

## Documentatie TEMA1

### 1. Obiectivul temei:

Se va proiecta si se va implementa un calculator pentru operatiile pe polinoame. Se va crea o interfata grafica in care utilizatorul va putea introduce polinoamele, va putea selecta operatia matematica dorita a fi realizata ( adunare, scadere, inmultire, impartire, derivare si integrare ) si va putea vedea rezultatele obtinute.

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

Pentru a rezolva problema am considerat un polinom ca fiind o lista de monoame. Monoamele fiind polinoame cu un singur termen care au asadar un coeficient si un grad. Am considerat ca pot exista monoame cu coeficienti intregi (int) sau monoame cu coeficienti reali (double).

O metoda de rezolvare a problemei este de a folosi o clasa generica polinom iar un polinom sa fie definit ca o lista de monoame. Un monom va fi o clasa abstracta care va fi parinte pentru o clasa monom care va avea coeficienti intregi ( numita IntMonom ) si o clasa monom care va avea coeficienti reali ( double in cazul nostru, numita DoubleMonom ). Asadar clasa Monom va avea ca si variabila instanta o variabila ce va defini gradul monomului iar clasele fiice ale acesteia vor avea fiecare o variabila instanta care va defini valoarea coeficientului. In functie de tipul de monom vom da si tipul coeficientului ( acestea trebuie sa coincida ).

Am presupus ca datele de intrare pentru polinoamele reale vor fi de forma: [semn][parteIntreagaCoeficient].[parteFractionalaCoeficient][x sau X]^ [exponent] ex:  $2.0x^2 + 3.76x^3$  iar pentru polinoamele cu coeficienti numere intregi am presupus ca datele de intrare pentru polinoame vor fi de forma: [semn] [coeficient] [x sau X] ^ [exponent] ex:  $2x^2 + x^3$ . De remarcat este faptul ca pentru coeficientii egali cu 1 sau 1.0 acestia vor trebui sa fie scrisi, altfel monomul va fi ignorat la executarea operatiei. Similar este si pentru monoamele de grad 0, gradul acestora va trebui precizat, de exemplu:

### 3. Proiectare ( decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator )

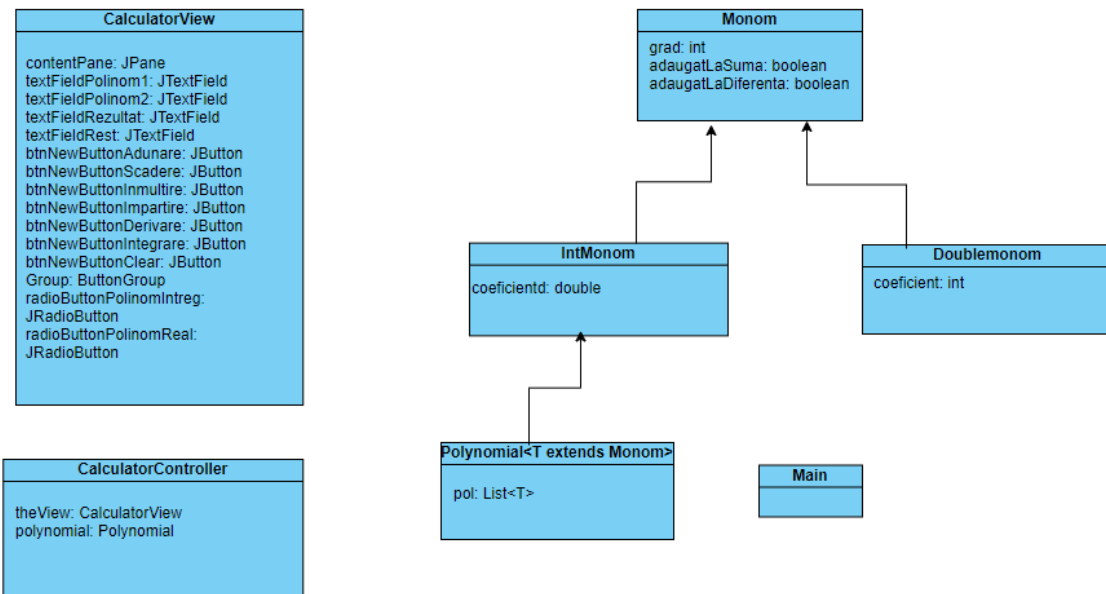
Structurile de date utilizate: List si ArrayList pentru a stoca obiecte de tipul IntMonom si DoubleMonom.

Pentru proiectare am folosit clasa generica Polynomial<T> unde T trebuie sa extinda clasa Monom.

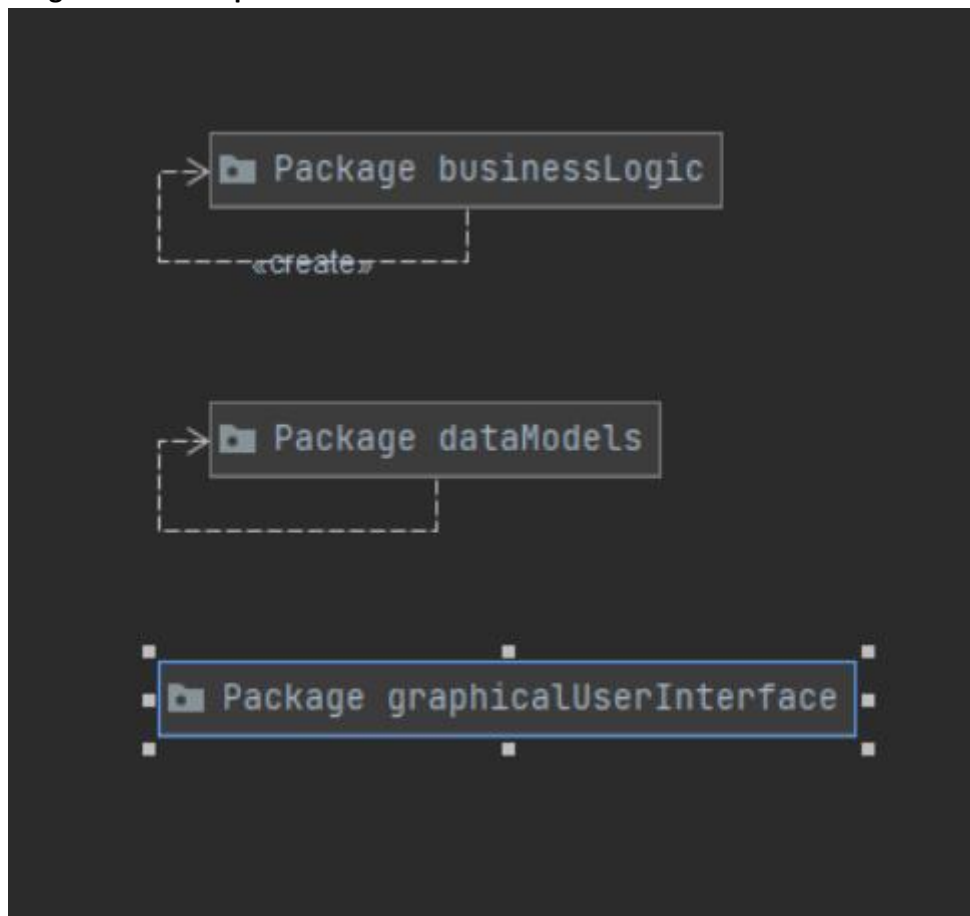
Am creat trei pachete pentru a reprezenta mai bine modelul arhitectural Model View Controller: businessLogic ( in care am stocat partea pentru Controller ), dataModels ( in care am stocat partea pentru Model ) si nu in ultimul rand graphicalUserInterface ( pentru a stoca tot ce tine de design-ul interfetei ). Clasele le-am adaugat in pachete astfel:

- In businessLogic am adaugat: CalculatorController
- In dataModels am adaugat: Polynomial, Monom, IntMonom, DoubleMonom, Main
- In graphicalUserInterface am adaugat: CalculatorView

## Diagrama UML de clase:



## Diagrama UML de pachete:



#### 4. Implementare

##### Clasa Polynomial ( Model )

Este modelul acestui proiect in care am implementat toate operatiile cerute. Un polinom poate avea coeficienti reali sau coeficienti double de aceea am facut aceasta clasa generica, astfel incat sa difere doar tipul coeficientilor monoamelor din lista polinomului, pentru a evita crearea mai multor clase similare. In aceasta clasa am implementat metodele pentru adunare, scadere, inmultire, impartire, derivare, integrare, extragere coeficienti (folosind regex) cat si cateva metode necesare pentru rezolvarea celor precedente cum ar fi:

```
public int gradPolinom(Polynomial<DoubleMonom> p)
( pentru returnarea gradului polinomului );
```

```
public void printPolinom(Polynomial<T> p)
( pentru afisarea unui polinom );
```

```
public String polinomToString(Polynomial<IntMonom> p)
( pentru transformarea unui polinom intr-un sir de caractere );
```

```
public void sortareCoefDescr(Polynomial<T> p)
( pentru ordonarea descrescatoare a termenilor polinomului in functie de gradul monoamelor );
```

Toate operatiile au fost implementate pentru polinoame cu coeficienti reali iar pentru polinoame cu coeficienti intregi am implementat toate operatiile cu exceptia celei de impartire. Metodele a caror denumire se termina cu litera „d” sunt metode dedicate polinoamelor cu coeficienti reali de exemplu:

```
public Polynomial<DoubleMonom> sumaPolinoamed(Polynomial<DoubleMonom> p1,
Polynomial<DoubleMonom> p2)
( pentru a calcula suma a doua polinoame transmise ca si parametru );
```

```
public void printPolinomd(Polynomial<T> p)
( pentru afisarea unui polinom cu coeficienti reali );
```

```
public String polinomToStringd(Polynomial<DoubleMonom> p)
( pentru transformarea unui polinom cu coeficienti reali intr-un string );
```

```
public Polynomial diferentaPolinoamed(Polynomial<DoubleMonom> p1,
Polynomial<DoubleMonom> p2) {
( pentru calcularea diferentei a doua polinoame cu coeficienti reali );
```

```
public Polynomial<DoubleMonom> regexd(String s)
( pentru extragerea coeficientilor unui polinom cu coeficienti reali si pentru transformarea unui sir de caractere intr-un polinom -> in aceasta metoda am folosit functionalitatile din bibliotecile java.util.regex.Matcher si java.util.regex.Pattern );
```

Pe langa aceste metode mai avem si:

```
public Polynomial inmultirePolinoame(Polynomial<DoubleMonom> p1,
Polynomial<DoubleMonom> p2)
( pentru inmultirea a doua polinoame cu coeficienti reali );
```

```
public Polynomial derivarePolinoame(Polynomial<DoubleMonom> p)
( pentru derivarea unui polinom cu coeficienti reali );
```

```
public Polynomial integrarePolinoame(Polynomial<DoubleMonom> p)
( pentru integrarea unui polinom cu coeficienti reali );
```

```
public void impartirePolinoame (Polynomial<DoubleMonom> p1,
Polynomial<DoubleMonom> p2 , Polynomial<DoubleMonom> rest ,
Polynomial<DoubleMonom> cat) throws Exception
```

( pentru impartirea a doua polinoame cu coeficienti reali );

Pentru polinoamele cu coeficienti intregi avem definite metodele:

```
public Polynomial sumaPolinoame (Polynomial<IntMonom> p1,
Polynomial<IntMonom> p2)
```

( pentru suma a doua polinoame cu coeficienti intregi );

```
public Polynomial diferentaPolinoame (Polynomial<IntMonom> p1,
Polynomial<IntMonom> p2)
```

( pentru suma a doua polinoame cu coeficienti intregi );

```
public Polynomial integrarePolinoamei (Polynomial<IntMonom> p)
```

( pentru integrarea unui polinom cu coeficienti intregi );

```
public Polynomial derivarePolinoamei (Polynomial<IntMonom> p)
```

( pentru derivarea unui polinom cu coeficienti intregi );

```
public Polynomial inmultirePolinoamei (Polynomial<IntMonom> p1,
Polynomial<IntMonom> p2)
```

( pentru inmultirea a doua polinoame cu coeficienti intregi );

```
public Polynomial<IntMonom> regexi (String s)
```

( pentru extragerea coeficientilor unui polinom cu coeficienti intregi si pentru transformarea unui sir de caractere intr-un polinom -> si in aceasta metoda am folosit functionalitatile din bibliotecile java.util.regex.Matcher si java.util.regex.Pattern );

## Clasa Monom

Am declarat aceasta clasa ca fiind una abstracta. Aceasta clasa are ca variabile instanta:

- o variabila privata `private int grad`; care reprezinta gradul monomului;

- o variabila privata `private boolean adaugatLaSuma`; care este `true` daca monomul a fost adaugat la la calculul sumei in momentul apelarii uneia dintre metodele `sumaPolinoame ( p1, p2 )` sau `sumaPolinoamed ( p1, p2 )` si `false` daca monomul nu a fost adaugata la calculul sumei in momentul apelarii uneia dintre metodele `sumaPolinoame ( p1, p2 )` sau `sumaPolinoamed ( p1, p2 )`;

- o variabila privata `private boolean adaugatLaDiferenta`; care ia valoarea `true` daca monomul a fost adaugat la calculul diferentei in momentul apelarii uneia dintre metodele `diferentaPolinoame ( p1, p2 )` sau `diferentaPolinoamed ( p1, p2 )` sau ia valoarea `false` daca monomul nu a fost adaugata la calculul diferentei in momentul apelarii uneia dintre metodele `diferentaPolinoame ( p1, p2 )` sau `diferentaPolinoamed ( p1, p2 )`;

## Clasa IntMonom

Mosteneste ( extinde ) clasa `IntMonom` avand in plus fata de aceasta variabila instanta `private int coeficient`; reprezentand coeficientul de tip `int` al monomului;

Am generat setter si getter pentru variabila coeficient si am creat niste constructori pentru aceasta clasa.

### Clasa DoubleMonom

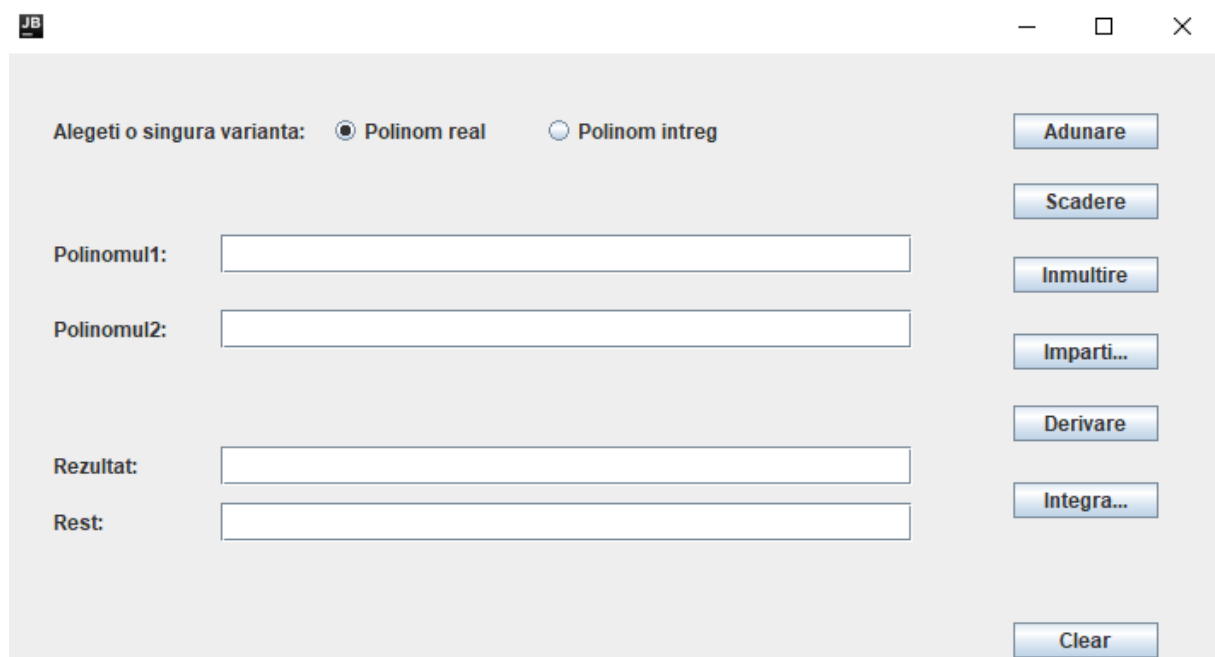
Mosteneste ( extinde ) clasa DoubleMonom avand in plus fata de aceasta variabila instanta `private double coeficientd;` reprezentand coeficientul de tip double al monomului;

Am generat setter si getter pentru variabila coeficient si am creat niste constructori pentru aceasta clasa.

### Clasa CalculatorView ( View )

In aceasta clasa am creat interfata calculatorului.

Pentru realizarea interfetei am adaugat 4 casute de text una pentru primul polinom, a doua pentru al doilea polinom iar a treia si a patra pentru rezultatul operatiei respectiv pentru rest in caz ca operatia este una de impartire.



In partea dreapta am pus butoanele responsabile cu executarea operatiei data de numele butonului si afisarea rezultatului obtinut in urma executarii operatiei respective.

In partea de sus am creat doua butoane radio care sa ne anunte ce fel de polinoame urmeaza sa introducem in casetele pentru Polinomul1 si Polinomul2 pentru a stii ce fel de metoda sa folosim pentru a transforma string-urile din interiorul casutelor in polinoame regex ( pt polinoamele cu coeficienti intregi ) sau regexd ( pt polinoamele cu coeficienti reali ).

Ultimul buton, anume butonul Clear se foloseste pentru a sterge ceea ce avem in casutele de Rezultat si de Rest pentru a usura munca utilizatorului si pentru a evita posibile confuzii.

## Clasa CalculatorController ( Controller )

In aceasta clasa am adaugat functionalitatile butoanelor, implementand ActionListeneri pentru toate butoanele.

Tot aici am creat si clasele:

- **AdunareListener**: care implementeaza interfata ActionListener. Aceasta suprascrie metoda din interfata *ActionListener* `public void actionPerformed(ActionEvent actionEvent)` care preia datele de intrare din Polinom1 si din Polinom2, le proceseaza si face transformarea din String in polinom daca polinomul este introdus in forma corecta, definita mai sus, le prelucreaza, face operatia de adunare definita in clasa Polynomial in functie de tipul polinomului selectat, converteste apoi rezultatul inapoi la String si il afiseaza in casuta Rezultat.

```
class AdunareListener implements ActionListener
```

( pentru adaugarea actiunii butonului de adunare din interfata grafica );

- **ScadereListener**: care implementeaza interfata ActionListener. Aceasta suprascrie metoda din interfata *ActionListener* `public void actionPerformed(ActionEvent actionEvent)` care preia datele de intrare din Polinom1 si din Polinom2, le proceseaza si face transformarea din String in polinom daca polinomul este introdus in forma corecta, definita mai sus, le prelucreaza, face operatia de scadere definita in clasa Polynomial in functie de tipul polinomului selectat, converteste apoi rezultatul inapoi la String si il afiseaza in casuta Rezultat.

```
class ScadereListener implements ActionListener
```

( pentru adaugarea actiunii butonului de scadere din interfata grafica );

- **InmultireListener**: care implementeaza interfata ActionListener. Aceasta suprascrie metoda din interfata *ActionListene* `public void actionPerformed(ActionEvent actionEvent)` care preia datele de intrare din Polinom1 si din Polinom2, le proceseaza si face transformarea din String in polinom daca polinomul este introdus in forma corecta, definita mai sus, le prelucreaza, face operatia de inmultire definita in clasa Polynomial in functie de tipul polinomului selectat, converteste apoi rezultatul inapoi la String si il afiseaza in casuta Rezultat.

```
class InmultireListener implements ActionListener
```

( pentru adaugarea actiunii butonului de inmultire din interfata grafica );

- **ImpartireListener**: care implementeaza interfata ActionListener. Aceasta suprascrie metoda din interfata *ActionListene* `public void actionPerformed(ActionEvent actionEvent)` care preia datele de intrare din Polinom1 si din Polinom2, le proceseaza si face transformarea din String in polinom daca polinomul este introdus in forma corecta, definita mai sus, le prelucreaza, face operatia de impartire definita in clasa Polynomial, converteste apoi rezultatul inapoi la String si il afiseaza in casuta Rezultat.

```
class ImpartireListener implements ActionListener
```

( pentru adaugarea actiunii butonului de impartire din interfata grafica );

- **DerivareListener**: care implementeaza interfata ActionListener. Aceasta suprascrie metoda din interfata *ActionListene* `public void actionPerformed(ActionEvent actionEvent)` care preia datele de intrare din Polinom1, le proceseaza si face transformarea din String in polinom daca polinomul este introdus in forma corecta, definita mai sus, le prelucreaza, face operatia de derivare definita in clasa Polynomial in functie de tipul polinomului selectat, converteste apoi rezultatul si restul obtinut inapoi la String si il afiseaza in casuta Rezultat respectiv in casuta pentru Rest .

```
class DerivareListener implements ActionListener
```

( pentru adaugarea actiunii butonului de derivare din interfata grafica );

- **IntegrareListener**: care implementeaza interfata ActionListener. Aceasta suprascrie metoda din interfata *ActionListene* `public void actionPerformed(ActionEvent actionEvent)` care preia datele de intrare din Polinom1, le proceseaza si face transformarea din String in polinom daca polinomul este introdus in forma corecta, definita mai sus, le prelucreaza, face operatia de integrare definita in clasa Polynomial in functie de tipul polinomului selectat, converteste apoi rezultatul inapoi la String si il afiseaza in casuta Rezultat.

```
class IntegrareListener implements ActionListener
```

( pentru adaugarea actiunii butonului de integrare din interfata grafica );

- **ClassListener**: care goleste campurile Rezultat si Rest in urma apasarii butonului din interfata

```
class ClearListener implements ActionListener
```

( pentru adaugarea actiunii butonului de Clear din interfata grafica );

## Main

In aceasta clasa am facut legatura intre model view si controller si am rulat efectiv programul pentru deschiderea ferestrei aplicatiei create.

## 5. Rezultate

Testarea adunarii a doua polinoame cu coeficienti reali:

Testarea scaderii a doua polinoame cu coeficienti reali:

JB

—

□

×

Alegeti o singura varianta: ☒ Polinom real ☐ Polinom intreg

Polinomul1:

1.0x^2+4.0x^1+4.0x^0

Polinomul2:

1.0x^1+2.0x^0

Rezultat:

+1x^2+3x^1+2x^0

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

Testarea inmultirii a doua polinoame cu coeficienti reali:

JB

—

□

×

Alegeti o singura varianta: ☒ Polinom real ☐ Polinom intreg

Polinomul1:

1.0x^2+4.0x^1+4.0x^0

Polinomul2:

1.0x^1+2.0x^0

Rezultat:

+1x^3+6x^2+12x^1+8x^0

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear



Testarea impartirii a doua polinoame cu coeficienti reali fara rest:

JB

—

□

×

Alegeti o singura varianta: ☒ Polinom real ☐ Polinom intreg

Polinomul1:

1.0x^2+4.0x^1+4.0x^0

Polinomul2:

1.0x^1+2.0x^0

Rezultat:

+1x^1+2x^0

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

Testare impartire doua polinoame cu coeficienti reali cu rest:

JB

—

□

×

Alegeti o singura varianta: ☒ Polinom real ☐ Polinom intreg

Polinomul1:

1.0x^2+4.0x^1+4.0x^0

Polinomul2:

3.0x^1+2.0x^0

Rezultat:

+0,33x^1+1,11x^0

Rest:

+1,78x^0

Adunare

Scadere

Inmultire

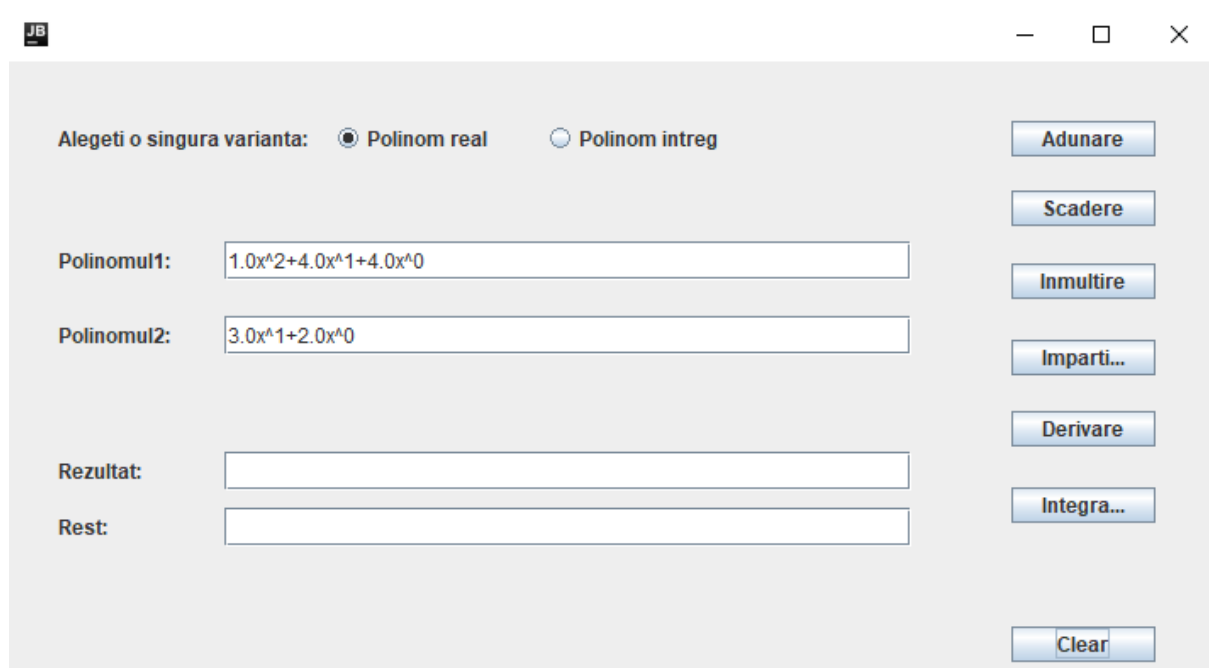
Imparti...

Derivare

Integra...

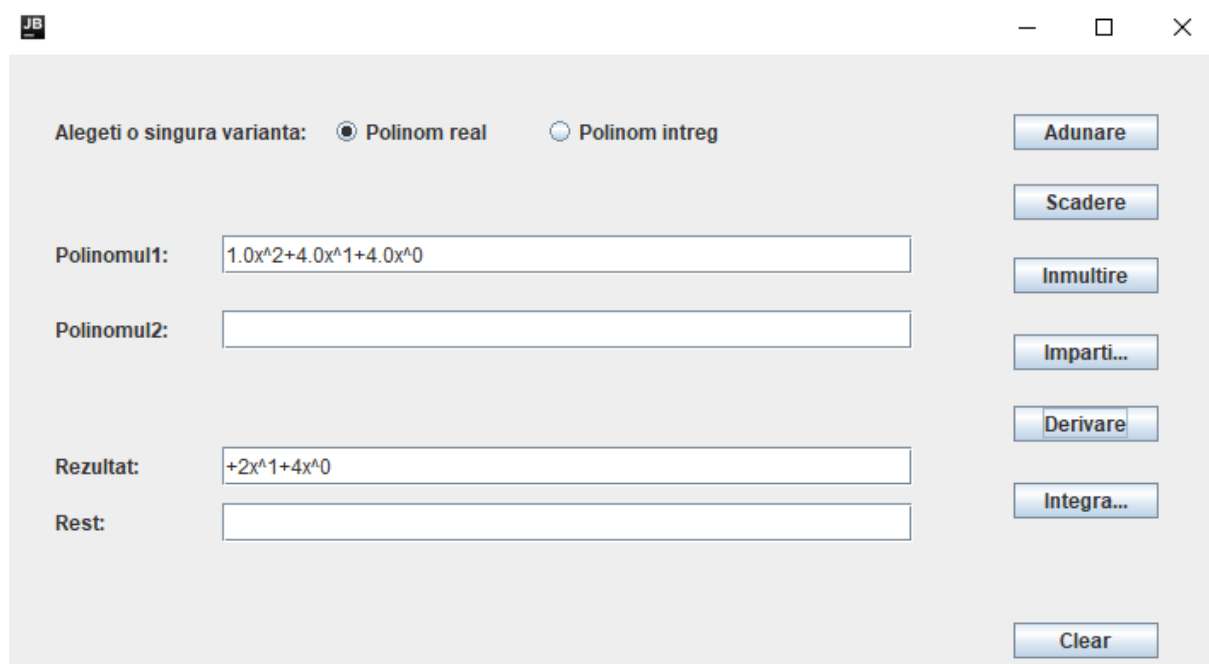
Clear

Testarea butonului de Clear:



A screenshot of a software window titled "JB" with standard Windows window controls (minimize, maximize, close). The window contains a polynomial calculator interface. At the top, it says "Alegeti o singura varianta:" followed by two radio buttons: "Polinom real" (which is selected) and "Polinom intreg". To the right of these are several buttons: "Adunare", "Scadere", "Inmultire", "Imparti...", "Derivare", "Integra...", and "Clear". The "Clear" button at the bottom right is highlighted with a blue border. On the left side, there are labels for "Polinomul1:", "Polinomul2:", "Rezultat:", and "Rest:", each followed by a text input field. The first field contains the polynomial  $1.0x^2+4.0x^1+4.0x^0$ , and the second field contains  $3.0x^1+2.0x^0$ . The "Rezultat:" and "Rest:" fields are currently empty.

Testarea derivarii unui polinom cu coeficienti reali:



A screenshot of the same software window after the "Derivare" button has been clicked. The "Clear" button is no longer highlighted. The "Rezultat:" text input field now contains the derivative of the first polynomial,  $+2x^1+4x^0$ . All other elements, including the "Polinomul1:" field, the "Polinomul2:" field, and the other buttons, remain in the same state as in the previous screenshot.

Testarea integrării unui polinom cu coeficienti reali:

JB

— □ ×

Alegeti o singura varianta: ☒ Polinom real ☐ Polinom intreg

Polinomul1:

Polinomul2:

Rezultat:

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

Testarea adunării a doua polinoame cu coeficienti întregi:

JB

— □ ×

Alegeti o singura varianta: ☐ Polinom real ☒ Polinom intreg

Polinomul1:

Polinomul2:

Rezultat:

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

Testarea scaderii a doua polinoame cu coeficienti intregi:

JB

— □ ×

Alegeti o singura varianta: ☐ Polinom real ☒ Polinom intreg

Polinomul1:

Polinomul2:

Rezultat:

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

Testarea inmultirii a doua polinoame cu coeficienti intregi:

JB

— □ ×

Alegeti o singura varianta: ☐ Polinom real ☒ Polinom intreg

Polinomul1:

Polinomul2:

Rezultat:

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

Testarea butonului de Clear:

The screenshot shows a window titled "JB" with standard Windows window controls (minimize, maximize, close). Inside the window, there is a section "Alegeti o singura varianta:" with two radio buttons: "Polinom real" and "Polinom intreg". The "Polinom intreg" button is selected. To the right of these buttons are several operation buttons: "Adunare", "Scadere", "Inmultire", "Imparti...", "Derivare", "Integra...", and "Clear". The "Clear" button is highlighted with a blue border. On the left side, there are input fields for "Polinomul1:" (containing  $1x^2+4x^1+4x^0$ ), "Polinomul2:" (containing  $1x^1+2x^0$ ), "Rezultat:" (empty), and "Rest:" (empty).

Testarea derivarii unui polinom cu coeficienti intregi:

This screenshot shows the same window after the "Derivare" button has been clicked. The "Polinomul1:" field still contains  $1x^2+4x^1+4x^0$ , but the "Polinomul2:" field is now empty. The "Rezultat:" field now contains the derivative  $+2x^1+4x^0$ . The "Rest:" field remains empty. The "Derivare" button is no longer highlighted, and the "Clear" button remains at the bottom right.

Testarea integrării unui polinom cu coeficienti întregi:

Alegeti o singura varianta: ☐ Polinom real ☒ Polinom întreg

Polinomul1:

Polinomul2:

Rezultat:

Rest:

Adunare

Scadere

Inmultire

Imparti...

Derivare

Integra...

Clear

## 6. Concluzii

În concluzie, am reușit să implementez un calculator pentru calculul operațiilor pe polinoame cu coeficienți reali dar și cu coeficienți întregi. Îar pentru aplicația realizată am creat și o interfață foarte prietenoasă pentru a ajuta utilizatorul să introducă datele, să își aleagă operația dorită apăsând pe un buton ca mai apoi să poată să vadă rezultatele obținute în urma executării operațiilor, alegând tipul polinomului înainte de toate. Interfața este asadar foarte ușor de înțeles și de folosit.

## 7. Bibliografie

<https://www.w3schools.com/java/>

<https://www.oracle.com/ro/java/>