TEHNICI DE PROGRAMARE FUNDAMENTALE
ASSIGNMENT 1

CALCULATOR DE POLINOAME

DOCUMENTATIE

Kovacs Alexandru
Grupa 30223

Cuprins

1. Obiectivul temei

1.1. Obiectivul principal

Obiectivul principal al acestei teme de laborator este de a proiecta si a implementa o aplicatie cu functionalitatea unui calculator de polinoame ce include o interfata grafica dedicata prin intermediul careia utilizatorul aplicatiei poate introduce operanzii doriti sub forma de polinoame, selecta o operatie care se va efectua asupra operanzilor introdusi si de a vedea rezultatul operatiei selectate.

1.2. Obiective secundare

Obiectivele secundare care vor fi abordate in aceasta tema sunt urmatoarele:

- Analiza problemei si identificarea cerintelor – **Capitolul 2**
- Proiectarea calculatorului de polinoama – **Capitolul 3**
- Implementarea calculatorului de polinoame – **Capitolul 4**
- Testarea calculatorului de polinoame – **Capitolul 5**

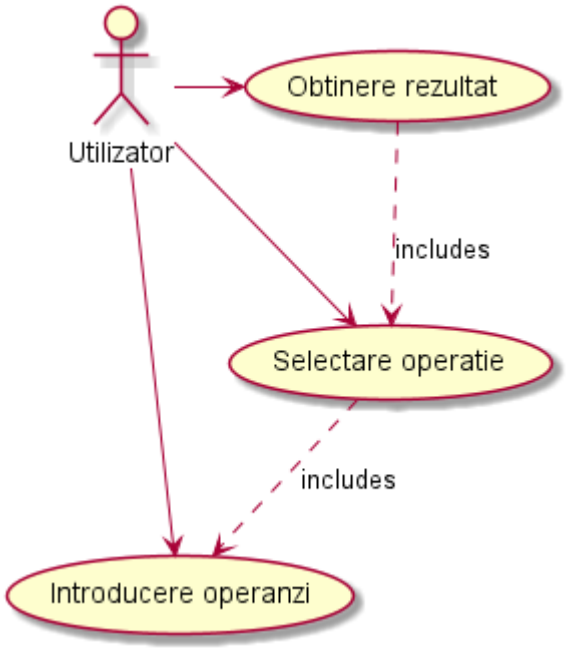
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

2.1. Analiza problemei

Cerintele functionale care reies din analiza enuntului temei de laborator sunt:

- Calculatorul de polinoame ar trebui sa permita utilizatorilor sa introduca polinoame
- Calculatorul de polinoame ar trebui sa permita alegerea unei operatii dintre cele specificate:
  - Adunarea polinoamelor
  - Scaderea polinoamelor
  - Inmultirea polinoamelor
  - Impartirea polinoamelor
  - Derivarea unui polinom
  - Integrarea unui polinom
- Calculatorul ar trebui sa afiseze rezultatul operatiei alese aplicata pe polinoamele introduse
- Calculatorul de polinoame ar trebui sa notifice utilizator in cazul aparitiei unei erori

2.2. Cazuri de utilizare



Caz de utilizare: adunarea polinoamelor
Actorul principal: utilizatorul
Scenariul principal:

1. Utilizatorul introduce primul polinom utilizand interfata grafica.
2. Utilizatorul selecteaza operatia „Add”
3. Utilizatorul introduce al doilea polinom utilizand interfata grafica.
4. Utilizatorul apasa butonul „=”.
5. Calculatorul de polinoame efectueaza adunarea pe cele doua polinoamele introduse si pune la dispozitie utilizatorului rezultatul operatiei.
6. Acum utilizatorul poate apasa butonul „Reset” pentru a efectua o alta operatie.

Secventa alternativa:

- In eventualitatea aparitiei unei erori interne, aceasta este afisata utilizatorului.
- Scenariul ajunge la pasul 6.

Scenariile pentru operatiile de scadere si inmultire decurg in aceeasi maniera ca si cazul prezentat anterior.

Caz de utilizare: impartirea polinoamelor
Actorul principal: utilizatorul
Scenariul principal:

1. Utilizatorul introduce primul polinom utilizand interfata grafica.
2. Utilizatorul selecteaza operatia „Divide”
3. Utilizatorul introduce al doilea polinom utilizand interfata grafica.
4. Utilizatorul apasa butonul „=”.
5. Calculatorul de polinoame efectueaza impartirea primului polinom la cel de-al doilea polinom si ofera utilizatorului un rezultat format dintr-un cat si un rest.
6. Acum utilizatorul poate apasa butonul „Reset” pentru a efectua o alta operatie.

Secventa alternativa:

- In eventualitatea aparitiei unei erori interne, aceasta este afisata utilizatorului.
- In cazul impartirii cu 0, utilizatorul este instiintat ca operatia dorita nu este permisa.
- Scenariul ajunge la pasul 6.

Caz de utilizare: derivarea unui polinom
Actorul principal: utilizatorul
Scenariul principal:

1. Utilizatorul introduce un polinom utilizand interfata grafica.
2. Utilizatorul selecteaza operatia „Derive”
3. Utilizatorul apasa butonul „=”.
4. Calculatorul de polinoame efectueaza derivarea polinomului introdus si afiseaza utilizatorului rezultatul operatiei.
5. Acum utilizatorul poate apasa butonul „Reset” pentru a efectua o alta operatie.

Secventa alternativa:

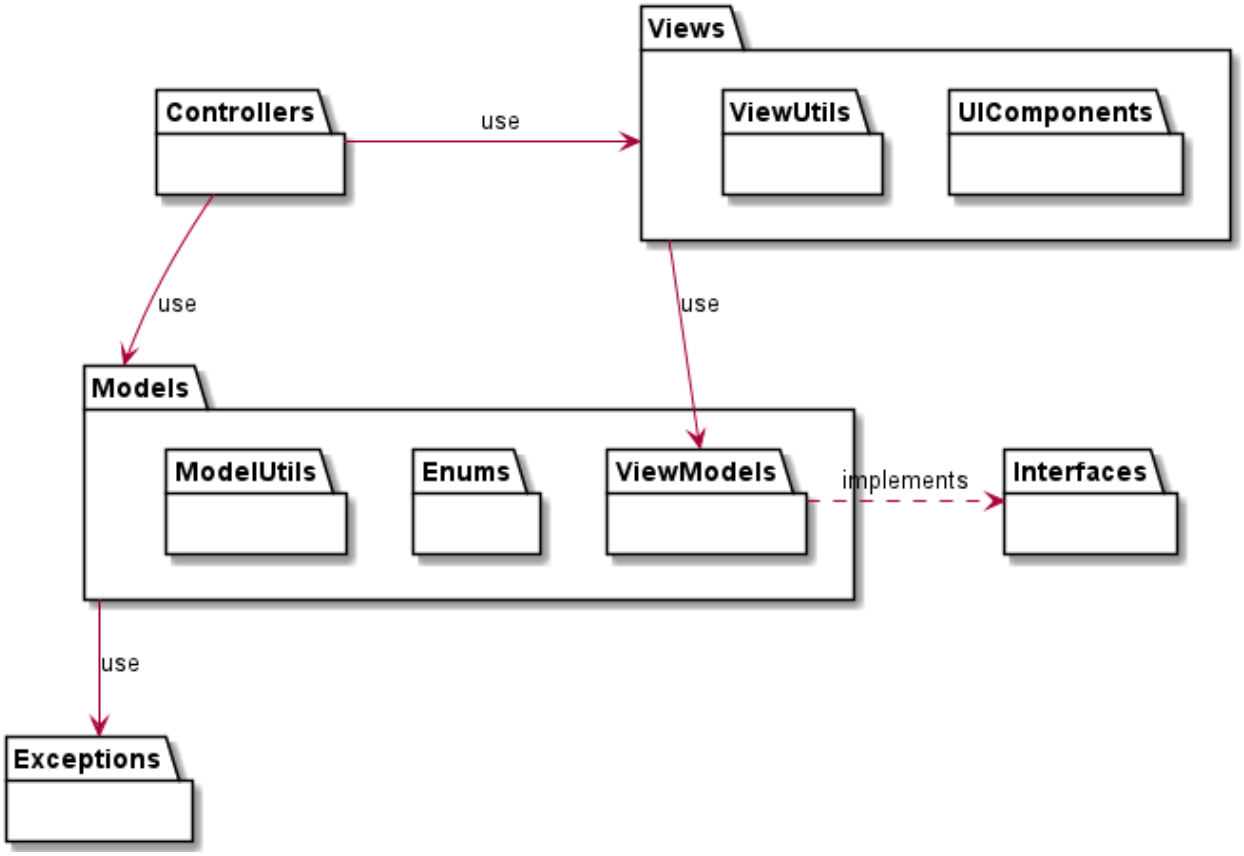
- In eventualitatea aparitiei unei erori interne, aceasta este afisata utilizatorului.
- Scenariul ajunge la pasul 5.

Scenariul pentru operatia de integrare este asemanator cu cel prezentat pentru operatia de derivare.

3. Proiectare

3.1. Diagrama de pachete

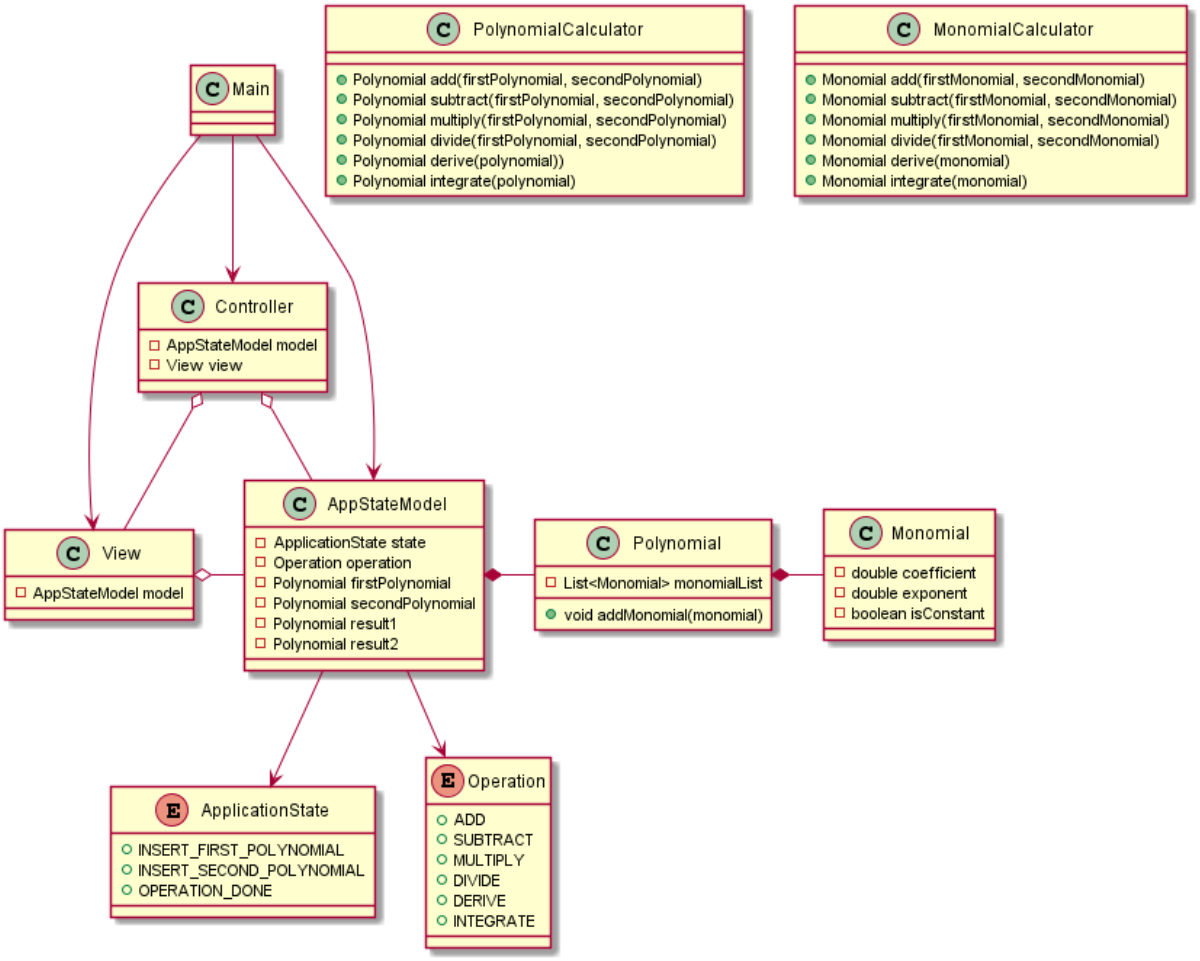
Am conceput acest proiect urmarind structura de pachete prezentata in figura de mai jos.
Se poate observa ca am utilizat un model arhitectural de tip MVC (Model View Controller).



Pachetele folosite sunt:

- **Controllers**
  - contine toate controllerele folosite in arhitectura MVC
- **Views**
  - contine toate view-urile folosite in arhitectura MVC
  - *UIComponents*
    - contine elemente de UI folosite in interiorul view-urilor
  - *Utils*
    - contine toate clasele care ajuta la crearea / initierea componentelor UI
- **Models**
  - contine toate clasele care memoreaza date si / sau stari folosite in arhitectura MVC
  - *Utils*
    - contine toate clasele care faciliteaza folosirea modelelor
  - *Enums*
    - contine toate enumerarile folosite de modelele
  - *ViewModels*
    - contine toate clasele folosite in view-uri care retin starea aplicatiei
- **Interfaces**
  - contine toate interfetele folosite in intregul proiect
- **Exceptions**
  - contine toate exceptiile folosite pe parcursul dezvoltarii proiectului

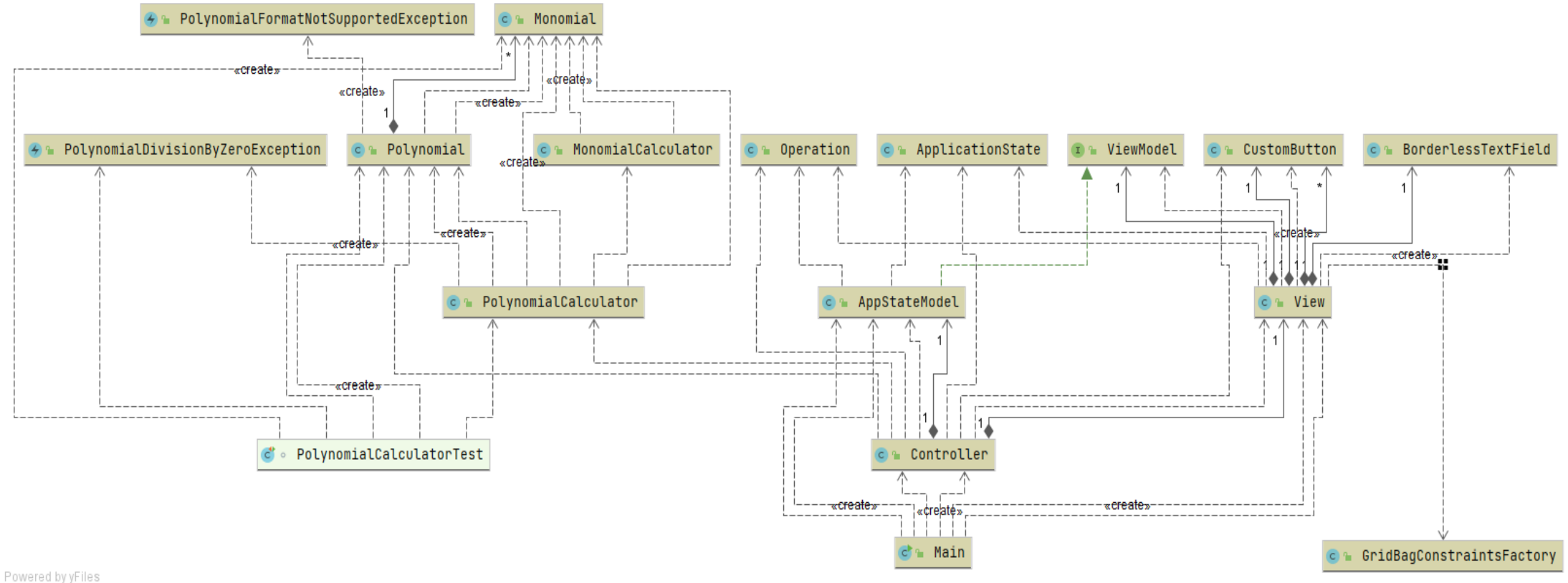
3.2. Diagrama de clase



Clasele identificate in timpul procesului de proiectare a acestei teme de laborator sunt:

- **Main**
  - este utilizata la crearea claselor principale (Model, View si Controller) folosite in arhitectura MVC
- **AppStateModel**
  - este modelul folosit de catre view pentru a retine starea curenta a aplicatiei
- **View**
  - este utilizata la construirea interfetei grafice dedicate
- **Controller**
  - este utilizata pentru a intercepta evenimentele cauzate de catre utilizator si pentru a modifica starea curenta a aplicatiei
- **Polynomial**
  - este utilizata pentru memorarea unei liste de monoame
- **Monomial**
  - este utilizata pentru a retine coeficientul si exponentul unui monom si / sau o constanta rezultata dupa operatia de integrare
- **PolynomialCalculator**
  - este utilizata pentru a efectua calcule cu polinoame
- **MonomialCalculator**
  - este utilizata pentru a efectua calcule cu monoame
- **ApplicationState**
  - este o enumeratie care include toate starile posibile in care se poate afla aplicatia
- **Operation**
  - este o enumeratie care specifica toate operatiile cu polinoame definite pentru aplicatia noastra

Aici avem diagrama de clase rezultata la finalul implementarii proiectului:



Singura interfata definita este interfata ViewModel. Prin aceasta interfata ne asiguram ca, clasa View primeste doar dreptul de citire a datelor retinute in modelul AppStateModel.

3.3. Algoritmi utilizati

3.3.1. Adunare

La algoritmul de adunare se aduna coeficientii monoamelor cu exponenti egali, iar monoamele rezultate impreuna cu restul monoamelor ramase se adauga la lista monoamelor polinomului rezultat.

3.3.2. Scadere

La algoritmul de scadere se scad coeficientii monoamelor cu exponenti egali, iar monoamele rezultate impreuna cu restul monoamelor ramase se adauga la lista monoamelor polinomului rezultat.

3.3.3. Inmultire

La algoritmul de inmultire se inmulteste fiecare monom din al doilea polinom cu primul polinom, iar rezultatele obtinute se aduna la polinomului rezultat.

3.3.4. Impartire

Algoritmul de impartire folosit urmeaza urmatorii pasi:

1. Se ordoneaza in maniera descrescatoare monoamele celor doua polinoame in functie de exponentul acestora.
2. Se impart monoamele cele mai semnificative (cu exponentul cel mai mare)
3. Monomul obtinut se adauga la polinomul cat.
4. Monomul obtinut se inmulteste cu polinomul impartitor.
5. Polinomul rezultat se scade din polinomul deimpartit.
6. Se repeta pasii de la 2 pana la 5 cat timp polinomul deimpartit este diferit de 0 sau gradul acestuia este mai mic sau egal decat gradul polinomului impartitor.
7. Rezultatul impartirii se regaseste in polinomul cat.
8. Restul impartirii este polinomul deimpartit rezultat dupa scaderile repetate.

3.3.5. Derivare

La algoritmul de derivare se parcurg toate monoamele polinomului si se modifica coeficientul si exponentul astfel: coeficient <- coeficient * exponent; exponent <- exponent - 1.

3.3.6. Integrare

La algoritmul de derivare se parcurg toate monoamele polinomului si se modifica coeficientul si exponentul astfel: coeficient <- coeficient / (exponent + 1); exponent <- exponent + 1.

4. Implementare

4.1. Descriere clase

Clasa Controller

- Este componenta care gestioneaza interactiunea utilizatorului cu aplicatia si lucreaza cu modelul.
- Este C-ul din modelul arhitectural MVC.

private final AppStateModel model;
- Obiect ce memoreaza starea curenta a aplicatiei (modelul folosit pentru interfata grafica)

private final View view;
- Obiect ce contine interfata grafica (view-ul)

private void setUpUserEvents()
- Metoda ce atribuie componentelor din interfata grafica ascultatoare de evenimente (ActionListener)

private void setTextBtnActionListeners()
- Metoda ce atribuie butoanelor folosite pentru a introduce polinoamele un ActionListener

private void setActionBtnActionListeners()
- Metoda ce atribuie butoanelor folosite pentru a selecta operatia dorita un ActionListener

private void setEqualsBtnActionListener()
- Metoda ce atribuie butonului "=" un ActionListener

private void updateOperationResult(Polynomial firstPolynomial, Polynomial secondPolynomial) throws PolynomialDivisionByZeroException
- Metoda care se ocupa cu modificarea datelor retinute de modelul de date

Clasa PolynomialDivisionByZeroException

- Este o exceptie folosita pentru a semnala incercarea impartirii unui polinom la 0.

Clasa PolynomialFormatNotSupportedException

- Este o exceptie folosita pentru a semnala introducerea unui format de polinom care nu exista suport.

Interfata ViewModel

- Este o interfata folosita pentru a ne asigura ca view-ul are doar drept de citire asupra modelului.

Enumeratia ApplicationState

- Este o enumeratie care contine toate starile posibile in care se poate afla aplicatia.

Enumeratia Operation

- Este o enumeratie care contine toate operatiile pe polinoame oferite de aplicatie.

Clasa MonomialCalculator

- Este o clasa statica de care ne folosim pentru a efectua operatii cu monoame.

public static Monomial multiply(Monomial firstMonomial, Monomial secondMonomial)

- Metoda ce multiplica doua monoame.

public static Monomial divide(Monomial firstMonomial, Monomial secondMonomial)

- Metoda ce imparte doua monoame.

public static Monomial derive(Monomial monomial)

- Metoda ce deriveaza un monom.

public static Monomial integrate(Monomial monomial)

- Metoda ce integreaza doua monoame.

Clasa PolynomialCalculator

- Este o clasa statica de care ne folosim pentru a efectua operatii cu polinoame.

public static Polynomial add(Polynomial firstPolynomial, Polynomial secondPolynomial)

- Metoda ce aduna doua polinoame.

public static Polynomial subtract(Polynomial firstPolynomial, Polynomial secondPolynomial)

- Metoda ce scade doua polinoame.

public static Polynomial multiply(Polynomial polynomial, double value)

- Metoda ce inmulteste un polinom cu un scalar.

public static Polynomial multiply(Polynomial polynomial, Monomial monomial)

- Metoda ce inmulteste un polinom cu un monom.

public static Polynomial multiply(Polynomial firstPolynomial, Polynomial secondPolynomial)

- Metoda ce inmulteste doua polinoame.

private static Polynomial divide(Polynomial firstPolynomial, Polynomial secondPolynomial, boolean remainder) throws PolynomialDivisionByZeroException

- Metoda ce imparte doua polinoame si permite alegerea catului sau restului.

public static Polynomial getDivisionResult(Polynomial firstPolynomial, Polynomial secondPolynomial) throws PolynomialDivisionByZeroException

- Metoda ce returneaza catul impartirii a doua polinoame.

public static Polynomial getDivisionRemainder(Polynomial firstPolynomial, Polynomial secondPolynomial) throws PolynomialDivisionByZeroException

- Metoda ce returneaza restul impartirii a doua polinoame.

public static Polynomial derive(Polynomial polynomial)

- Metoda ce deriveaza un polinom.

public static Polynomial integrate(Polynomial polynomial)

- Metoda ce integreaza un polinom.

Clasa AppStateModel

- Este componenta care retine starea aplicatiei.
- Este M-ul din modelul arhitectural MVC.

private final PropertyChangeSupport support;
- Obiect ce faciliteaza informarea view-ului de schimbari de date in vederea afisarii acestora.

private int state;
- Variabila ce retine starea aplicatiei.

private int operation;
- Variabila ce memoreaza operatia selectata.

private String firstPolynomialText;
- Obiect ce memoreaza textul care urmeaza sa fie afisat in primul Text Field.

private String operationText;
- Obiect ce memoreaza textul care urmeaza sa fie afisat in al doilea Text Field.

private String secondPolynomialText;
- Obiect ce memoreaza textul care urmeaza sa fie afisat in al treilea Text Field.

public void reset()
- Metoda ce reseteaza starea aplicatiei si textul care urmeaza sa fie afisat utilizatorului.

public String getTarget()
- Metoda care intoarce textul care urmeaza sa fie modificat din interfata vizuala.

public void addPropertyChangeListener(PropertyChangeListener pcl)
- Metoda care memoreaza obiecte care urmeaza sa fie notificate in momentul schimbarii valorilor retinute.

public void sendUpdate()
- Metoda care trimite notificarea cu privire la schimbarea valorilor retinute de model.

Clasa Monomial

- Este un model care retine informatii depsre un monom.

private double coefficient;
- Variabila ce retine coeficientul unui monom.

private double exponent;
- Variabila ce retine exponentul unui monom.

private boolean isConstant;
- Variabila ce indica faptul daca monomul este o constanta sau nu (rezultat in urma integrarii).

public String toString()
- Metoda folosita la transpunerea monomului in format text.

Clasa Polynomial

- Este un model care retine informatii depsre un monom.

private final List<Monomial> monomialList;

- O lista care contine toate monoamele din componenta polinomului.

public List<Monomial> getMonomialList()

- Metoda care face vizibila lista de monoame.

public Polynomial addMonomial(Monomial monomial)

- Metoda care cauta in lista de monoame a polinomului un monom cu exact acelasi exponent ca si al monomului de adaugat. Daca se gaseste un astfel de monom, se aduna coeficientul monomului de adaugat la monomul deja existent. Altfel, se adauga monomul in lista de monoame.

public Polynomial orderByExponent()

- Metoda care ordoneaza polinomul in ordine descrescatoare in functie de exponentii monoamelor.

public Polynomial prune()

- Metoda care sterge toate monoamele egale cu 0 din lista de monoame.

public double getDegree()

- Metoda care returneaza gradul polinomului.

public Monomial getMonomial(int index)

- Metoda care permite alegerea unui polinom situat la indexul specificat.

public boolean isEmpty()

- Metoda care verifica daca polinomul nu are niciun monom.

private static String prepString(String str)

- Metoda care aplica o serie de modificari pe sirul de caractere primit ca si polinom.

public static Polynomial fromString(String str) throws PolynomialFormatNotSupportedException

- Metoda care transforma sirul de caractere primit intr-un Obiect de tip Polynomial folosind regex.
- In cazul in care formatul este unul invalid se va arunca o exceptie.

public String toString()

- Metoda folosita la transpunerea polinomului in format text.

Clasa BorderlessTextField & Clasa CustomButton

- Este o componenta ui cu niste setari predefinite.

private void applySettings()

- Metoda care aplica componentei setarile dorite.

private void setResizable()

- Metoda care initializeaza scriptul care se ocupa de modificarea marimii textului din componenta in functie de marimea ferestrei.

private void adapt()

- Metoda care face calculul pentru noua marime a fontului in functie de noua dimensiune a ferestrei.

Clasa GridBagConstraints

- Este o clasa statica care faciliteaza obtinerea usoara a unor GridBagConstraints cu diferite optiuni aplicate.

Clasa View

- Este componenta care structureaza interfata grafica dedicata a aplicatiei.
- Este V-ul din modelul arhitectural MVC.

private ViewModel model;

- O referinta catre o interfata de tip ViewModel pentru a asigura posibilitatea citirii datelor retinute in model.

protected void validateButtons()

- Metoda care decide care butoane se activeaza si care se dezactiveaza in functie de starea curenta a aplicatiei.

public void propertyChange(PropertyChangeEvent evt)

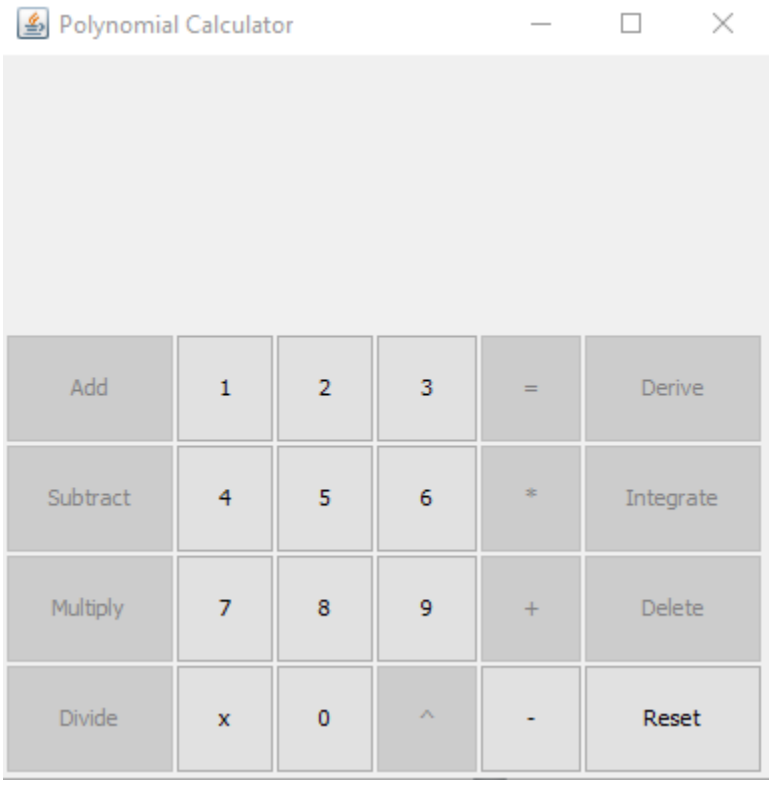
- Aceasta metoda este apelata cand este declansat un eveniment de schimbare a datelor din model.

Clasa Main

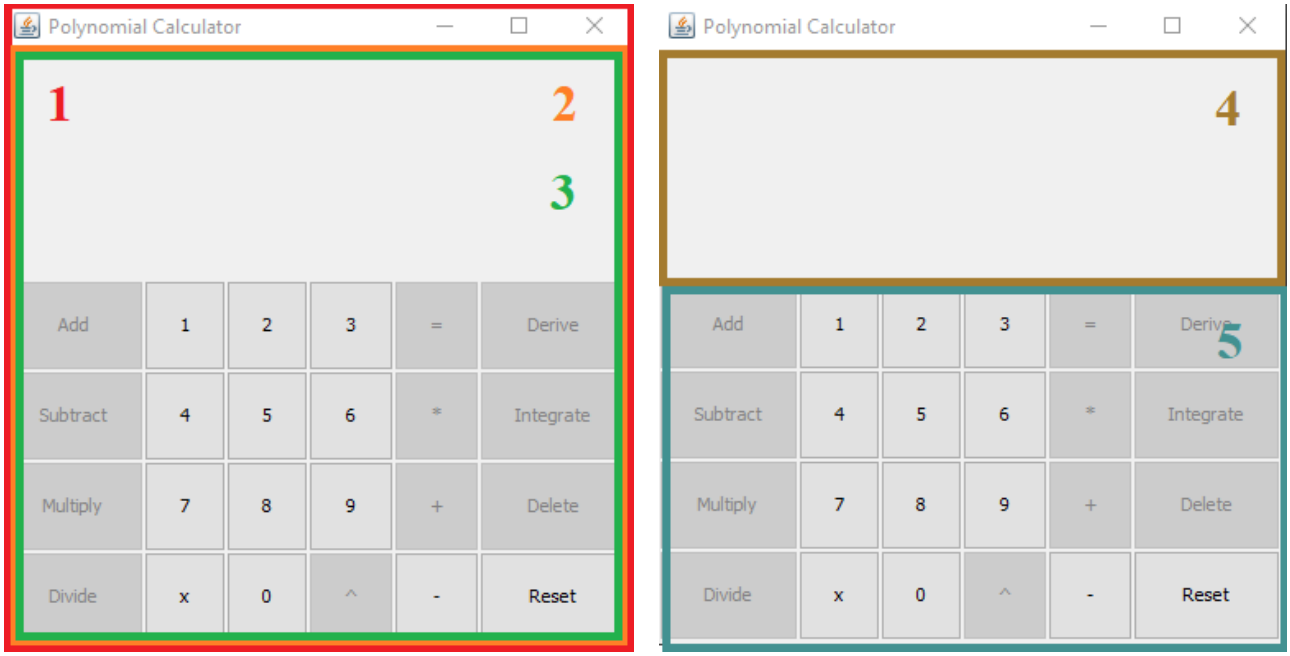
- Este o clasa statica in care se face initierea componentelor necesare pentru a completa modelul arhitectural MVC.
- Este punctul de start al aplicatiei.

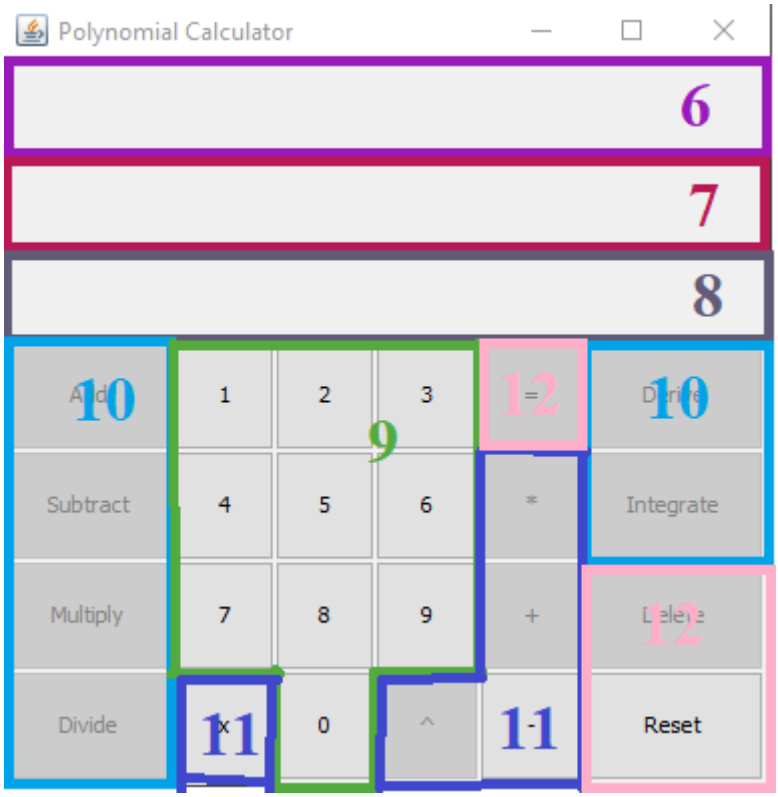
4.2. Descriere implementare interfata utilizator

Interfata grafica este una user-friendly. Aceasta este usor de folosit, intuitiva si nu permite introducerea de date incorecte. De asemenea, dimensiunea butoanelor si a scrisului se modifica in functie de dimensiunea ferestrei.



Aceasta interfata grafica este alcatuita dintr-un JFrame (1) care contine un singur content pane (2), caruia i-am atribuit un layout de tip GridLayout cu o singura linie si o singura coloana. Scopul acestui content (2) pane este de a redimensiona componentele interne lui o data cu schimbarea dimensiunii ferestrei. Acest lucru se datoreaza GridLayout-ului. In acest content (2) pane am introdus un panel principal (3) al aplicatiei care, la randul lui, contine alte doua panel-uri secundare: un panel pentru componentele de tip Text Field (4) si un panel pentru butoane (5). Acestui panel principal (3) i-am adaugat un layout de tip GridBagLayout pentru a putea obtine o impartire procentuala a inaltimii pentru componentele interne cuprinse de acesta.





Panel-ul pentru componente de tip Text Field (4) si pentru butoane (5) s-a folosit un layout de tip GridBagLayout.

Panel-ul pentru componentele de tip Text Field (4) contine trei astfel de componente care au fost plasate in interiorul panel-ului in pozitiile dorite cu ajutorul unor GridBagConstraints. Acest panel (4) are inaltimea egala cu 30% din inaltimea ferestrei (1). Primul Text Field (6) este folosit pentru introducerea primului polinom si pentru afisarea rezultatului final. Al doilea Text Field (7) are fontul ingrosat si este folosit pentru afisarea operatiei selectate, restului impartirii si posibilele erori care pot aparea. Ultimul Text Field (8) este folosit pentru a introduce al doilea polinom.

Aceste Text Field-uri (6, 7, 8) sunt de tip BorderlessTextField si au bordura dezactivata si atasat un script care redimensioneaza marimea fontului odata cu schimbarea dimensiunii ferestrei.

Panel-ul pentru butoane (5) are in componenta sa douazeci si patru de butoane:

- **Butoane pentru cifre (10 butoane) (9)**
  - sunt folosite pentru a introduce numere in timpul scrierii polinoamelor

- **Butoane pentru simboluri speciale (5 butoane) (11)**
  - *Butonul „x”*
    - folosit pentru a insera variabila x in polinom
  - *Butonul „^”*
    - folosit pentru a scrie simbolul putere
    - acesta poate fi folosit doar dupa apasarea butonului „x”
  - *Butonul „-”*
    - folosit pentru a adauga termeni negativi la polinom
  - *Butonul „+”*
    - folosit pentru a adauga termeni pozitivi la polinom
  - *Butonul „**”*
    - folosit pentru a reprezenta explicit inmultirea unui coeficient cu x
    - utilizarea acestui buton este optionala

- **Butoane pentru operatii (6 butoane) (10)**
  - *Butonul „Add”*
    - seteaza operatia dorita care urmeaza sa se efectueze
    - se poate apasa doar dupa introducerea primului polinom
    - necesita introducerea unui al doilea polinom dupa apasare
  - *Butonul „Subtract”*
    - seteaza operatia dorita care urmeaza sa se efectueze
    - se poate apasa doar dupa introducerea primului polinom
    - necesita introducerea unui al doilea polinom dupa apasare
  - *Butonul „Multiply”*
    - seteaza operatia dorita care urmeaza sa se efectueze
    - se poate apasa doar dupa introducerea primului polinom
    - necesita introducerea unui al doilea polinom dupa apasare
  - *Butonul „Divide”*
    - seteaza operatia dorita care urmeaza sa se efectueze
    - se poate apasa doar dupa introducerea primului polinom
    - necesita introducerea unui al doilea polinom dupa apasare
  - *Butonul „Derive”*
    - seteaza operatia dorita care urmeaza sa se efectueze
    - se poate apasa doar dupa introducerea primului polinom
    - **NU** se mai introduce un al doilea polinom dupa apasare
  - *Butonul „Integrate”*
    - seteaza operatia dorita care urmeaza sa se efectueze
    - se poate apasa doar dupa introducerea primului polinom
    - **NU** se mai introduce un al doilea polinom dupa apasare

- **Butoane speciale (3 butoane) (12)**
  - *Butonul „Delete”*
    - folosit pentru a sterge ultimul caracter introdus
    - **NU** poate fi folosit pentru a sterge operatia selectata
    - **NU** poate fi folosit pentru a reveni la modificarea primului polinom
  - *Butonul „Reset”*
    - folosit pentru a reseta starea aplicatiei
    - nu mai este retinuta operatia selectata
    - sterge tot textul din componentele BorderlessTextField
  - *Butonul „=”*
    - folosit pentru afisarea rezultatului operatiei selectate
    - poate fi folosit doar dupa introducerea numarului necesar de operanzi
    - schimba starea aplicatiei

Butoanele (9, 10, 11, 12) folosite sunt de tip CustomButton si au capacitatea de a intra in focus dezactivata si atasat un script care modifica marimea fontului dupa schimbarea dimensiunii ferestrei. Aceste butoane (9, 10, 11, 12) sunt plasate in interiorul panel-ului (5) cu ajutorul unor GridBagConstraints care le determina marimea si pozitia. Panel-ul care contine butoanele (5) are inaltimea egala cu 70% din inaltimea ferestrei (1).

5. Rezultate

Pentru a verifica rezultatele operatiilor din cadrul aplicatiei am folosit o clasa de testare unitara numita PolynomialCalculatorTest.

Metodele prezente in clasa de testare sunt:

public static void init()
- se executa o singura data inaintea inceperii executiei testelor
- initializeaza numarul de teste efectuat si numarul de teste trecute cu success

public void next()
- se executa dupa fiecare test
- creste numarul de teste efectuat

public static void done()
- se executa o singura data dupa terminarea executiei testelor
- afiseaza numarul de teste efectuat si numarul de teste trecute cu success

void add() throws PolynomialFormatNotSupportedException
- testeaza operatia de adunare

Scenariul 1

First polynomial: x^3 - 5x^2 + 7x +3
Second polynomial: x^2 - 3x + 1
Expected result: x^3 - 4x^2 + 4x + 4
Trecut: Da
Observatii: -

Scenariul 2

First polynomial: 3x^2 + 2x + 1 + 4x^3
Second polynomial: -x^3 +5x + 1 + 4x^4
Expected result: 4x^4 + 3x^3 + 3x^2 + 7x + 2
Trecut: Da
Observatii: Monoamele nu sunt ordonate dupa exponent.

void subtract() throws PolynomialFormatNotSupportedException
- testeaza operatia de scadere

Scenariul 1

First polynomial: x^2 - 1
Second polynomial: x^3 + 2x^2 - 1
Expected result: -x^3 - x^2
Trecut: Da
Observatii: -

Scenariul 2

First polynomial: -7x^3 + 3x - 4 + 11x^4
Second polynomial: 5x^2 + 2x + 1 + 4x^3
Expected result: 11x^4 - 11x^3 - 5x^2 + x - 5
Trecut: Da
Observatii: Monoamele nu sunt ordonate dupa exponent.

void multiply() throws PolynomialFormatNotSupportedException

- testeaza operatia de inmultire

Scenariul 1

First polynomial: x^4 - 7x^2 + 3
Second polynomial: x^3 - 2x
Expected result: x^7 - 9x^5 + 17x^3 - 6x
Trecut: Da
Observatii: -

Scenariul 2

First polynomial: 3x^2 - x^7 + 5
Second polynomial: 4 + 2x^2
Expected result: -2x^9 - 4x^7 + 6x^4 + 22x^2 + 20
Trecut: Da
Observatii: Monoamele nu sunt ordonate dupa exponent.

void getDivisionResult() throws PolynomialFormatNotSupportedException, PolynomialDivisionByZeroException

- testeaza operatia de impartire

Scenariul 1

First polynomial: x^2 + 1
Second polynomial: x + 1
Expected result: x - 1
Trecut: Da
Observatii: -

Scenariul 2

First polynomial: x^2 + 5
Second polynomial: 0
Expected result: Division by 0 exception
Trecut: Da
Observatii: Division by 0.

Scenariul 3

First polynomial: x^2 + 7x - 11
Second polynomial: x^7 + 5x^2 + 3x + 19
Expected result: 0
Trecut: Da
Observatii: Impartitor mai mare decat deimpartit.

void getDivisionRemainder() throws PolynomialFormatNotSupportedException, PolynomialDivisionByZeroException

- testeaza operatia de impartire

Scenariul 1

First polynomial: x^2 + 1
Second polynomial: x + 1
Expected result: 2
Trecut: Da
Observatii: -

Scenariul 2

First polynomial: x
Second polynomial: 0
Expected result: Division by 0 exception
Trecut: Da
Observatii: Division by 0.

Scenariul 3

First polynomial: x^2 + 7x - 11
Second polynomial: x^7 + 5x^2 + 3x + 19
Expected result: x^2 + 7x - 11
Trecut: Da
Observatii: Impartitor mai mare decat deimpartit.

void derive() throws PolynomialFormatNotSupportedException

- testeaza operatia de derivare

Scenariul 1

First polynomial: x^5 - 12x^4 + 3x^3 - 7x^2 + 420
Second polynomial: -
Expected result: 5x^4 - 48x^3 + 9x^2 - 14x
Trecut: Da
Observatii: -

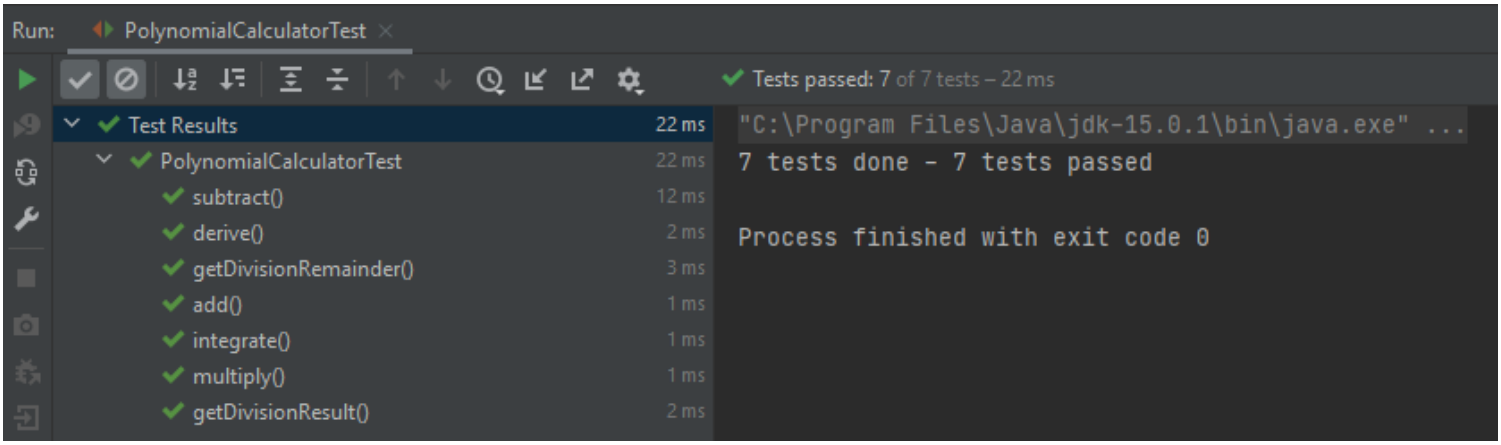
void integrate() throws PolynomialFormatNotSupportedException

- testeaza operatia de integrare

Scenariul 1

First polynomial: 4x^3 - x^2 + 4x - 11
Second polynomial: -
Expected result: x^4 – 0.33x^3 + 2x^2 – 11x + C
Trecut: Da
Observatii: -

Rezultatul executiei testelor:



6. Concluzii

Am realizat o aplicatie Java care efectueaza operatii cu polinoame, acestea fiind introduse dintr-o interfata grafica dedicata de catre utilizator.

Acest proiect este unul pentru uz general, se poate folosi oricand e nevoie de efectuarea unor operatii cu polinoame. Functioneaza fara probleme, au fost tratate diferite cazuri, astfel incat sa se obtina rezultatul dorit sau ca sa fie afisat un mesaj de eroare in cazul aparitiei unei erori.

In cadrul acestei teme am invatat sa folosesc regular expressions, sa stapanesc mai bine folosirea modelului arhitectural Model-View-Controller si sa acumulez mai multa experienta in utilizarea limbajului Java pentru a rezolva diverse probleme.

Ca dezvoltari ulterioare se pot aduce optimizari operatiilor deja existente (algorimului de impartire) sau se pot implementa noi functionalitati precum un istoric al operatiilor efectuate, posibilitatea de a folosi mai multe variabile, includerea a diverse functii matematice (sin, cos, ln, etc.) sau chiar posibilitatea de a importa / exporta rezultatele unor calcule.

7. Bibliografie

- ✓ Regex
  - https://regexr.com/
  - https://regex101.com/
  - https://www.baeldung.com/regular-expressions-java
  - https://www.youtube.com/watch?v=rhzKDrUiJVk
- ✓ Swing
  - https://docs.oracle.com/javase/8/docs/api/java/awt
  - https://stackoverflow.com/
  - https://docs.oracle.com/javase/tutorial/uiswing/
- ✓ JUnit Testing
  - https://www.baeldung.com/junit-5
- ✓ Diagrame
  - https://plantuml.com/sitemap-language-specification
- ✓ Modelul arhitectural MVC
  - Programare Orientata Obiect (An 2, Semestrul 1)
  - https://medium.com/@socraticsol/why-mvc-architecture-e833e28e0c76
- ✓ Java naming conventions
  - https://google.github.io/styleguide/javaguide.html