Documentatie

Tema 1

Nume Student : Constantin Alexandru-Ciprian

Grupa : 30227

CUPRINS

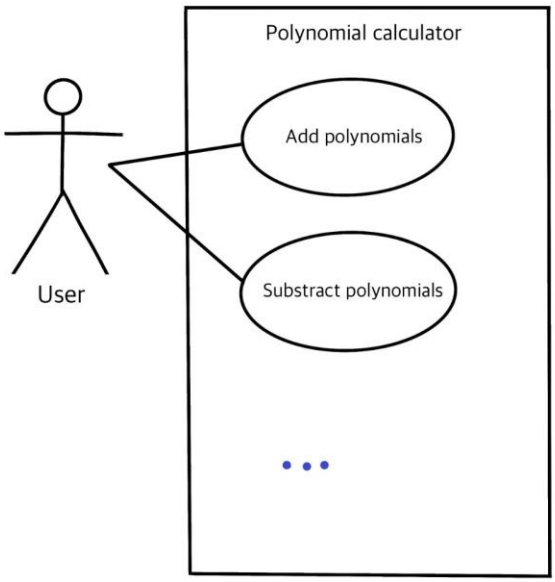
1. Obiectivul temei

- **Obiectivul principal**

- Implementarea unui model de calculator cu o interfata grafica prin care utilizatorul introduce polinoame, iar prin selectarea unei operatii sa se realizeze afisarea raspunsului;

- **Sub-obiective**
  - **Analiza problemei si identificarea cerintelor;**
  - **Modelarea calculatorului de polinoame;**
  - **Implementarea calculatorului de polinoame;**
  - **Testarea calculatorului de polinoame;**

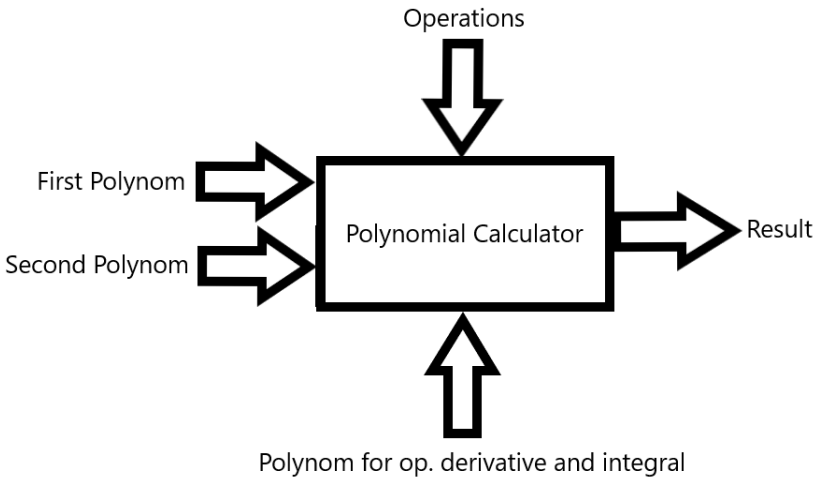
2. Analiza problemei, modelare, scenarii, cazuri de utilizare



- Cerinte functionale :
  - Calculatorul permite utilizatorului sa introduca polinoame intr-un textField;
  - Calculatorul permite utilizatorului sa selecteze operatia matematica;

- Calculatorul incorporeaza operatiile de adunare, scadere, inmultire si de impartire a doua polinoame, cat si operatiile de derivare si integrare a unui polinom;

- Cerinte non-functionale :
  - Calculatorul are zone special denumite pentru introducerea polinoamelor in functie de operatiile pe care utilizatorul vrea sa le faca;
  - Afisarea raspunsului are de altfel zone speciale, respectiv pentru operatiile pe doua polinoame, distinctive pentru operatia de impartire, cat si pentru operatiile de derivare si integrare;



- Use case : add, substract, multiplication, division
- Primary Actor : user-ul
- Main Succes Scenario:
  1. Utilizatorul introduce 2 polinoame in interfata grafica
  2. Utilizatorul selecteaza butonul cu unul din operatiile specificate
  3. Calculatorul realizeaza operatia comandata verificand eventualele erori de introducere;

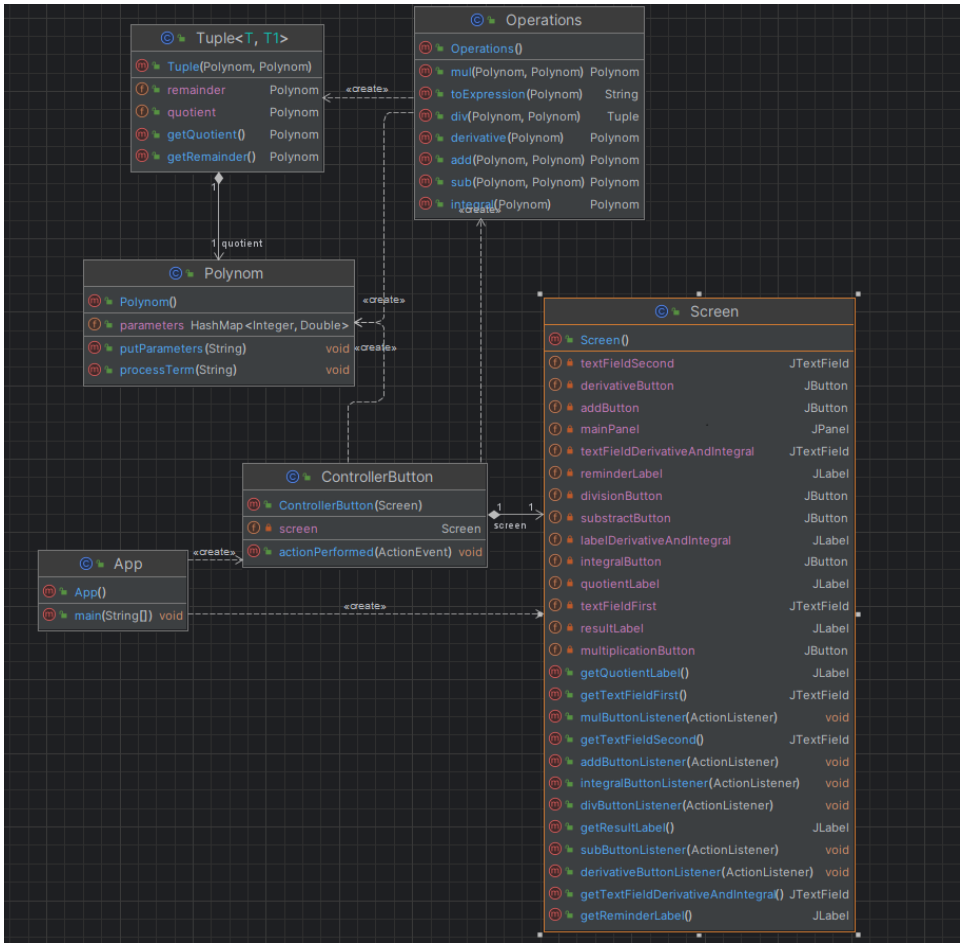
3. Proiectare

- Mai jos este prezentata diagrama UML, care contine 3 pachete (application, calculator, polynomial);

- application package contine clasa App;

- calculator package contine clasele ControllerButton si Screen;

- polynomial package contine clasele Operations, Polynom si Tuple;


Tuple<T, T1>
Tuple(Polynom, Polynom)
remainder Polynom
quotient Polynom
getQuotient() Polynom
getRemainder() Polynom
Operations
Operations()
mul(Polynom, Polynom) Polynom
toExpression(Polynom) String
div(Polynom, Polynom) Tuple
derivative(Polynom) Polynom
add(Polynom, Polynom) Polynom
sub(Polynom, Polynom) Polynom
integral(Polynom) Polynom
«create»
1 quotient
Polynom
Polynom()
parameters HashMap<Integer, Double>
putParameters(String) void
processTerm(String) void
Screen
Screen()
textFieldSecond JTextField
derivativeButton JButton
addButton JButton
mainPanel JPanel
textFieldDerivativeAndIntegral JTextField
reminderLabel JLabel
divisionButton JButton
substractButton JButton
labelDerivativeAndIntegral JLabel
integralButton JButton
quotientLabel JLabel
textFieldFirst JTextField
resultLabel JLabel
multiplicationButton JButton
getQuotientLabel() JLabel
getTextFieldFirst() JTextField
mulButtonListener(ActionListener) void
getTextFieldSecond() JTextField
addButtonListener(ActionListener) void
integralButtonListener(ActionListener) void
divButtonListener(ActionListener) void
getResultLabel() JLabel
subButtonListener(ActionListener) void
derivativeButtonListener(ActionListener) void
getTextFieldDerivativeAndIntegral() JTextField
getReminderLabel() JLabel
ControllerButton
ControllerButton(Screen)
screen Screen
actionPerformed(ActionEvent) void
1 screen
App
App()
main(String[]) void

4.Implementare

- Din clasa App se face executarea interfetei grafice impreuna cu functionalitatiile sale;
- In clasa Screen sunt instantiate componenetele interfetei grafice, impreuna cu getter-ele pentru label-uri si ActionListener-ele pentru butoane, implementarea functionalitatii fiecaruia fiind realizata in clasa ControllerButton;
- In clasa ControllerButton se realizeaza functionalitatiile butoaneleor si controlul acestora, astfel utilizatorul prin apasarea butonului va conduce la returnarea comenzii si prelucrarea acesteia, folosind operatiile implememntate in clasa Operations;
- In clasa Operations se regasesc implementate operatiile de adunare, scadere, inmultire, impartire, derivare si integrare a polinoamelor, dealtfel folosind o clasa ce incapsuleaza parametrii importantii a unui polinom necesari realizarii acestor actiuni;
- Totodata in clasa Operations exista metoda toExpression care realizeza trecerea in parametrii utilizabili intr-un String , facand mai usoara comunicarea cu utilizatorului;

- In clasa Polynom se instantiaza un atribut de tip HashMap care contine cheiile power -> de tip Integer si valorile coef -> de tip Double;
- Metoda putParameters(String expression) face trecerea din String a raspunsului introdus de utilizator in parametrii utilizabili in procesarea operatiilor;
- Clasa Tuple are rolul de a putea returna o tupla in cadrul operatiei de impartire, respectiv cat si rest, a celor doua polinoame introduse de utilizator;

5.Rezultate

- In proiectul prezentat am folosit testarea JUnit pentru toate operatii existente, respectiv in Clasa OperationsTest.
- Testarea fiecarei operatii se face prin introducerea de polinoame sub forma de String si compararea raspunsului asteptat cu rezultatul returnat de metoda specializata operatiei testate;
- Pentru operatiile de adunare, scadere, inmultire, impartire s-a testat pe doua polinoame in format String, iar pentru operatii de derivare si integrare s-a testat pe un polinom in format String;
- Un exemplu de implementare a unui test in cazul operatiilor ce utilizeaza doua polinoame este exemplificat mai jos in figura;

```
@Test
public void addTest(){
    Polynom polynom1 = new Polynom();
    Polynom polynom2 = new Polynom();
    Polynom resultPolynom = new Polynom();
    polynom1.putParameters( expression: "4x^5-3x^4+x^2-8x^1+1");
    polynom2.putParameters( expression: "3x^4-x^3+x^2+2x^1-1");
    resultPolynom.putParameters( expression: "4x^5-x^3+2x^2-6x^1");
    assertEquals(Operations.add(polynom1, polynom2).parameters, resultPolynom.parameters);
}
```

- Pentru testarea operatiei de integrare si derivare avem exemplificat in imaginea de mai jos;

```
@Test
public void derivativeTest(){
    Polynom polynom = new Polynom();
    Polynom resultPolynom = new Polynom();
    polynom.putParameters( expression: "x^3-2x^2+6x^1-5");
    resultPolynom.putParameters( expression: "3x^2-4x^1+6");
    assertEquals(Operations.derivative(polynom).parameters, resultPolynom.parameters);
}
```

- Pentru partea de testare finala, in imaginea de mai jos este prezentat rezultatul testelor;

6.Concluzii

- **Proiectul a constat in crearea unei aplicatii in java, respectiv intr-un Calculator de Polinoame, in care s-au implementat diferite operatii, facandu-se comunicarea cu utilizatorul printr-o interfata grafica folosind JavaSwing;**
- **Pentru logica operatiilor s-a utilizat structura de date HashMap care stocheaza datele in functie de cheie si valoare, in proiectul prezent cheia reprezentand puterea monomului extras din polinom si valoarea fiind coeficientul monomului;**
- **Proiectul a fost structurat in pachete si clase specializate pe cerintele problemei;**