Technical University of

Cluj-Napoca

Faculty of Automation

and

Computer Science

Polynomial Processing System

Discipline: Programming Techniques

Date: 19.03.2018

Gavril Luca

Group:30422

1.Main objective:

Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.

1.2.In-depth objectives:

The aim of the project is to implement arithmetic operations, on 2 different polynomials that are given as input. The system is considered to be a multiple phase process:

- Give input
- Performing the desired operation
- Output the result

The operations considered for this system are:

- Addition of two polynomials.
- Subtraction of two polynomials.
- Multiplication of two polynomials.
- Differentiation of a polynomial.
- Integration of a polynomial.

The expression of the polynomial is considered to be of one variable, and of constant integer coefficients. The final two operations are performed on one polynomial, not two. The Implementation section contains thorough details about the system.

2.Analysis:

2.1.Definition:

A polynomial is a mathematical expression consisting of a sum of terms, each term including a variable or variables raised to a power and multiplied by a coefficient. The simplest polynomials have one variable. A one-variable (univariate) polynomial of degree n has the following form:

$$a_n x^n + a_n\text{-}1x^n\text{-}1 + \ldots + a_2 x^2 + a_1 x^1 + ax$$

where the a’s represent the coefficients and x represents the variable.

The sum of two polynomials is obtained by adding together the coefficients of the variable having the same degree, as such:

$$\begin{array}{r} x^5+3x^3+4 \\ +\ 6x^6+\quad +4x^3 \\ \hline 6x^6+x^5+7x^3+4 \end{array}$$

The difference of two polynomials is obtained by subtracting the coefficients of the variable having the same degree, as such:

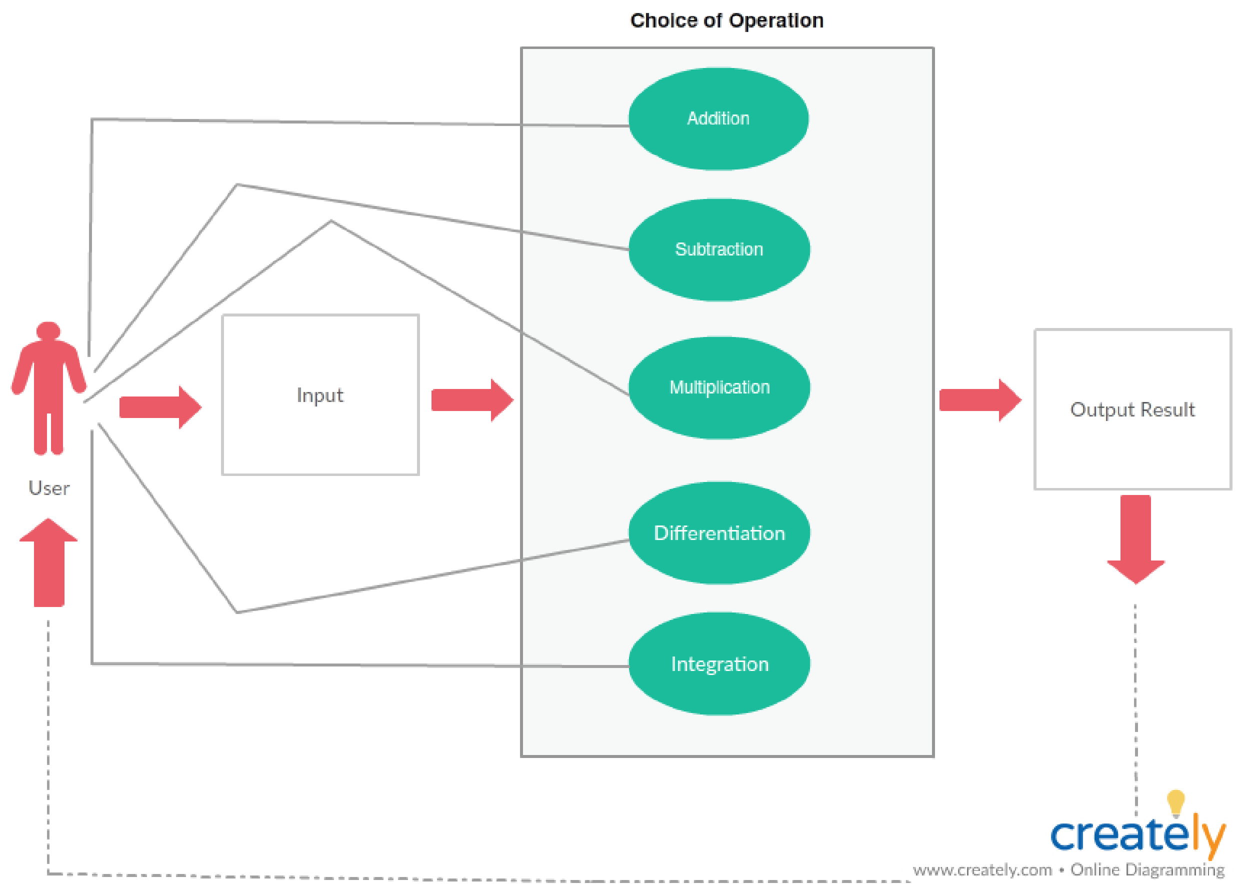
$$\begin{aligned} &(-5x^3-2x^2+7)-(4x^3+7x^2-3x+2) \\ &=-5x^3-2x^2+7-4x^3-7x^2+3x-2 \\ &=-9x^3-9x^2+3x+5 \end{aligned}$$

The multiplication of two polynomials is obtained by taking each member of the first polynomial, and multiplying that member with each member of the second polynomial.

$$\begin{aligned} &(x+2)(x-3) \\ &=x\cdot x+x(-3)+2\cdot x+2(-3) \\ &=x^{1+1}-3x+2x-6 \\ &=x^2-x-6 \end{aligned}$$

For integration and differentiation, we respect the rules regarding the two operations (in the case of polynomials, both are quite easy operations as we integrate/differentiate term-by-term).

2.2.Use-case diagram:



The user is presented with the possibility to either input 1 or 2 polynomials, depending on which operation he choses to get the result of.

The polynomial expression has to be input as such:

Expression: 4x^4+2x^2+2x+2;

Input: 4x^4+2x^2+2x^1+2;

It is necessary to not forget the “1” exponent, otherwise the program will notice the error or will produce a wrong result. It is also wished that the user inserts the polynomials starting with the highest exponent. More on this, in the Errors section.

The result will show once the desired operation is selected.

2.3.Assumptions:

Given the previous explanation, we ought to assume that the user is capable of inserting a correct input and that if the user inserts a wrong input, he will be capable of changing to a proper insertion.

2.4.Errors:

The user will be signaled by the program if the input he has inserted is correct or not. Also, the result will obviously be wrong.

3.Design:

The project is designed in a manner of object-oriented programming. Each polynomial is perceived as an array list of monomials. Each monomial has a coefficient(real or integer), and an exponent that is always an integer. Because of this, besides the superclass Monomial, there are 2 extra subclasses defined : IntegerMonomial and RealMonomial. As presumed, IntegerMonomial has the coefficient of integer type, while RealMonomial has its coefficient of double type. Both subclasses hold an integer exponent.

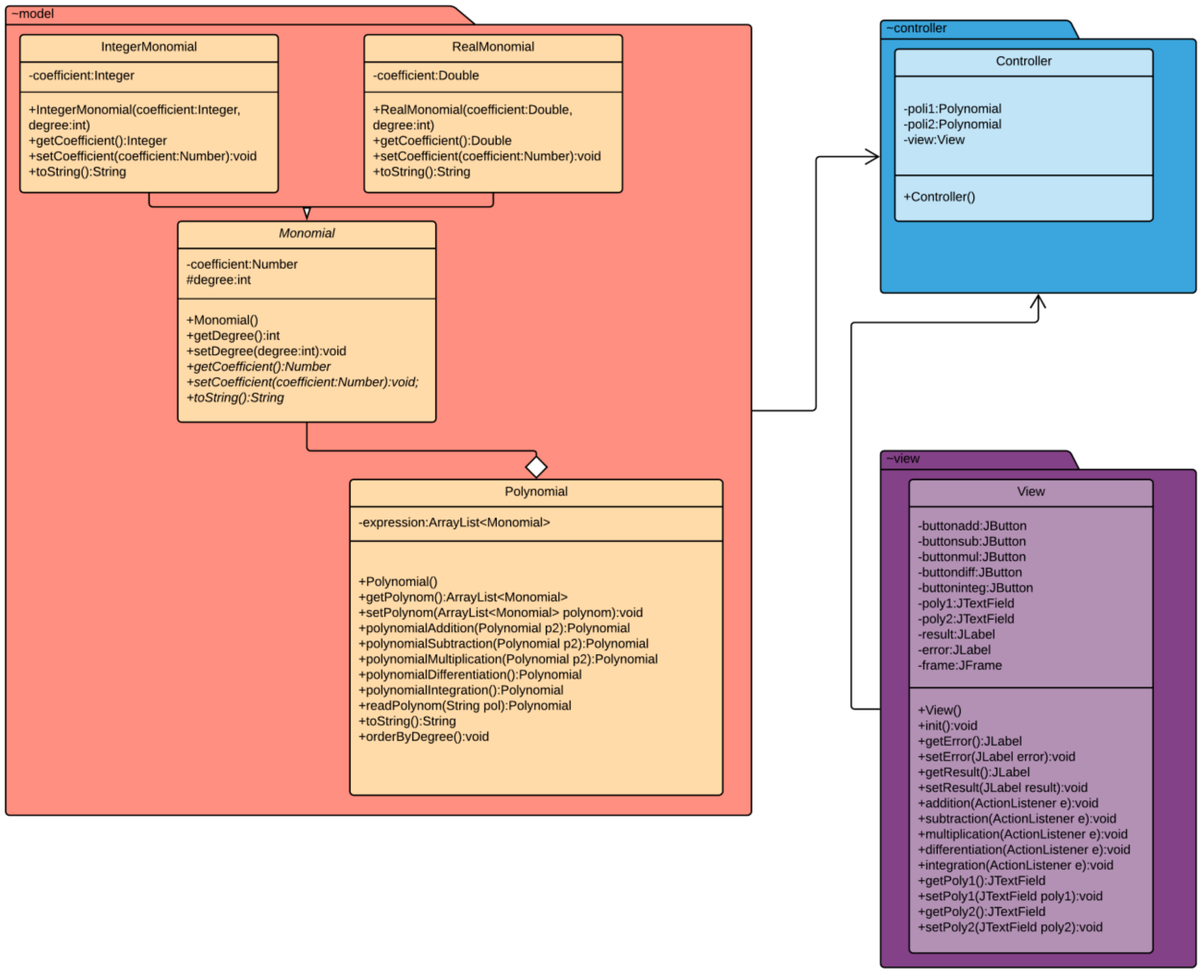
For the superclass Monomial, we used the type Number for the coefficient.

Being that the polynomial is a list of monomials, between the classes Monomial and Polynomial, there exists an aggregation relationship. An inclusion in the ArrayList of monomials would have the following structure:

<index(type int),Monomial(coefficient,exponent)>

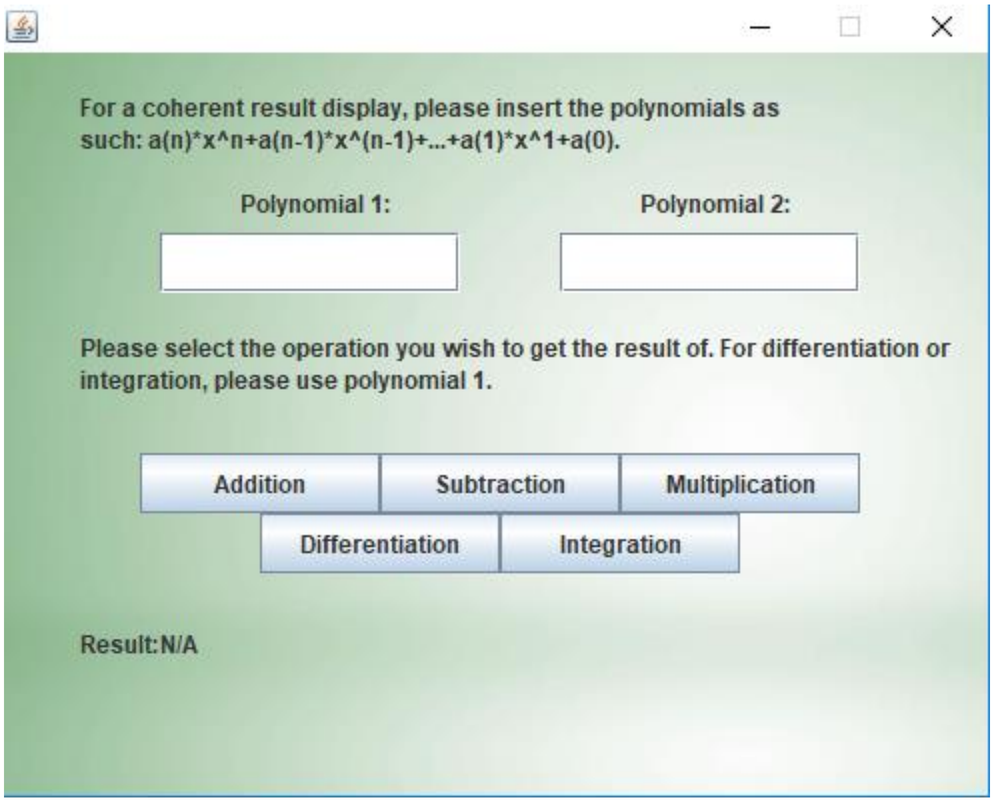
Given that the way an user inserts the input starts with the highest exponent, in general, the element at index 0 in the ArrayList will be the monomial of the highest exponent, thus giving the degree of the polynomial. For the project at hand, the ArrayList and the way it is used in the code proved to be a good choice of storing monomials. While there was the possibility to store the monomials in a List of type : <key(exponent),value(coefficient)> , because of the way the polynomials are read, having the exponents decrease as an iterator moves through the list helped a lot with coding the operations.

3.1.UML Diagram:


~model
IntegerMonomial
-coefficient:Integer
+IntegerMonomial(coefficient:Integer, degree:int)
+getCoefficient():Integer
+setCoefficient(coefficient:Number):void
+toString():String
RealMonomial
-coefficient:Double
+RealMonomial(coefficient:Double, degree:int)
+getCoefficient():Double
+setCoefficient(coefficient:Number):void
+toString():String
Monomial
-coefficient:Number
#degree:int
+Monomial()
+getDegree():int
+setDegree(degree:int):void
+getCoefficient():Number
+setCoefficient(coefficient:Number):void;
+toString():String
Polynomial
-expression:ArrayList<Monomial>
+Polynomial()
+getPolynom():ArrayList<Monomial>
+setPolynom(ArrayList<Monomial> polynom):void
+polynomialAddition(Polynomial p2):Polynomial
+polynomialSubtraction(Polynomial p2):Polynomial
+polynomialMultiplication(Polynomial p2):Polynomial
+polynomialDifferentiation():Polynomial
+polynomialIntegration():Polynomial
+readPolynom(String pol):Polynomial
+toString():String
+orderByDegree():void
~controller
Controller
-poli1:Polynomial
-poli2:Polynomial
-view:View
+Controller()
~view
View
-buttonadd:JButton
-buttonsub:JButton
-buttonmul:JButton
-buttondiff:JButton
-buttoninteg:JButton
-poly1:JTextField
-poly2:JTextField
-result:JLabel
-error:JLabel
-frame:JFrame
+View()
+init():void
+getError():JLabel
+setError(JLabel error):void
+getResult():JLabel
+setResult(JLabel result):void
+addition(ActionListener e):void
+subtraction(ActionListener e):void
+multiplication(ActionListener e):void
+differentiation(ActionListener e):void
+integration(ActionListener e):void
+getPoly1():JTextField
+setPoly1(JTextField poly1):void
+getPoly2():JTextField
+setPoly2(JTextField poly2):void

The UML Diagram represents the MVC model that the project's design is based upon. In the model package, we can see the inheritance relationship between the superclass Monomial and the subclasses RealMonomial and IntegerMonomial. The relationship between Monomial and Polynomial is one of aggregation. Because in a MVC model , the Controller/Model and Controller/View groupings need to be able to communicate, we describe these connections with 2 association relationships.

3.2.Graphic User Interface:


For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1:

Polynomial 2:

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

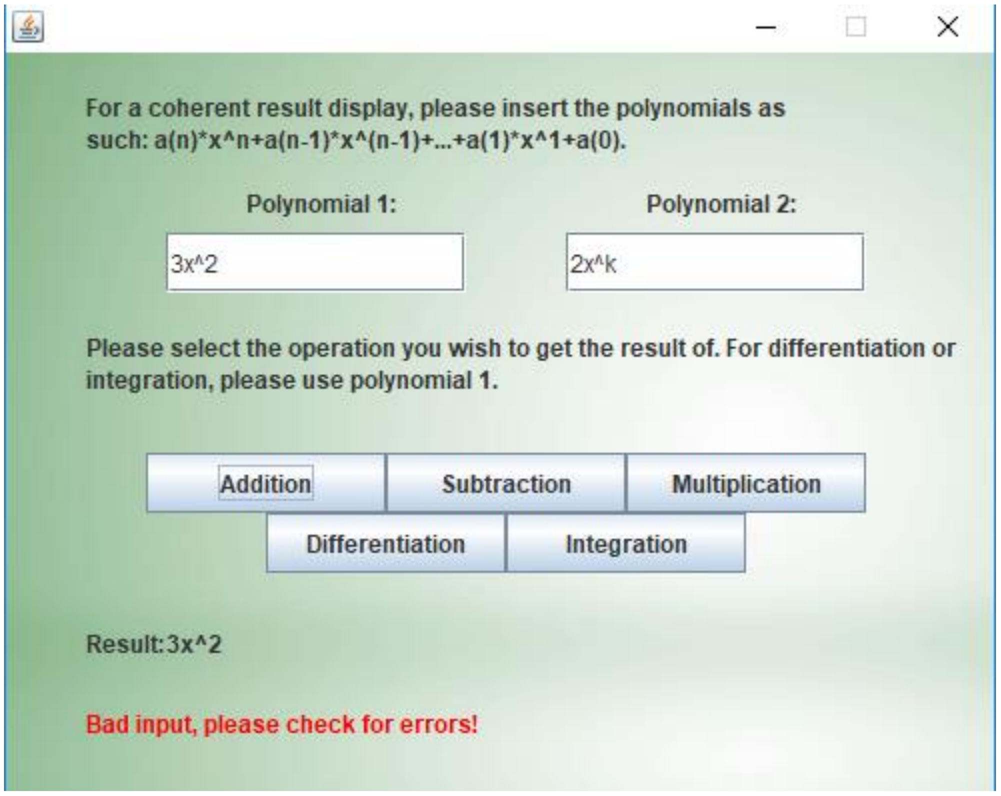
Result:N/A

As it can be seen, the GUI provides the user with 2 field, where he will be able to insert the desired values for the two polynomials. After inserting the data, the user is free to choose which operation to execute, by clicking on the button specific to said operation. After the operation is completed , the result will appear on the bottom of the screen, if the input data was inserted correctly. As we can see, the result is "N/A" when the application has booted.

The result will be a polynomial, converted to a string, so it can appear in the frame.

Two help labels are provided for the user, in order to help with the input data and the operation choice.

Also, there exists an error label, that checks whether or not the input data is correct. If the input placed is correct, the label will show "**Ok**".


For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1: 3x^2

Polynomial 2: 2x^k

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

Result:3x^2

Bad input, please check for errors!

4.Implementation:

4.1.model package:

The model package holds 4 different classes:

- Monomial
- IntegerMonomial
- RealMonomial
- Polynomial

Monomial Class:

The Monomial class is defined as being abstract, because it holds the abstract methods getCoefficient(),setCoefficient() and toString(), which are required to be implemented in both of its subclasses. The class has two attributes : coefficient of type Number and degree of type int, both of which are characteristic to a monomial.

The constructor of the class simply initializes a new Monomial, the user has the possibility to initialize an IntegerMonomial or RealMonomial object. Besides the constructor and the abstract methods the class also holds a getter and a setter for the degree attribute, that holds the same type of value for either of Monomial's subclasses.

IntegerMonomial Class:

This class extends the superclass Monomial. It has one attribute, an Integer type coefficient. Of course, the reason we created the subclass was for using monomials with integer coefficients, which means that the constructor holds as parameter a variable of type Integer for the coefficient and another of type int for the degree. We implement the getCoefficient() method, of type Integer, as well as the setCoefficient() method, where we cast (Integer) over the coefficient parameter. Inside the class we also implemented the toString() method, which will be specific for integer coefficient monomials, taking into account numerous cases: if the coefficient is 0 , -1, 1, etc.

RealMonomial Class:

This class extends the superclass Monomial. It has one attribute, an Integer type coefficient. Of course, the reason we created the subclass was for using monomials with real coefficients, which means that the constructor holds as parameter a variable of type Double for the coefficient and another of type int for the degree. We implement the getCoefficient() method, of type Double, as well as the setCoefficient() method, where we cast (Double) over the coefficient parameter. Inside the class we also implemented the toString() method, which will be specific for real coefficient monomials, taking into account numerous cases: if the coefficient is 0 , -1, 1, etc.

Polynomial Class:

This class has as attribute an ArrayList of Monomials, defined by name as "expression". The class holds a getter and a setter for the polynomial's expression, a method that takes the string from the input and reads the polynomial, a method that orders the elements of a polynomial's expression by each element's degree and a toString() method, specific to the Polynomial class (we make use of the toString()

overridden methods from the monomial classes). Besides these methods, the class has the operation methods:

- polynomialAddition;
- polynomialSubtraction;
- polynomialMultiplication;
- polynomialIntegration;
- polynomialDifferentiation;

The method polynomialAddition(Polynomial p2) takes the elements of the two polymonials and stores each of them in the expression of the result polynomial. In order for the addition to work, we then move through the expression of the result, and if we find elements that have the same degree, we add them. We must of course eliminate one of the two elements that have the same degree, so the result is correct.

The method polynomialSubtraction(Polynomial p2) takes the elements of the two polymonials and stores each of them in the expression of the result polynomial. Of course, the elements of the second polynomial have to be stored with a negative sign, because we must attend to the rule: minus changes the sign. In order for the subtraction to work, we then move through the expression of the result, and if we find elements that have the same degree, we add them. We must of course eliminate one of the two elements that have the same degree, so the result is correct.

The method polynomialMultiplication(Polynomial p2) saves the result of the multiplication between the 2 input polynomials in polynomial 1.

The algorithm is quite straight forward, we multiply each element of the first polynomial, with each element of the second polynomial. After we get our result(an ArrayList), we must check to see if any two elements of the result hold the same degree. If so, we add them.

The method polynomialIntegration() saves the result of the integration of polynomial 1 in that same object. For integration, we followed the mathematical algorithm of integrating a monomial. The integration method only uses the first polynomial of the GUI, so the user has to remember that he must insert a new value in the second polynomial's text field, so he can use the other operations.

The method polynomialDifferentiation() saves the result of the differentitation of polynomial 1 in that same object. For differentiation, we followed the mathematical algorithm of differentiating a monomial. Like the integration method, the differentiation method only uses the first polynomial in the GUI.

4.2.controller package:

The controller package holds one class : **Controller**.

The class has three private attributes, two Polynomials poli1, poli1, and a view object of type View. The Controller class has a sole method, the constructor. The constructor calls the initialization of the three private attributes and also makes the View object visibile when the application is ran.(view.init()). The constructor then makes the right calls towards all the ActionListener methods of class View. Each call is made for a specific button on the GUI and usually contains the following steps:

- get input data from text fields
- try to read the polynomials
- if not, catch the error and alert the user
- do operation
- set result

4.3.view package:

The view package holds one class : **View**.

The class has a number of private attributes, all important for building the GUI, and for the implementation of the Controller() class. The View() class has a constructor : public View() where we build the JFrame itself. The constructor adds buttons, labels, text fields to the frame. We also have a init() method which allows the frame to appear on screen. The class also has getters and setters for the private attributes that we use outside of this class. The class also contains 5 different ActionListener methods, for the 5 arithmetic operations the processing system offers.

5.Testing with JUnit:

All the operation methods are tested with the help of the JUnit Framework offered by Java. We hard-code a result as we see fit considering the input that the operation will receive, and we compare this result to the actual method solution.

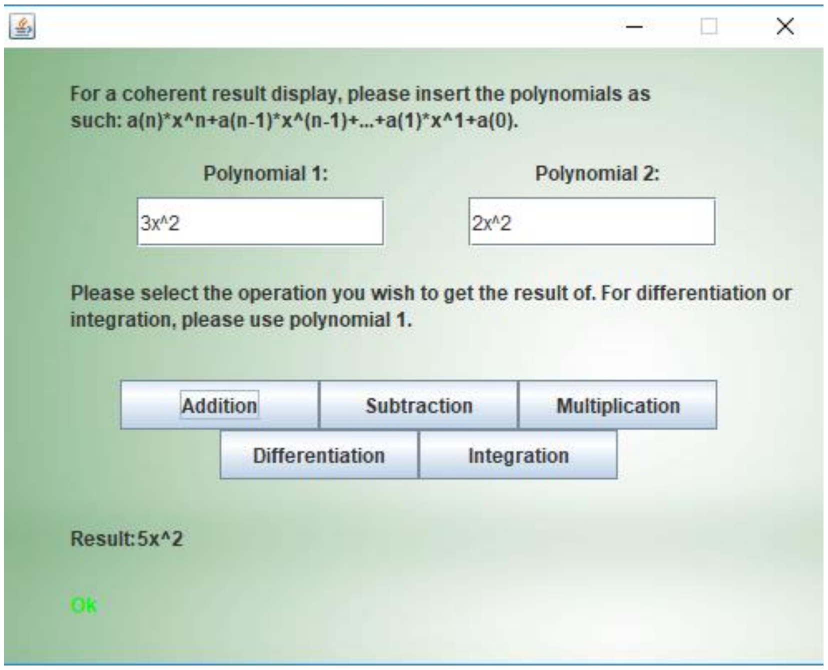
The class AssignmentTests contains the following methods:

- testAddition();
- testSubtraction();
- testMultiplication();
- testIntegration();
- testDifferentiation();

6.Results:

In this section we will show the results of the operations for the polynomials : 3x^2 and 2x^2. In the case of integration and differentiation, we use the first polynomial only.

Addition: Correct Result.


For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1: 3x^2

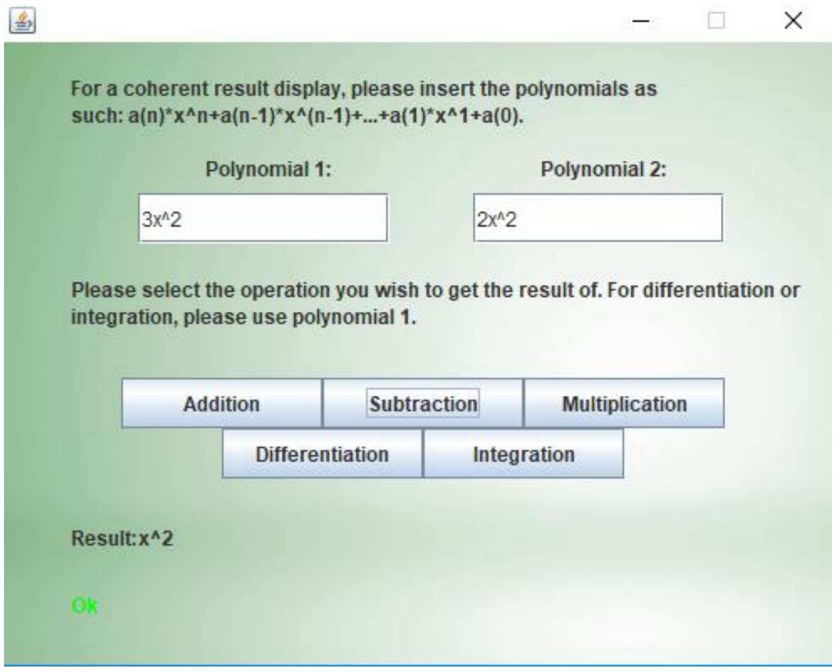
Polynomial 2: 2x^2

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

Result:5x^2

Ok

Subtraction: Correct Result.


For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1: 3x^2

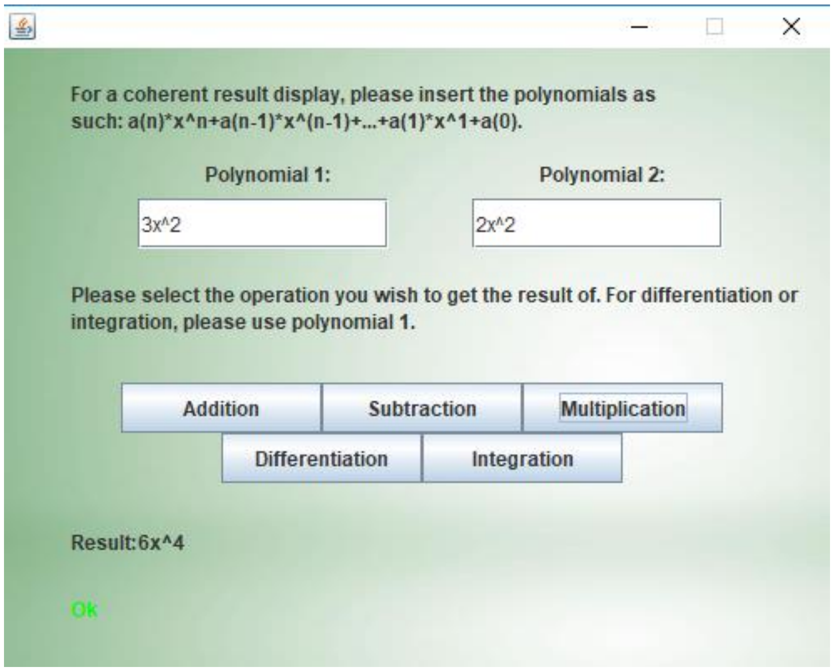
Polynomial 2: 2x^2

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

Result:x^2

Ok

Multiplication: Correct Result.


For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1: 3x^2

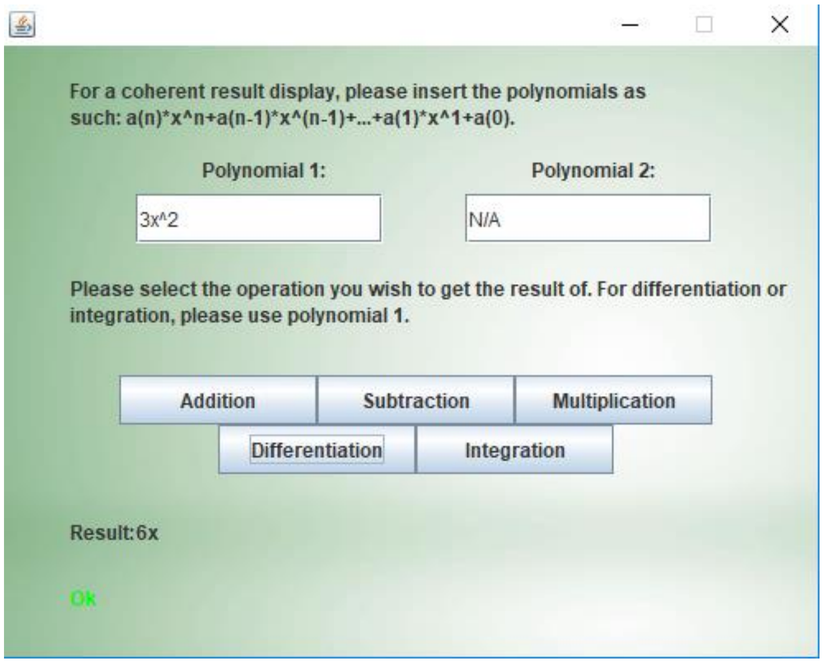
Polynomial 2: 2x^2

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

Result:6x^4

Ok

Differentiation: Correct Result.



For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1: 3x^2

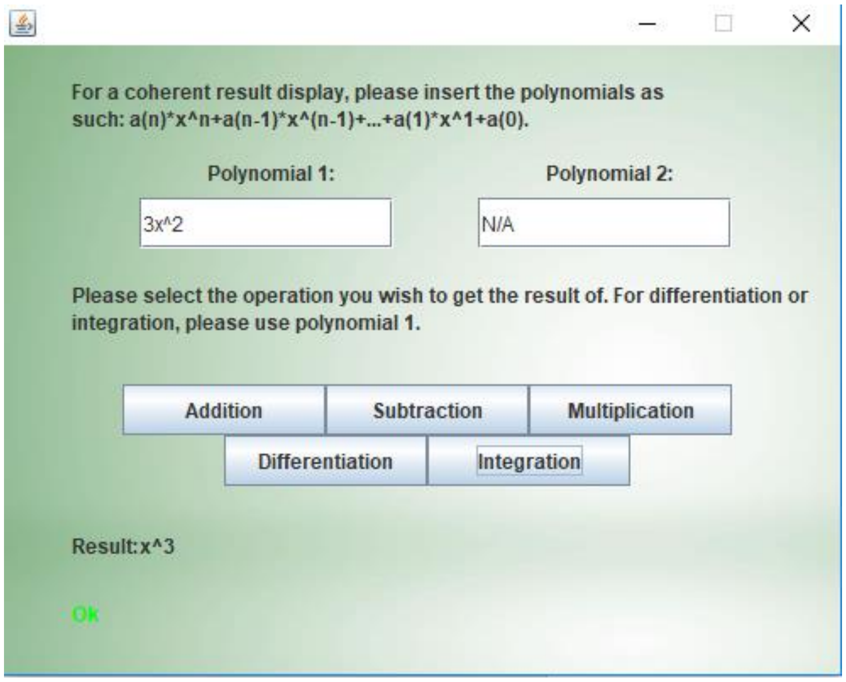
Polynomial 2: N/A

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

Result:6x

Ok

Integration: Correct Result.



For a coherent result display, please insert the polynomials as such: a(n)*x^n+a(n-1)*x^(n-1)+...+a(1)*x^1+a(0).

Polynomial 1: 3x^2

Polynomial 2: N/A

Please select the operation you wish to get the result of. For differentiation or integration, please use polynomial 1.

Result:x^3

Ok

Though not in this case, the integration method's result is a polynomial of real monomials (no fractions for the coefficients, but real numbers).

7.Conclusion:

The project has offered me the chance to understand the concepts of object-oriented programming and Java at a higher level. Also, it has taught me how to better make use of inheritance and it has allowed me to become more accustomed to the MVC design pattern.

Further developments:

- a more dynamic input insertion , not requiring to only use "x" as the variable.
- additional operations such as root finding, multiplication by scalar, etc.
- a full implementation for the division operation.
- A polynomial processing system that works with any given number of polynomials, or a set larger number than 2.

8.Bibliography:

- www.webmath.com
- www.wikipedia.org
- www.symbolab.com
- www.wolframalpha.com
- www.creately.com