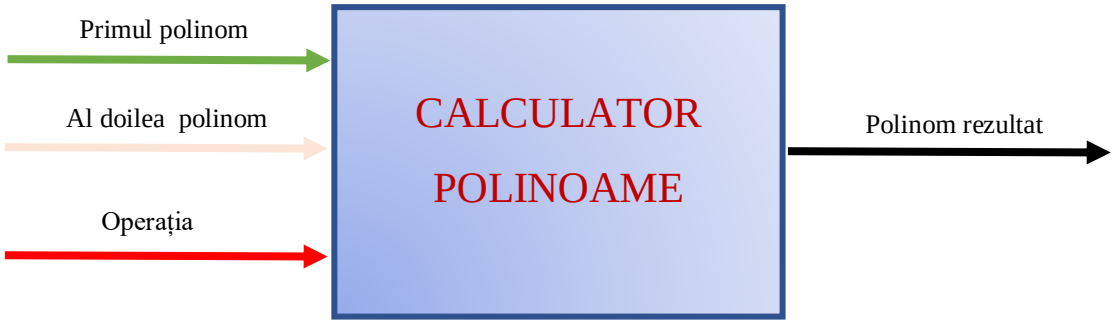


Figura 1

2. Analiza problemei

2.1. Cerințe funcționale

Aplicația dezvoltată trebuie să permită utilizatorului să introducă polinoame care vor fi supuse procedeelor matematice; acest lucru este asigurat printr-o zonă de text, utilizatorul având posibilitatea de a introduce valorile dorite de la tastatură.

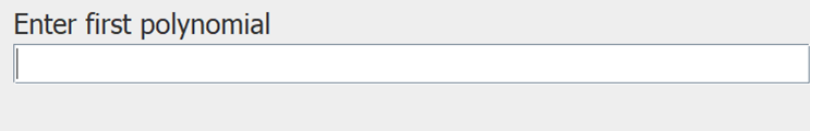


Figura 2.1.1

Calculatorul de polinoame trebuie să permit utilizatorului să aleagă operația matematică ce va fi aplicată asupra polinoamelor introduce de către acesta; acest fapt este înfăptuit prin selectarea unui buton ce corespunde unei operații matematice dorite.

După ce s-au introdus polinoamele dorite iar operația matematică a fost selectată, se va începe calcularea rezultatului. Acest procedeu constă în câțiva pași: în primul rând, se validează input-ul utilizatorului. Dacă utilizatorul nu a introdus un polinom scris corect, a lăsat zona de text goală sau a introdus orice altceva, va fi afișat un mesaj de eroare corespunzător.

Figura 2.1.2

Așa cum a fost menționat în capitolul 1, aplicația dezvoltată trebuie să execute operații de bază pe polinoame, precum adunarea sau scăderea acestora. Așa cum se observă și în figura 2.1.2, această versiune pe care am implementat-o și dezvoltat-o permite nu numai operații de bază, punând la dispoziție o gamă mai variată de operații matematice, precum înmulțirea sau împărțirea celor două polinoame introduse de utilizator, dar și derivarea sau integrarea primului polinom.

2.2. Cerințe non-funcționale

Pe lângă cerințele de bază, am considerat că următoarele cerințe vor asigura o mai bună interacțiune între calculatorul de polinoame și utilizator. Prima dintre ele, poate chiar cea mai importantă, este ca interfața grafică să fie una intuitivă, astfel ca și cuiva care nu obișnuiește să folosească o aplicație pe calculator să îi fie simplu să o folosească.

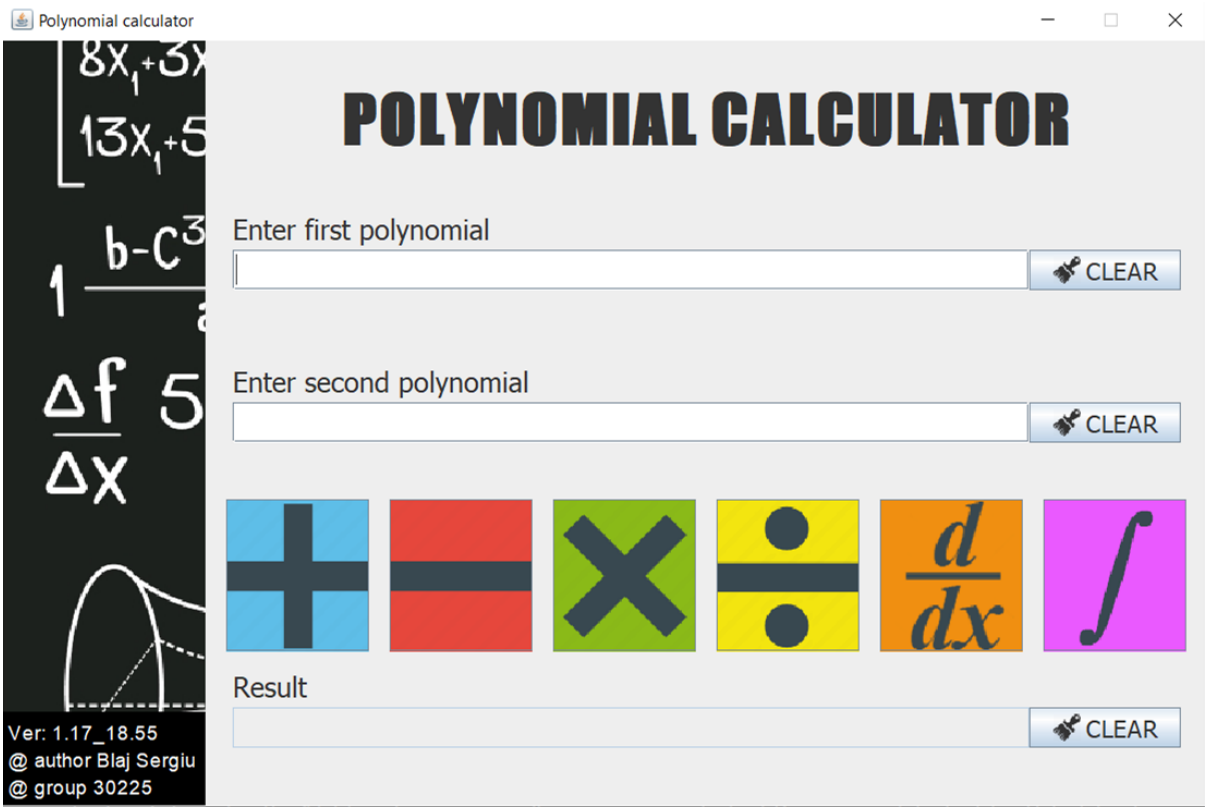


Figura 2.2.1

În figura 2.2.1 se poate observa că aplicația îndeplinește acest criteriu. În partea stângă, este prezentă o imagine legată de tematica aplicației cât și date despre dezvoltator. În partea principală, apare titlul aplicației, urmat de două zone de text. Așa cum este specificat, acestea sunt pentru a introduce cele două polinoame. Urmează, apoi, selectarea operației matematice dorite, alegând unul din cele șase butoane puse la dispoziție, fiecare având câte o imagine sugestivă. Concret, dacă utilizatorul dă click pe primul buton ("+"), se va face adunarea celor două polinoame, dacă utilizatorul dă click pe al cincilea buton ("*d/dx*"), se va integra primul polinom ș.a.m.d., rezultatul fiind afișat în câmpul de jos al aplicației.

O altă cerință non-funcțională o reprezintă posibilitatea utilizatorului de a șterge valorile introduse, în cazul unei greșeli sau a dorinței de a introduce alte noi polinoame. Acest lucru este realizat prin apăsarea butonului de "Clear", corespunzător câmpului dorit.

Figura 2.2.2

2.3. Scenarii de utilizare

2.3.1. Adunarea, scăderea, înmulțirea sau împărțirea polinoamenlor

Figura 2.3.1. descrie modul în care se va desfășura interacțiunea utilizatorului cu programul în cazul în care acesta dorește să adune/ scadă/ înmulțească/ împartă două polinoame. Se observă, așadar, că în cazul celor patru operații matematice menționate anterior, programul urmează aceiași pași.

Erorile care pot apărea în acest scenariu de utilizare, sunt: utilizatorul nu introduce nimic sau introduce spații goale în câmpul destinat unui polinom; în acest caz, un mesaj de atenționare este afișat, care va dispărea după 3 secunde. Dacă utilizatorul introduce o formă invalidă de polinom, de exemplu: *5a + x, 2x^ + 4x* ș.a.m.d, programul semnalează acest lucru printr-un mesaj de atenționare.

În cazul în care utilizatorul selectează operația de împărțire, dar al doilea polinom este nul (polinom format dintr-un monom cu coeficientul zero), în locul destinat afișării rezultatului operației va apărea mesajul *"Division by zero error!"*.

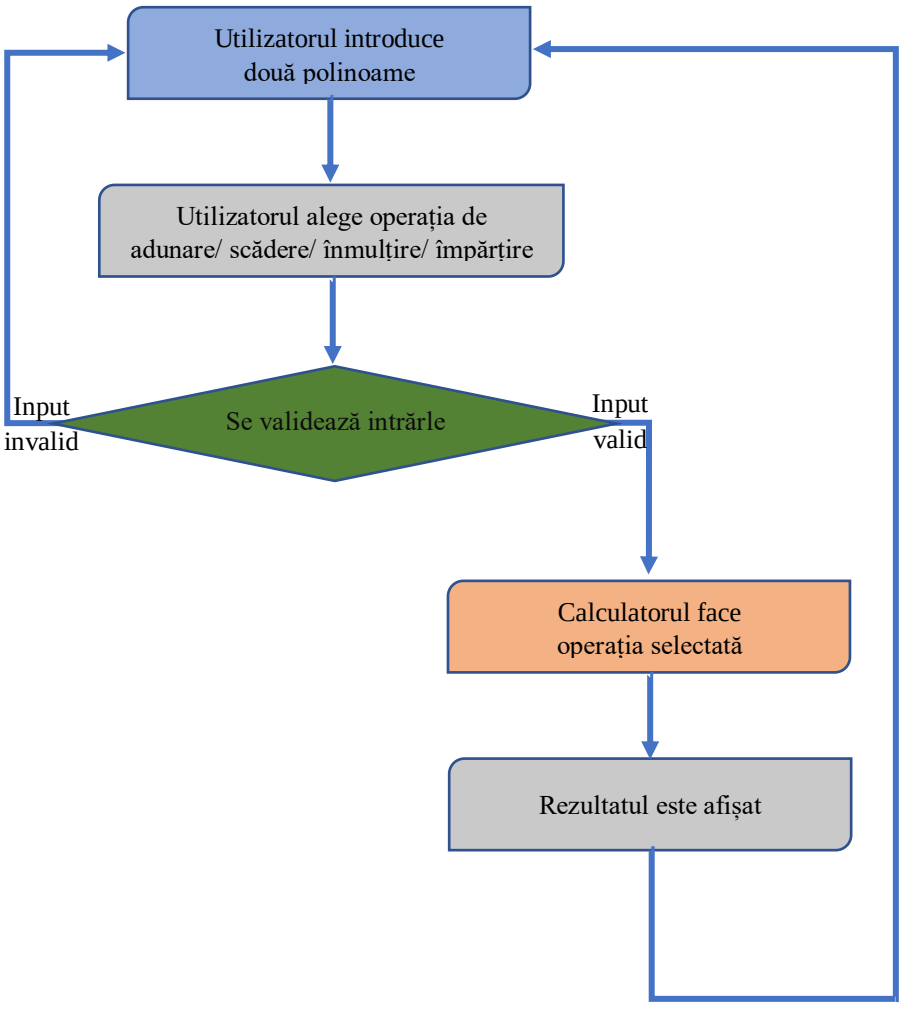


Figura 2.3.1

2.3.2. Derivarea sau împărțirea polinoamelor

Figura 2.3.1. descrie modul în care se va desfășura interacțiunea utilizatorului cu programul în cazul în care acesta dorește să deriveze sau să integreze un polinom. Se observă, așadar, că în cazul celor două operații matematice menționate anterior, programul urmează aceiași pași. Diagrama este asemănătoare cu cea prezentată anterior, însă, în acest caz, utilizatorul trebuie să introducă polinomul dorit în prima zonă de text.

Erorile care pot apărea în acest scenariu de utilizare, sunt: utilizatorul nu introduce nimic sau introduce spații goale în câmpul destinat polinomului; în acest caz, un mesaj de atenționare este afișat, care va dispărea după 3 secunde. Dacă utilizatorul introduce o formă invalidă de polinom, de exemplu: *5a + x, 2x^ + 4x* ș.a.m.d, programul semnalează acest lucru printr-un mesaj de atenționare.

În cazul în care utilizatorul selectează operația de integrare, dar polinomul introdus este format dintr-un monom care are exponentul '-1', programul își va întrerupe execuția, iar în câmpul destinat ieșirii apărând mesajul *"ln(x)+C"*. Se observă așadar că la rezultat a fost precizată și mulțimea constantelor.

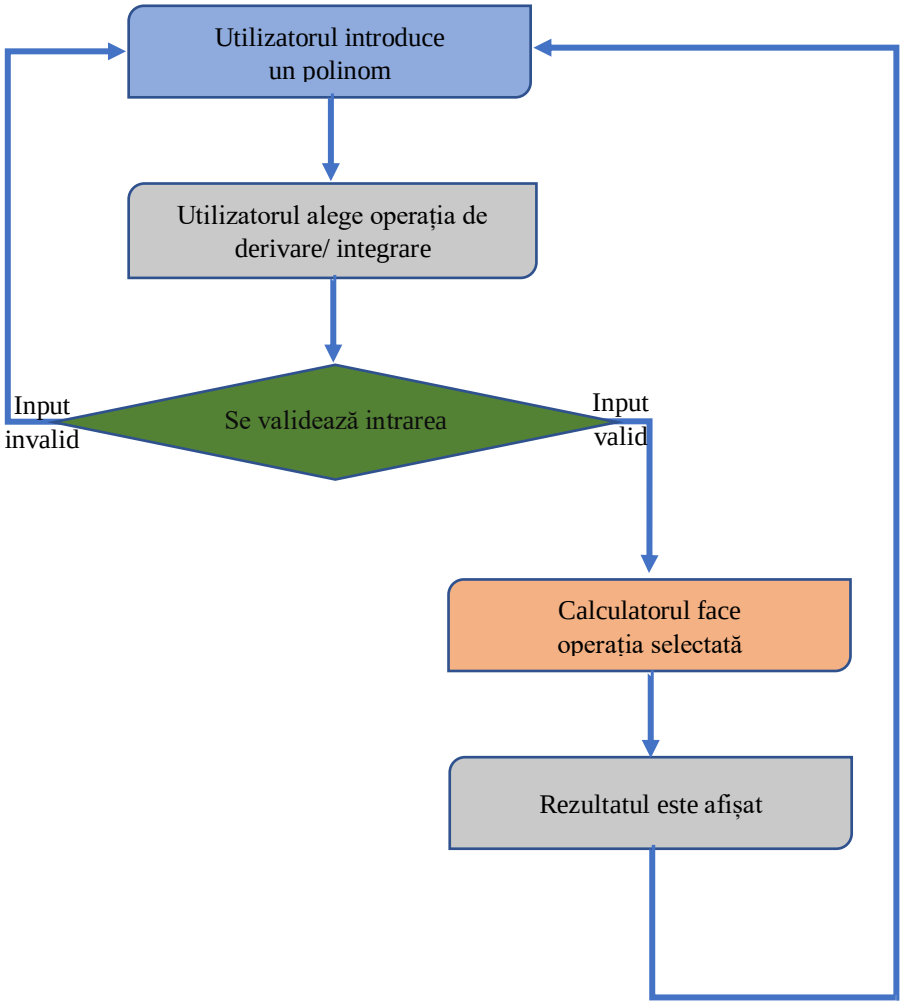


Figura 2.3.2

3. Proiectare

3.1. Concepte ale programării orientate pe obiect folosite

Aplicația creată a fost implementată în limbajul de programare Java. Fiind dezvoltată într-un limbaj de programare orientat pe obiecte, aplicația modelează obiecte, precum *Monomial* sau *Polynomial*, utilizează concepte ale programării orientate pe obiect, precum moștenirea (concret, avem o clasă abstractă *Monomial*; deoarece intrările vor fi formate din monoame cu coeficienți întregi, dar rezultatul poate avea coeficienți reali, au fost create două clase concrete, *IntegerMonomial* și *DoubleMonomial*, ce moștenesc din clasa părinte *Monomial*; asemănător s-a procedat pentru clasa *Polinomial*), polimorfismul, suprascrierea metodelor (clasele concrete *IntegerMonomial* și *DoubleMonomial*), supraîncărcarea metodelor (în special a constructorilor), *implementarea de interfețe* (în cadrul claselor "ascultători", spre exemplu *IntegrationListener*, care implementează interfața *ActionListener*, utilă pentru interacțiunea utilizatorului cu interfața grafică).

Un obiect de tip polinom este alcătuit dintr-o listă de obiecte de tip monom. Profitând de limbajul în care a fost scrisă aplicația, au fost folosite liste, și nu array-uri pentru a defini obiectele menționate.

De asemenea, aplicația a fost dezvoltată după pattern-ul arhitectural model-view-controller, care va fi prezentat în cele ce urmează.

3.2. Pachetul Model

În figura 3.2 este reprezentată structura pachetului *Model*. Clasa de bază o reprezintă clasa *Monomial*. Clasele *IntegerMonomial* și *DoubleMonomial* moștenesc din *Monomial*. Apoi avem clasa *Polynomial*. Clasele *IntegerPolynomial* și *DoublePolynomial* moștenesc din *Polynomial*. Clasa *Polynomial* conține o listă de *Monomial*, urmând mai apoi ca fiecare copil care moștenește din clasa *Polynomial* să conțină o listă de monoame de tipul lui (*IntegerPolynomial* va conține o listă de *IntegerMonomial*, iar *DoublePolynomial* va conține o listă de *DoubleMonomial*).

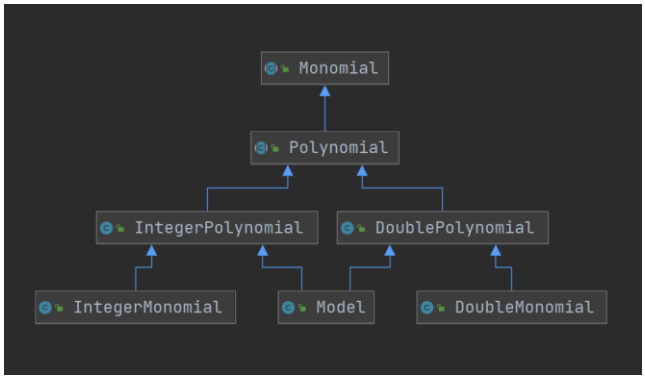


Figura 3.2

3.3. Pachetul View

Pe lângă clasa *View*, pachetul mai conține clasa *PolynomialPanel*, care conține obiecte swing specifice fiecărei zone de input, dar care va fi descrisă anterior.

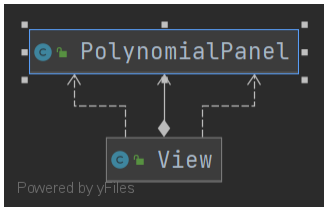


Figura 3.3

3.4. Pachetul Controller

În acest pachet sunt conținute clasele "ascultător", cele care controlează interacțiunea dintre utilizator și interfața grafică, și clasa *Controller*, care distribuie fiecărui buton o anumită funcție.

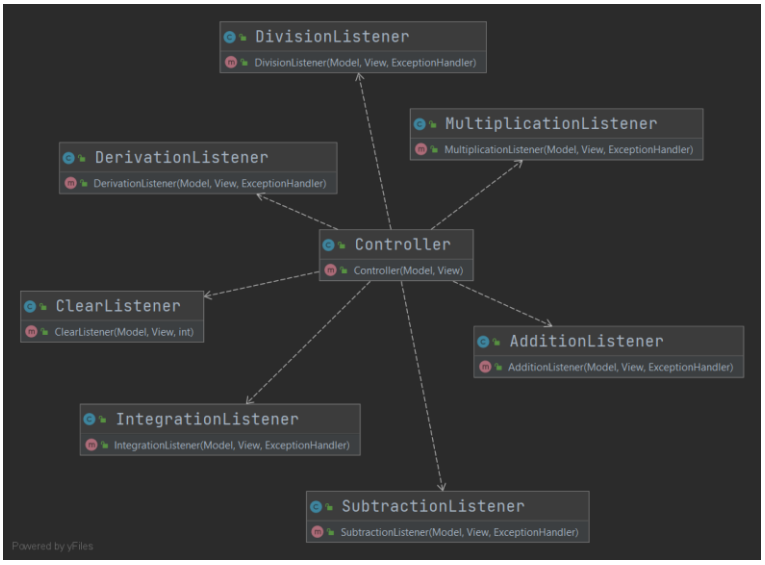


Figura 3.4

3.5. Pachetul Validator

Acest pachet conține clasele responsabile de modelarea polinomului și aducerea acestuia la o formă în care poate fi spart mai ușor în liste de monoame. Mai sunt conținute și clasele excepții, care vor fi aruncate în cazul unor erori.

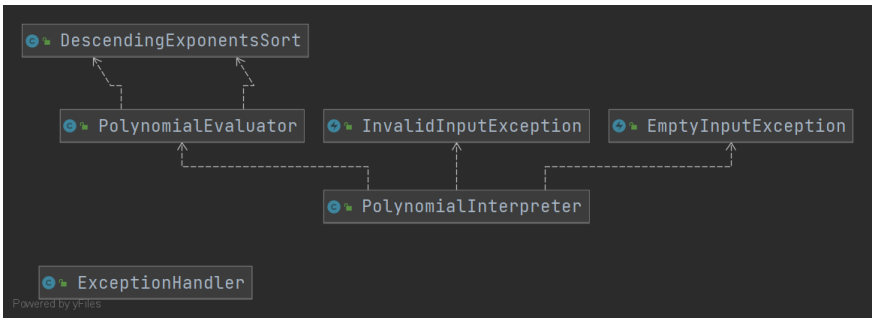


Figura 3.5

4. Implementare

În cele ce urmează, vor fi descrise clasele folosite de aplicație și modul lor de funcționare. Pentru o mai bună organizare, acestea vor fi descrise în cadrul secțiunii pachetului din care acestea fac parte

4.1. Pachetul View

Pachetul conține două clase: clasa *View* și clasa *PolynomialPanel*.

Clasa *View* este responsabilă pentru afișarea interfeței grafice. Clasa conține constante String și int, pentru a evita folosirea "numerelor magice" în program, apoi o listă de panouri și butoane. Construcția ferestrei a fost împărțită în mai mulți pași, fiecare pas fiind dictat de constructorul clasei. Spre exemplu, *void setUpFrame()* este responsabilă de a pregăti JFrame-ul (fereastra aplicației), setând dimensiunea ei, titlul ș.a.m.d., astfel că fiecare funcție face ceea ce conține în nume.

Sunt adăugate, rând pe rând, partea din stânga a interfeței, cea care conține imaginea legată de tematica aplicației, informații despre autor și apoi, titlul aplicației.

Urmează partea interesantă. Observând că fiecare zonă corespunzătoare unui polinom conține aceleași informații, am creeat clasa *PolynomialPanel*, care extinde clasa JPanel. Această clasă conține un text, ce descrie scopul zonei de input de mai jos (de exemplu, *"Enter first polynomial"*), zona de input, un câmp destinat afișării mesajelor de alertă, iar în lateral, un buton, pentru a curăța zona de text la apăsarea acestuia. Pentru a face mesajul de alertă să dispară după un anumit timp, a fost folosită metoda *schedule()* din clasa *Executor* și o funcție lamda, echivalentul funcției *setTimeout()* din JavaScript.

Pe lângă zonele dedicate polinoamelor, interfața grafică mai conține șase butoane, ce reprezintă operațiile matematice care se vor executa asupra polinoamelor. Fiecărui buton i-a fost asignată o imagine represzentativă.

În final, pe lângă getteri și setteri, sunt prezente metodele care vor fi folosite de *Controller* pentru a adăuga funcționalitate butoanelor.

4.2. Pachetul Model

Pachetul conține două subpachete, *monomial* și *polynomial*, care conține obiectele principale ale acestei aplicații. Clasa abstractă *Monomial* conține două variabile instanță generice, ce vor fi înlocuite cu *Integer*, respectiv *Double*. Clasele concrete *IntegerMonomial* și *DoubleMonomial* extind clasa *Monomial*, ele fiind obiectele concrete pe care le vom modela și conțin diferiți constructori, necesari anumitor situații. La fel am procedat și pentru clasa *Polynomial*, care are o listă de generice, ce va conține ulterior *IntegerMonomials* sau *DoubleMonomials*.

Clasa *Monomial* va prelucra obiectele mai sus menționate, ea conținând logica din spatele aplicației, cea care execută operațiile matematice. Operația de adunare, *public void addPolynomials()*, va adăuga lista de monoame a celui de-al doilea polinom la lista de monoame a primului polinom, urmând ca mai apoi restrângerea polinomului și aranjarea listei sale de monoame în ordinea descrescătoare a exponenților, metode ce vor fi descrise în cadrul capitolului *„4.4. Pachetul Validator"*. Pentru scădere se procedează aproximativ la fel, în sensul că ne folosim de proprietatea matematică: **a – b = a + (-b)**, așadar la lista de monoame a primului polinom vom adăuga lista de monoame a celui de-al doilea polinom, dar cu semn schimbat.

Operația de înmulțire se face clasic, înmulțind fiecare monom din primul polinom cu fiecare monom din al doilea polinom, la final având loc operația de reducere a coeficienților. Pentru derivare și integrare sunt derivate, respectiv integrate fiecare monom din primul polinom, după legile matematice bine definite, anume **d/dx (a*x^n) = a*n*x^(n+1)**, respectiv **S (a*x^n) dx = a/(n+1) * x^(n+1) + C**.

Am lăsat la final operația de împărțire a polinoamelor, *public void dividePolynomials()*, deoarece aceasta necesită mai multă atenție. Pentru aceasta, vom trece în revistă algoritmul de împărțire a două polinoame:

```
Fie n – deîmpărțit, d – împărțitor, q – cât și r – rest. Atunci rezultatul
împărțirii lui n la d se calculează astfel:
q <- 0
r <- n
cât timp r != 0 ȘI grad(r) >= grad(d) execută
      t <- r / d
      q <- q + t
      r <- r – t * d
afișează q, r
```

Așadar, se împarte monomul cu gradul cel mai mare din primul termen (conținut în rest) la monomul cu gradul cel mai mare din al doilea termen. Monomul rezultat va fi adăugat la cât. Din primul termen (rest), vom scădea produsul dintre monomul rezultat și al doilea operand. Procesul continuă până când deîmpărțitul va avea grad mai mare decât împărțitorul. Restul operației va fi polinomul corespunzător primului termen care nu poate fi împărțit la împărțitor.

4.3. Pachetul Controller

Este pachetul care conține clasele ce vor asigura interacțiunea utilizatorului cu programul. Clasa principala, *Controller*, asignează fiecărui buton din interfața grafică, o utilizare. Așadar, fiecare buton din program are câte o clasă ce corespunde unui "ascultător" diferit. *ClearListener* asigură, în momentul în care utilizatorul dorește să șteargă un polinom, ștergerea acelui polinom din "baza de date" (din model) și golirea câmpului corespunzător (din view).

Pentru fiecare astfel de "ascultător", se procedează astfel: din input-ul userului se vor forma polinoamele folosind metoda *void parseString()* (detaliată în următorul capitol), se vor "introduce în baza de date" (în model, în variabilele rezervate pentru input), se va apela metoda care efectuează operația selectată, se va apela metoda *void parseValue()* (detaliată în urmptorul capitol), iar apoi se va afișa rezultatul în câmpul destinat acestui lucru. În caz de eroare, utilizatorul va fi înștiințat.

4.4. Pachetul Validator

În clasa *PolynomialInterpreter*, un String va fi transformat într-un obiect de tip polinom, și invers. Funcția *void parseString()* va modela string-ul primit de utilizator într-un obiect de tip polinom. După ce s-a verificat că zona de text nu este goală, se transformă input-ul într-un string pe care expresia regulată îl poate recunoaște ca fiind un polinom valid. Rolul funcției *String stringToPolynomialString(String buffer)* este să aducă acel text *buffer*, din forma "5x^2+x-6", de exemplu, la forma "5x^2+1x^1-6x^0". Așa, expresia regulata va identifica numărul dinaintea lui 'x' ca fiind coeficientul, iar cel de după '^', ca fiind exponentul. Cealaltă metodă, *String parseValue(DoublePolynomial buffer),* va modela un obiect polinom într-un String, pentru afișare. Pentru fiecare monom, se va concatena la rezultat semnul coeficientului, apoi coeficientul cu două zecimale sau valoarea sa întreagă, după caz, urmat de 'x^' exponent, dacă acesta există.

Cele două metode din clasa *PolynomialEvaluator* aduc polinomul la o formă cu care se poate lucra mai ușor. Metoda *void sortByExponents()* va ordona lista de monoame a polinomului descrescător după exponent, iar metoda *void reduceCoefficients()* va aduna monoamele care au același exponent și le va șterge pe cele care au coeficientul nul.

Tot în acest pachet au fost definite excepțiile care pot apărea, generate de inputul gol sau cel invalid, care nu respectă forma unui polinom, dar și clasa *ExceptionHandler* care are rol de a afișa în interfața grafică eventualele mesaje de eroare.

5. Rezultate

Pentru a testa aplicația și a vedea că aceasta execută corect operațiile matematice, am folosit JUnit. Scopul acestuia este să testeze fiecare operație matematică și să semnaleze eventualele erori. În fiecare test, se creează două polinoame care reprezentă posibile inputuri ale utilizatorului. După ce se apelează metoda care execută operația matemaică, se compară polinomul rezultat cu rezultatul așteptat. Dacă acestea două diferă, acest lucru este semnalat cu ajutorul metodei *assert*.

```
@Test
void addPolynomials() {
        firstOperand.addMonomialToList(new IntegerMonomial(2, 3));
        firstOperand.addMonomialToList(new IntegerMonomial(1, 2));
        firstOperand.addMonomialToList(new IntegerMonomial(3, 0));
        appModel.setFirstOperand(firstOperand);

        secondOperand.addMonomialToList(new IntegerMonomial(1, 3));
        secondOperand.addMonomialToList(new IntegerMonomial(2, 2));
        secondOperand.addMonomialToList(new IntegerMonomial(4, 0));
        appModel.setSecondOperand(secondOperand);

        appModel.addPolynomials();
        resultTerm = appModel.getResultTerm();

        DoublePolynomial expectedResult = new DoublePolynomial();
        expectedResult.addMonomialToList(new DoubleMonomial(3, 3));
        expectedResult.addMonomialToList(new DoubleMonomial(3, 2));
        expectedResult.addMonomialToList(new DoubleMonomial(7, 0));

        assertTrue(resultTerm.equalsPolynomial(expectedResult),
AdditionListener.ADDITION_SUCCESS);
}
```

În urma testelor, se constată că implementarea operațiilor a fost făcută în mod corespunzător.

```
ModelTest.....40ms
.........passed.....dividePolynomials().....32ms
.........passed.....integratePolynomial().....2ms
.........passed.....multiplyPolynomials().....2ms
.........passed.....subtractPolynomials().....1ms
.........passed.....addPolynomials().....2ms
.........passed.....derivePolynomial().....2ms
```

6. Concluzii

Operațiile pe polinoame pe hârtie necesită cunoștințe matematice, atenție și timp. Dacă lipsește un element din cele trei menționate anterior, rezultatul procedeului va fi, cel mai probabil, eronat. Așadar, această aplicație este potrivită pentru a efectua operații matematice pe polinoame, ținând cont că ea arată și funcționează impecabil.

În urma acestei teme, consider că mi-am îmbunătățit stilul de programare, respectând convențiile de numire Java sau de organizare a codului în pachete și subpachete. De asemenea, mi-am consolidat cunoștințele deja existente legate despre programarea orientată pe obiecte, precum moștenirea, polimorfismul sau elementele de interfață grafică din pachetul Java.swing, punând bazele unor cunoștințe noi de Java, cum ar fi tipurile generice.

În momentul de față, aplicația permite operații pe polinoame cu coeficienți întregi. Aceasta ar putea fi dezolvată ulterior, permițând utilizatorului să introducă și polinoame cu coeficienți reali. O altă posibilă dezvoltare ulterioară a acestei aplicații, ar fi vizualizarea într-un sistem cartezian xOy funcția corespunzătoare polinomului introdus, evidențiind rădăcinile acesteia. O versiune viitoare a acestui program ar putea dispune de opțiunea de a evalua polinomul într-un punct dat.

Pentru cei mai competitivi, aplicația ar putea lua forma unei provocări. După ce utilizatorul introduce cele două polinoame de evaluat, acesta își setează un timp, în secunde, rezolvă operația cu polinoame pe hârtie, iar la momentul expirării timpului rezultatul evaluării va fi afișat pe ecran, urmând ca, mai apoi, utilizatorul să își verifice rodul muncii sale. Pe de altă parte, aplicația ar putea avea o zonă de text nouă, în care utilizatorul va introduce rezultatul obținut de el în urma efectuării operației cu polinoame pe hârtie, iar aplicația va semnala dacă rezultatul este corect sau nu.

7. Bibliografie

https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

https://medium.com/factory-mind/regex-tutorial-a-simple-cheatsheet-by-examples-649dc1c3f285

https://regex101.com/

https://www.baeldung.com/java-generics

https://www.mathsisfun.com/algebra/polynomials-division-long.html

https://www.vogella.com/tutorials/JUnit/article.html