TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA
ROMANIA

ASSIGNMENT 1 Polynomial Calculator

Programming Techniques

Student: Livia Mitrica
Group: 30424
Teacher Assistant: Viorica Chifu

Contents

Objective

The aim of this assignment is to design and implement a polynomial calculator with a dedicated graphical interface through which the user can enter polynomials, select the operation to be performed (i.e. addition, subtraction, multiplication, division, derivative, integration) and display the result. The polynomials considered will be of one variable and will have integer coefficients.

Problem analysis, modelling, scenarios, use cases

In mathematics, a polynomial is an expression consisting of variables (also called indeterminates) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents of variables.

Generally, a polynomial is written as

$$\sum_{i=0}^{n}(a_i x^i) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Therefore, each polynomial consists of a number of monomials. This way of representation is very helpful when performing the most common operations on polynomials: addition, subtraction, multiplication, division, differentiation, and integration.

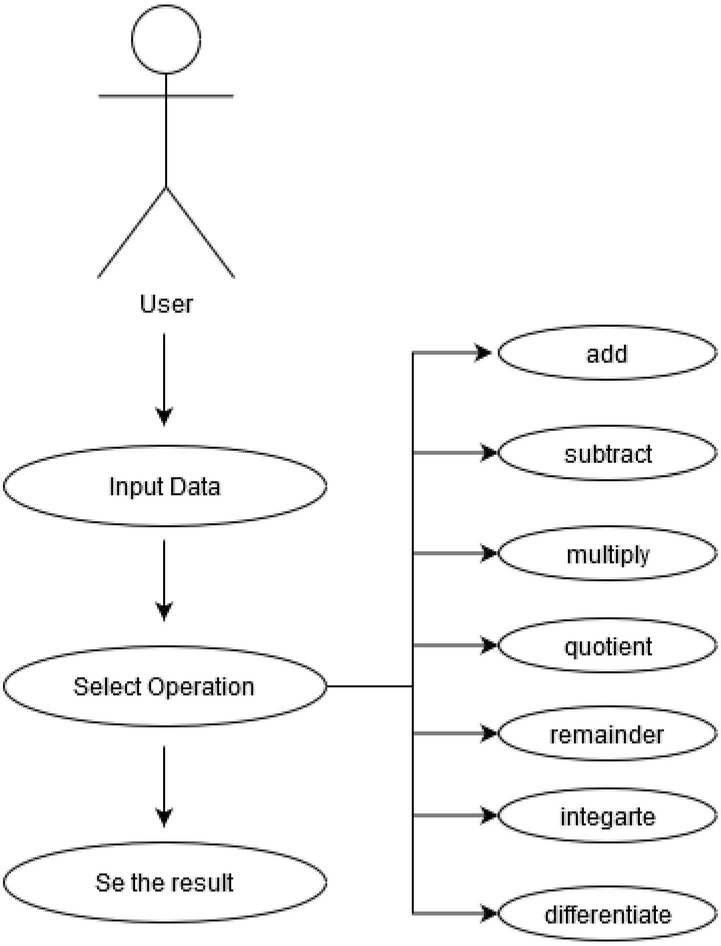
As far as our application is concerned, it must be capable of performing the operations mentioned above.

One big concern can be the division by zero, which is not allowed, so the user should not be able to do that.

We will assume that the user will only introduce polynomials whose coefficient is an integer and the power is a positive integer.

The accepted input will match the pattern ax^b±cx^d±…

Use Case:



Use case scenarios:

1) Addition:

Main success scenario:

- user enters two polynomials
- user presses “+” button
- the two polynomials are valid
- the sum of the two polynomials is computed and shown

Alternative sequence:

- One or both entered polynomials are not valid or one of the fields is still empty – a pop up window appears showing an error message
- The user can press “clear” for one/both polynomials and write again the polynomials

2) Subtraction:

Main success scenario:

- user enters two polynomials
- user presses “-” button
- the two polynomials are valid
- the difference of the two polynomials is computed and shown

Alternative sequence:

- One or both entered polynomials are not valid or one of the fields is still empty – a pop up window appears showing an error message
- The user can press “clear” for one/both polynomials and write again the polynomials

3) Multiplication:

Main success scenario:

- user enters two polynomials
- user presses “x” button
- the two polynomials are valid
- the multiplication of the two polynomials is computed and shown

Alternative sequence:

- One or both entered polynomials are not valid or one of the fields is still empty – a pop up window appears showing an error message
- The user can press “clear” for one/both polynomials and write again the polynomials

4) Division (quotient):

Main success scenario:

- user enters two polynomials
- user presses “/” button
- the two polynomials are valid
- the second polynomial is not 0
- the quotient of the division of the two polynomials is computed and shown

Alternative sequence:

- One or both entered polynomials are not valid or one of the fields is still empty – a pop up window appears showing an error message
- The second polynomial is 0, a pop window will show when attempting to divide by 0
- The user can press “clear” for one/both polynomials and write again the polynomials

5) Division (remainder):

Main success scenario:

- user enters two polynomials
- user presses “%” button
- the two polynomials are valid
- the second polynomial is not 0
- the quotient of the division of the two polynomials is computed and shown

Alternative sequence:

- One or both entered polynomials are not valid or one of the fields is still empty – a pop up window appears showing an error message
- The second polynomial is 0, a pop window will show when attempting to divide by 0
- The user can press “clear” for one/both polynomials and write again the polynomials

6) Integration:

Main success scenario:

- user enters a polynomial in the first text field
- user presses “∫” button
- the polynomial is valid
- the integration of the polynomial is computed and shown

Alternative sequence:

- The polynomial is not valid or the field is still empty – a pop up window appears showing an error message
- The user can press “clear” write again the polynomial

7) Differentiation:

Main success scenario:

- user enters a polynomial in the first text field
- user presses “f’” button
- the polynomial is valid
- the integration of the polynomial is computed and shown

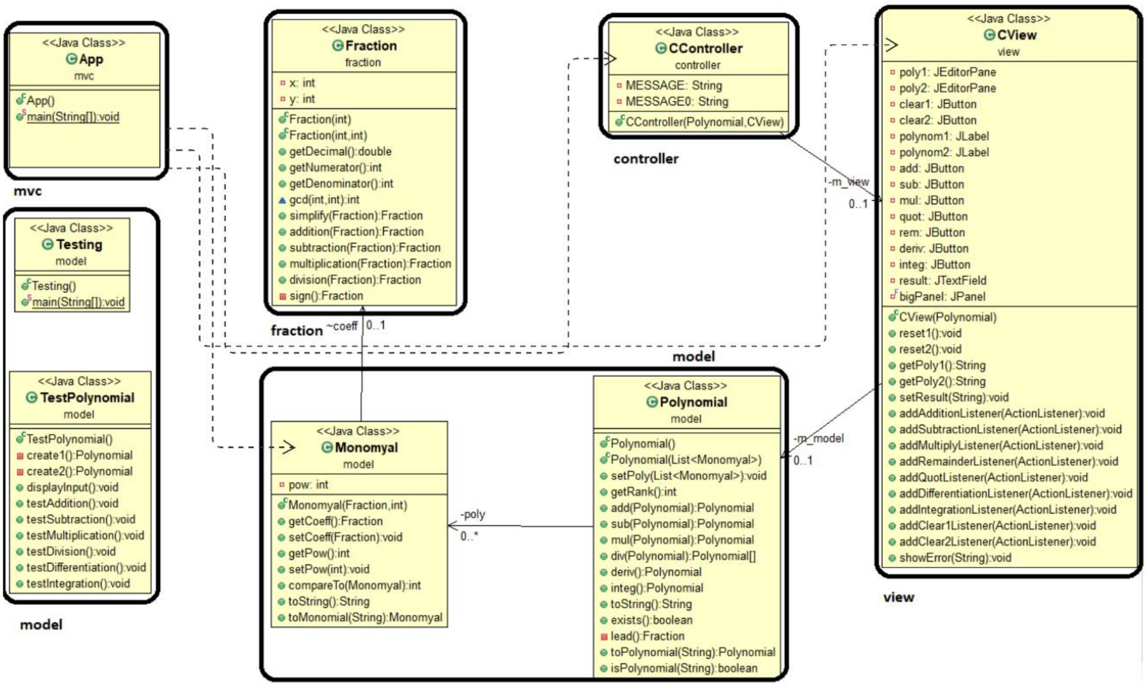
Alternative sequence:

- The polynomial is not valid or the field is still empty – a pop up window appears showing an error message
- The user can press “clear” write again the polynomial

Problem design

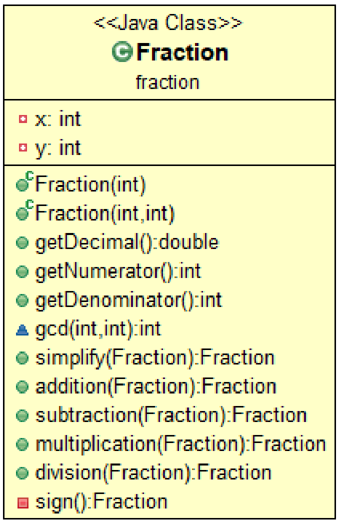
When designing the solution I have realized that the best implementation for me would be using 6 packages as it is shown above.

Apart from the packages model, view, controller and MVC I have decided to create another two packages, fraction and model (name coincidence was just my mistake, this package was intended for testing with JUnit).



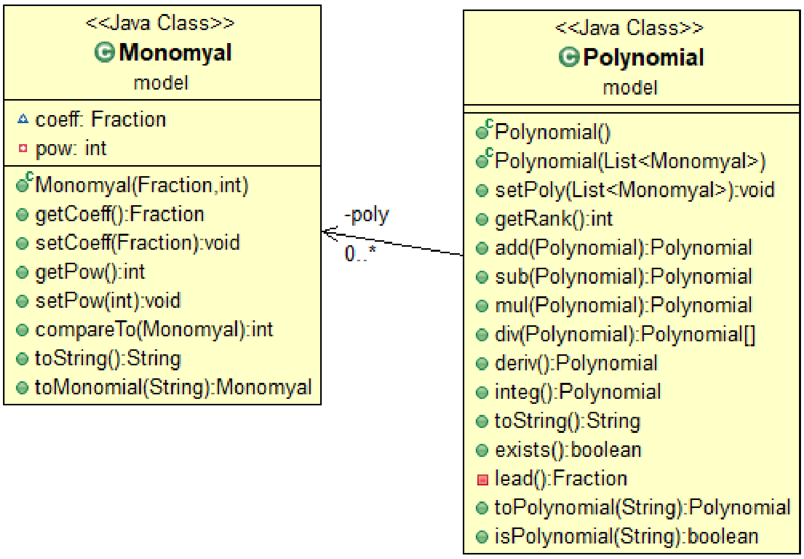
Package 1 : fraction

For the polynomial coefficients I have decided to use fractions, due to the fact that when division or integration are performed, we might need to store a non-integer coefficient.



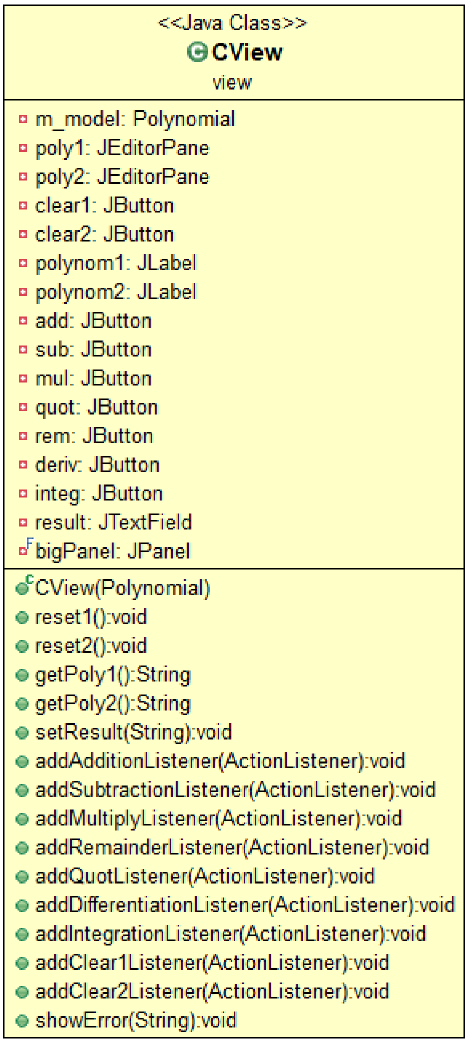
Package 2 : model

The second package is composed of two classes, polynomial and monomial. Polynomials are composed of an array of monomials, as requested and implement the operations mentioned in this document.



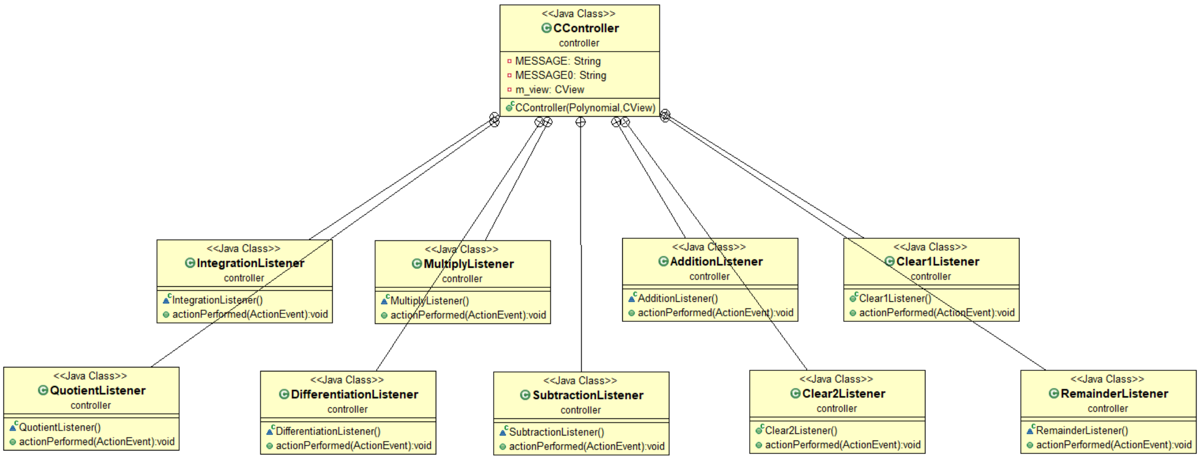
Package 3 : view

The third package contains the class view.



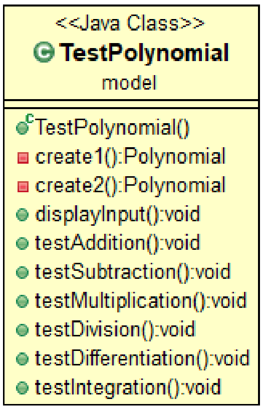
Package 4: controller

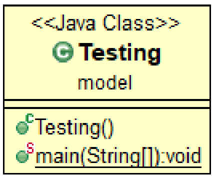
This package contains only one class, the controller and several inner classes for the listeners.

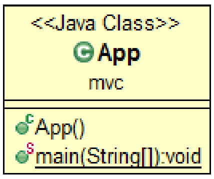


Package 5: mvc

This package contains only one class, the most important one, which is App, where you can run the actual application.



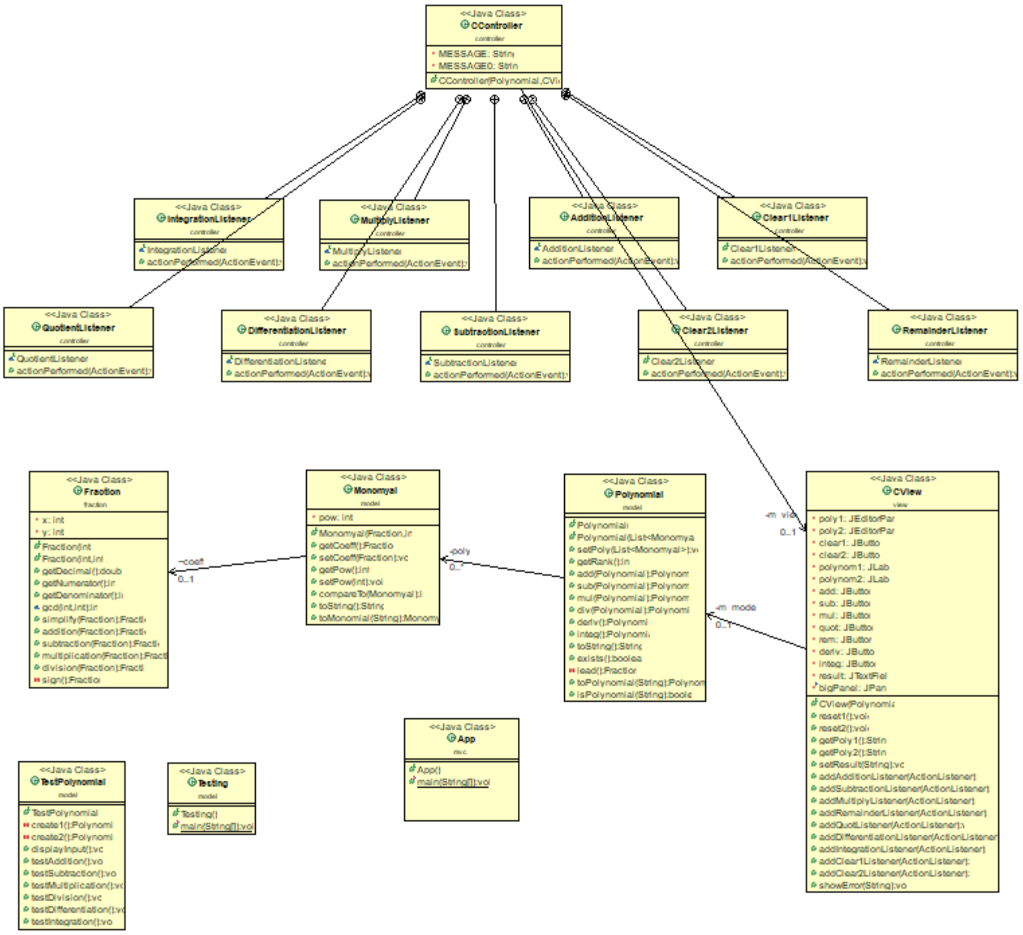




Package 6: model (although it has the same name as the previous, this one has been used for making the tests with JUnit)

This package has two independent classes which have been used for checking if the operations performed on the polynomials are correct.

The data structures that I have been working with in this problem are either primitive data types, especially integers, booleans and rarely chars, and a more complex one, such as ArrayList type object or new created object such as Monomial, Polynomial and Fraction.



Algorithms used:

Addition

Create a new polynomial having the power equal to the greatest power of the two polynomials, having initially all coefficients 0. Set all the terms of the result equal to the first polynomial. Then, add each term of the second polynomial to its corresponding power in the final result. This ensures no term is lost in case the two polynomials do not share all the powers.

Subtraction

Create a new polynomial having the power equal to the greatest power of the two polynomials, having initially all coefficients 0. Set all the terms of the result equal to the first polynomial. Then, subtract each term of the second polynomial to its corresponding power in the final result. This ensures no term is lost in case the two polynomials do not share all the powers.

Multiplication

Create a new polynomial having the power equal to the sum of powers of the two polynomials, having initially all coefficients 0. Then start multiplying each monomial from the first polynomial with those in the second one, and add the intermediate multiplication to the result, updating the coefficient of the corresponding power.

Division

For division I implemented the following algorithm. Remainder and quotient are saved in an array of polynomials.

```
function n / d is
    require d ≠ 0
    q ← 0
    r ← n             // At each step n = d × q + r

    while r ≠ 0 and degree(r) ≥ degree(d) do
        t ← lead(r) / lead(d)       // Divide the leading terms
        q ← q + t
        r ← r − t × d

    return (q, r)
```

For more information, see bibliography [2].

Differentiation

Create a new polynomial where each monomial from the input polynomial is obtained as follows: for each monomial having power greater than 0, multiply the coefficient by its power and subtract 1 from the power.

Integration

Create a new polynomial where each monomial from the input polynomial is obtained as follows: for each monomial divide the coefficient by its power and increase power by 1.

User interface

The user interface is a very friendly and easy to use one. I have decided to add a field for each polynomial, with the possibility of clearing the text which has been typed. The user can select the operation he/she wants to perform from the seven options and the result will be displayed in the appropriate field if the string(s) given as input is/are correct or an error message will pop up in case the opertion cannot be performed.

Implementation

Class description

Package 1 : fraction

Class Fraction :

- Used to represent the coefficient as a fraction, having a numerator and a denominator
- Can perform basic operations like addition, subtraction, multiplication and division, simplification
- A special function called sign needed to be implemented because for some function the sign “-“ was saved in the denominator not the numerator
- Has two constructors, one of them has one parameter and it is used when the coefficient is an integer and the denominator is set automatically to 1 and another one with two parameters when we deal with a rational number represented as a fraction

Package 2: model

Class Monomyal:

- Used to represent a monomial with a coefficient (Fraction) and a power
- Implements only getters, setters, compareTo and toString and a function which transforms a string in a monomial (if the string does not contain a proper monomial, a 0 monomial is returned)

Class Polynomial:

- Represents a polynomial as a list of monomials
- Implements the requested operations: addition, subtraction, multiplication, division, integration, differentiation which return a polynomial, respectively an array of polynomials for division, where the first polynomial represents the quotient and the other is the remainder
- Implements a toString() method which creates string like a_nx^n+a_{n-1}x^(n-1)+…+a_0 from the array of monomials, returns a string
- Implements a toPolynomial (String) method which receives a string as an input and converts it into a polynomial only if the string matches the required format, returns a polynomial
- Implements an isPolynomial(String) method which receives a string as an input and returns a Boolean, true if the string represents a polynomial, false otherwise
- Implements a lead() function which returns the coefficient(Fraction) of the leading monomial

Package 3: view

Class View:

- It is a class extending JFrame, used for the graphical interface
- For each input there is a JEditorPane and for each operation there is a button labeled accordingly
- The result is displayed in a JTextField
- There are also two buttons designed to clear the specific polynom, clearing the polynom also clears the result unless the text field is already empty
- The constructor has the purpose of initializing the frame and setting its characteristic properties like DefaultCloseOperation,size, visibility, layout, and add graphic elements on it, regarding the layouts, I used GridLayout() for the buttons because it allows easier customization

Package 4: controller

Class Controller:

- Contains three private fields, two of them are the error messages and the third is the view
- The constructor initializes the view and calls the methods implemented by the view
- Provides anonymous class listener for handling the events on the application

Package 5: mvc

Class App:

- Contains only the main method where a controller, a view and a model are defined

Package 6: model

Class Testing:

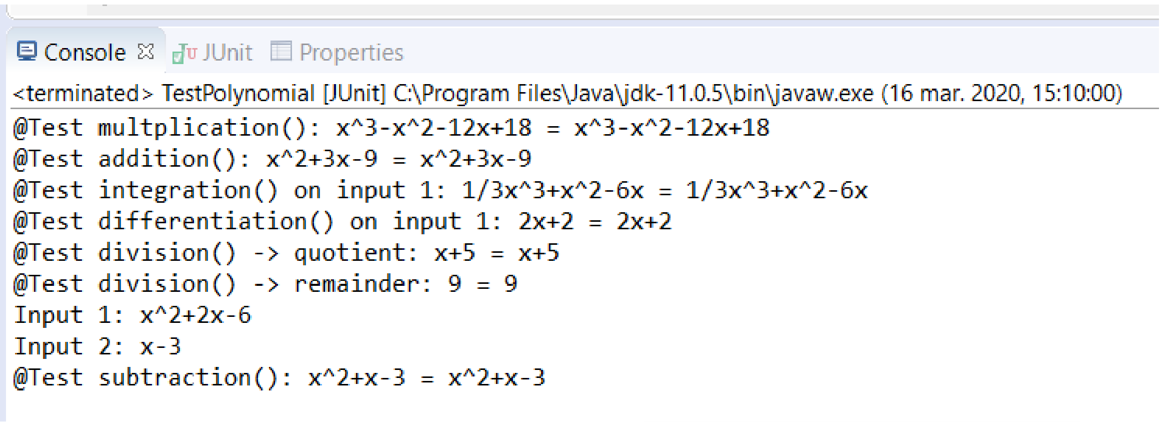
- This is class I have used to test if polynoms are created and displayed properly and also if the operations are performed as expected, but without using JUnit

Class TestPolynomial:

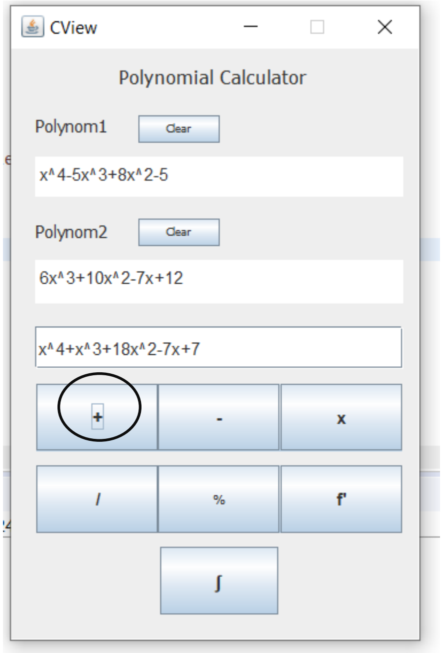
- This class uses JUnit to verify the correctness of the operations performed and to assert any irregularities
- Contains only methods
- There are two private methods meant to create two polynomials (used as inputs for the operations)
- There is also a public test method for each available operation

Results and test cases

Operation	Input	Expected result	Actual result	Pass/fail
+	x^2+2x-6 x-3	x^2+3x-9	x^2+3x-9	Pass
-	x^2+2x-6 x-3	x^2+x-3	x^2+x-3	Pass
X	x^2+2x-6 x-3	x^3-x^2-12x+18	x^3-x^2-12x+18	Pass
/	x^2+2x-6 x-3	x+5	x+5	Pass
%	x^2+2x-6 x-3	9	9	Pass
f’	x^2+2x-6	2x+2	2x+2	Pass
∫	x^2+2x-6	1/3x^3+x^2-6x	1/3x^3+x^2-6x	pass



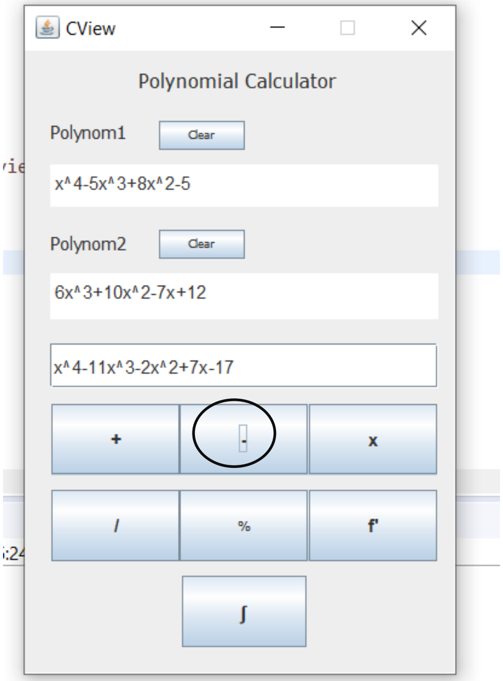
```
<terminated> TestPolynomial [JUnit] C:\Program Files\Java\jdk-11.0.5\bin\javaw.exe (16 mar. 2020, 15:10:00)
@Test multplication(): x^3-x^2-12x+18 = x^3-x^2-12x+18
@Test addition(): x^2+3x-9 = x^2+3x-9
@Test integration() on input 1: 1/3x^3+x^2-6x = 1/3x^3+x^2-6x
@Test differentiation() on input 1: 2x+2 = 2x+2
@Test division() -> quotient: x+5 = x+5
@Test division() -> remainder: 9 = 9
Input 1: x^2+2x-6
Input 2: x-3
@Test subtraction(): x^2+x-3 = x^2+x-3
```



Polynom1: x^4-5x^3+8x^2-5

Polynom2: 6x^3+10x^2-7x+12

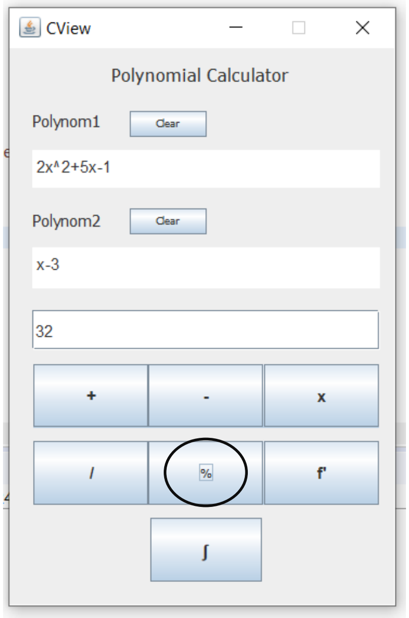
x^4+x^3+18x^2-7x+7



Polynom1: x^4-5x^3+8x^2-5

Polynom2: 6x^3+10x^2-7x+12

x^4-11x^3-2x^2+7x-17



Polynomial Calculator

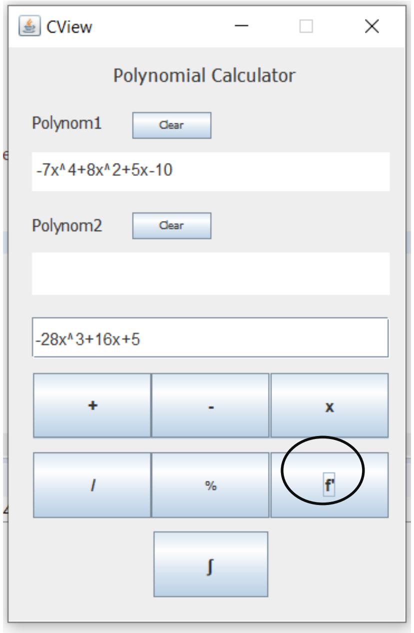
Polynom1

2x^2+5x-1

Polynom2

x-3

32



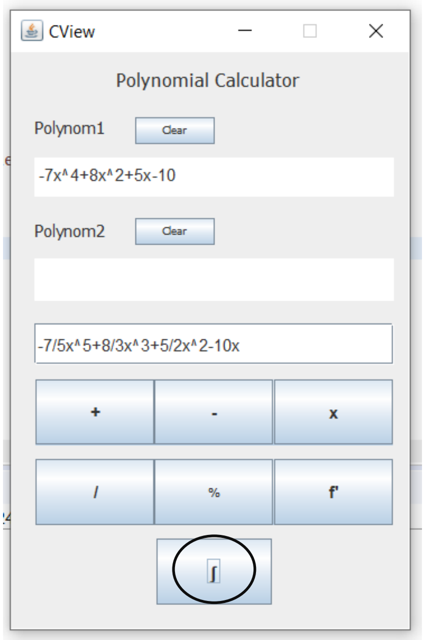
Polynomial Calculator

Polynom1

-7x^4+8x^2+5x-10

Polynom2

-28x^3+16x+5



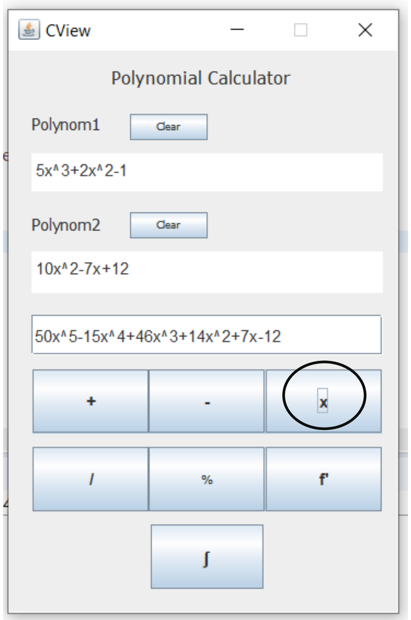
Polynomial Calculator

Polynom1

-7x^4+8x^2+5x-10

Polynom2

-7/5x^5+8/3x^3+5/2x^2-10x



Polynomial Calculator

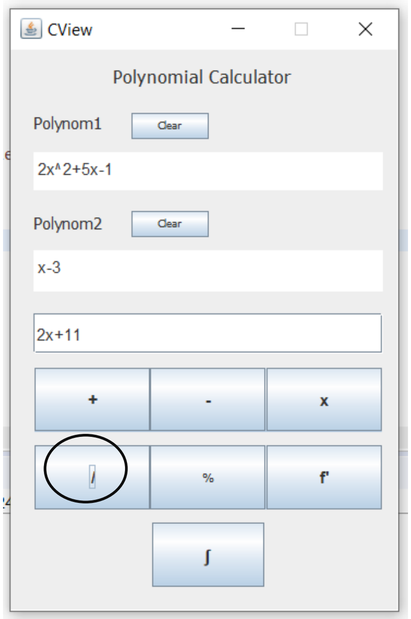
Polynom1

5x^3+2x^2-1

Polynom2

10x^2-7x+12

50x^5-15x^4+46x^3+14x^2+7x-12



Polynomial Calculator

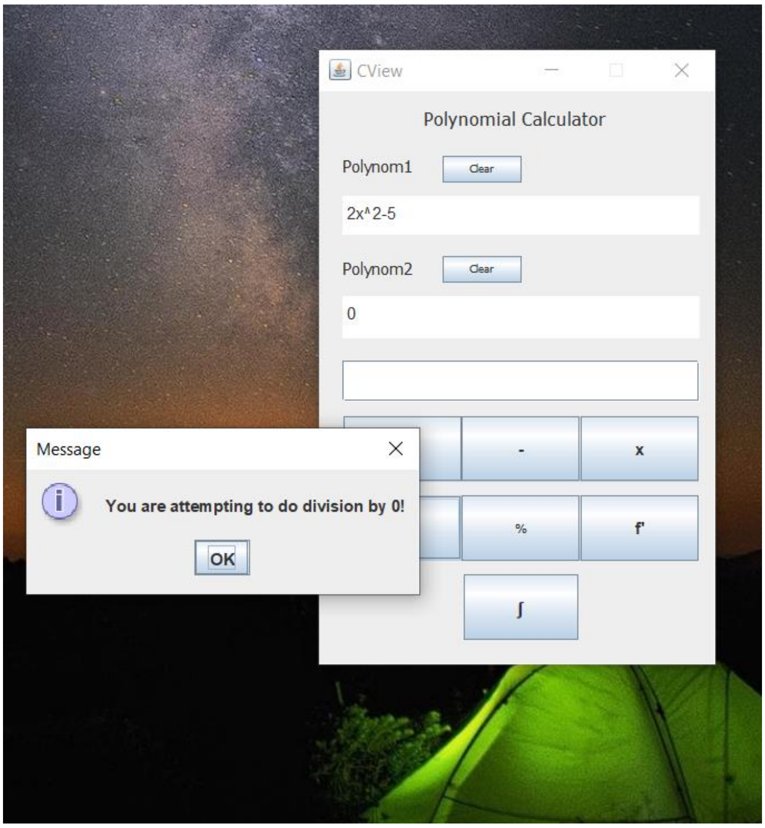
Polynom1

2x^2+5x-1

Polynom2

x-3

2x+11



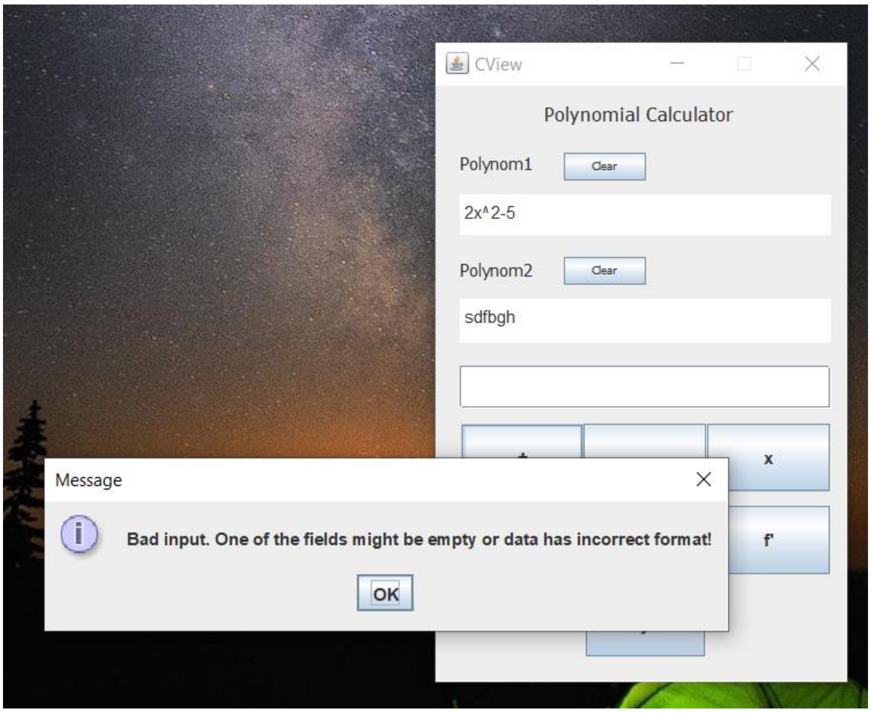
Polynomial Calculator

Polynom1: 2x^2-5

Polynom2: 0

Message

You are attempting to do division by 0!



Polynomial Calculator

Polynom1: 2x^2-5

Polynom2: sdfbgh

Message

Bad input. One of the fields might be empty or data has incorrect format!

Conclusions

This project has taught me the importance of thinking about dealing with user inputs and how important test cases are. As the creator of the application I know how to parse the input and which requirements it should fulfill, but I also had to think that the user might not know that and be tempted to introduce wrong data.

I have also learnt to use Regex and Junit Tests, which made checking inputs and verifying the overall correctness of the program much more easier computing the square of the polynomial, compute the value of a polynomial in a certain point or finding the roots of a second degree polynomial.

Bibliography

1. https://www.tutorialspoint.com/java/java_regular_expressions.htm
2. https://en.wikipedia.org/wiki/Polynomial_long_division
3. https://www.javatpoint.com/junit-tutorial
4. https://www.youtube.com/watch?v=ahbE30-kNmk&t=422s // java swing calculator
5. https://www.leepoint.net/GUI/structure/40mvc.html
6. https://docs.oracle.com/javase/7/docs/api/java/util/regex/Matcher.html