MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

Assignment 1

Polynomial Processing System

- Documentation -

20.03.2018

Linca Paul Tudor

Group: 30422

MINISTERUL EDUCAŢIEI NAŢIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

1. Task:

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

2. Brief description:

This assignment consists of developing an application which receives as input two polynomials represented by two strings and has as output a string containing the result of an arithmetic operation between the first two polynomials or just the first one, thus the output is also a polynomial.

The arithmetic operations that will be implemented are: addition, subtraction, multiplication, division, differentiation and integration. The first four operations will be done using both the input polynomials, while the latter two will be done using only one of them (I considered the first polynomial for the two operations).

3. Defining the Polynomial

A polynomial is a mathematical expression of one or more algebraic terms each of which consists of a constant multiplied by one or more variables raised to a nonnegative integral power.

A monomial is a polynomial which has only one term.

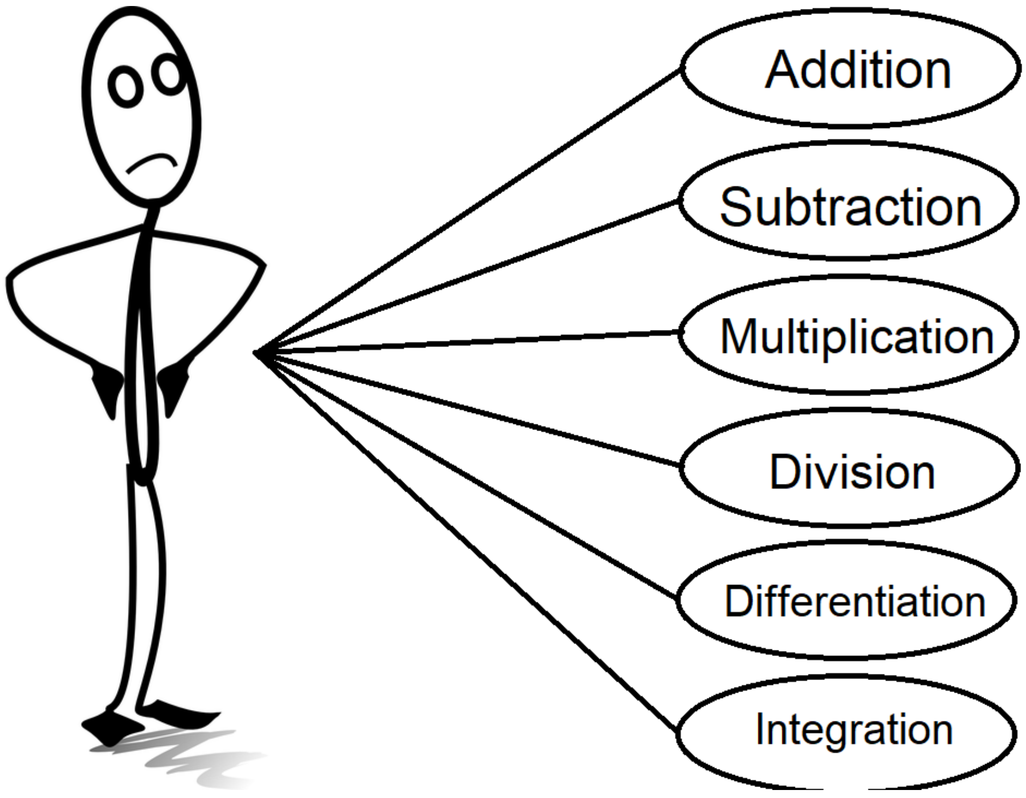
A polynomial's structure looks like this:

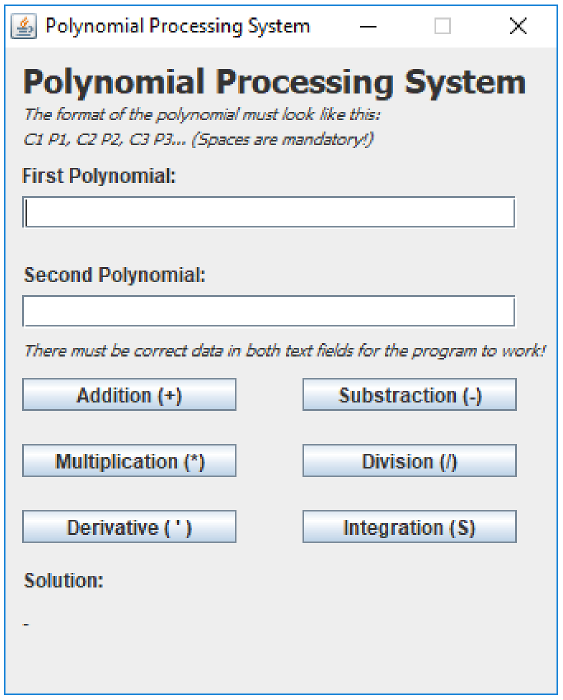
$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

3.1. Polynomial operations

- The sum of two polynomials is done by adding together the coefficients of the monomials sharing the same power. It results in a polynomial having the degree the maximum degree of the two polynomials.
- Similarly, the difference of two polynomials is done by subtracting the coefficients of the monomials sharing the same power. It results in a polynomial having the maximum degree the maximum of degree of the two.
- The product of two polynomials is computed by multiplying each term of the first polynomial by each term of the second polynomial, thus resulting in a polynomial of a degree that results in adding the degrees of both the polynomials.
- The derivative of a polynomial is simply computed by differentiating each term it consists of.
- The integral of a polynomial is computed by integrating each term the polynomial consists of.
- The division of two polynomials is done by following this algorithm:
    - Divide the first term of the dividend by the highest term of the divisor;
    - Multiply the divisor by the result just obtained;
    - Subtract the product just obtained from the appropriate terms of the original dividend. The result will be the new dividend;
    - Divide the highest degree term of the new dividend by the highest degree term of the divisor;
    - Multiply each term of the divisor by the result of the previous step and subtract it from the dividend. This will result in the remainder;
    - If the remainder has a degree lower than that of the divisor than the division cannot continue any further. If not, the algorithm is repeated form the 4 step onward;
    - This results in a quotient and a remainder and the division is done;

4. User Diagram





Polynomial Processing System

The format of the polynomial must look like this:
C1 P1, C2 P2, C3 P3... (Spaces are mandatory!)

First Polynomial:

Second Polynomial:

There must be correct data in both text fields for the program to work!

Solution:

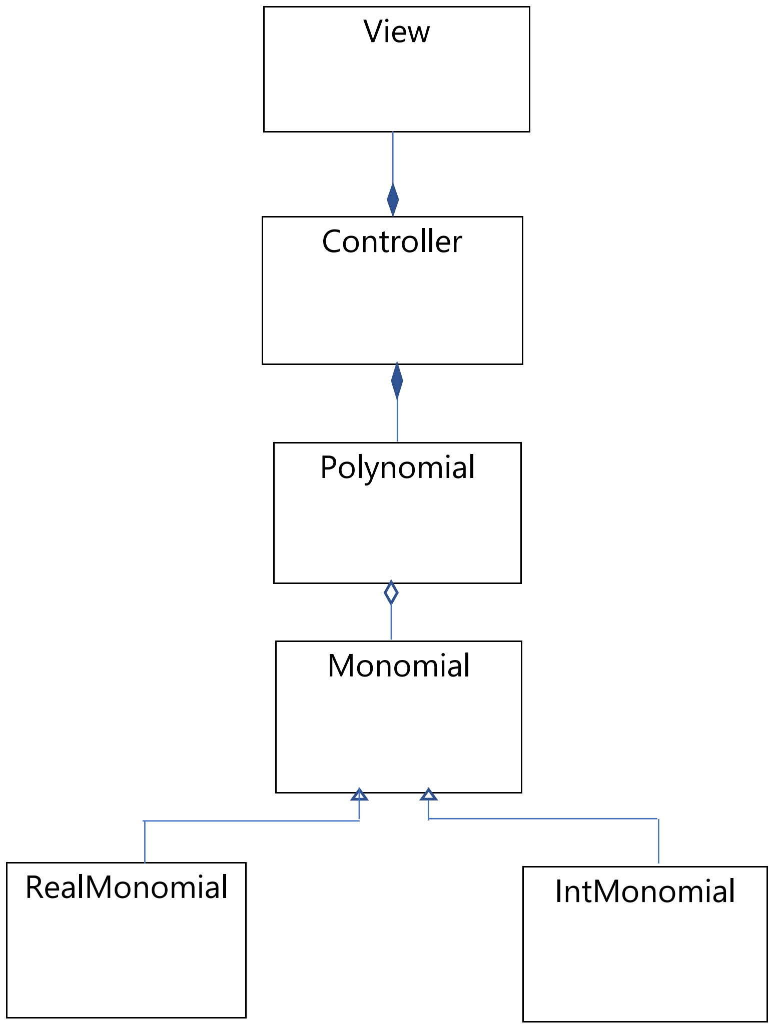
-

The user is presented with a simple and smooth graphical interface. It consists of two text fields in which the user will insert the two polynomials that the operation will be performed on, six buttons, each corresponding to an arithmetic operation and a label that will be updated to show the result of an operations. Other than these basic, imperative components, I also added numerous labels that will help the user understand how the program works and how the format of the input should be.

In my opinion, the user interface is pretty straight forward in how the user is supposed to use the system. Two polynomials are supposed to be written in the two text fields and upon pressing the button that corresponds to the desired operation, the result will appear under the "Solution:" label.

5. Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



Polynomial
(-) polynomial: List <Monomial>
(+) addition(Polynomial pol): Polynomial (+) subtraction(Polynomial pol): Polynomial (+) multiplication(Polynomial pol): Polynomial (+) division(Polynomial pol): Polynomial (+) differentiation(): Polynomial (+) integration(): Polynomial (+) toString(Polynomial pol): String (+) getPolynomial(): List <Monomial> (+) setPolynomial(List <Monomial>): void

Monomial
(-) coeff: Number (-) power: int
(+) Monomial(Number c, int p) (+) getCoeff(): Number (+) setCoeff(Number coeff): void (+) getPower(): int (+) setPower(int power): void

IntMonomial
(+) IntMonomial(Integer c, int p)

RealMonomial
(+) RealMonomial(Double c, int p)

View
(-) firstPol: JTextField (-) secondPol: JTextField (-) frame: JFrame (-) btn: JButton (-) btn: JButton (-) btn: JButton (-) btn: JButton (-) btn: JButton (-) btn: JButton (-) lblFinal: JLabel
(+) init(): void … getters and setters

Controller
(-) view: View (-) firstPol: Polynomial (-) secondPol: Polynomial
(+) Controller() (+) read(): void (+) initPols(String[] s, String[] s): void (+) addMonom(): void (+) addition(): void (+) subtraction(): void (+) multiplication(): void (+) division(): void (+) differentiation(): void (+) integration(): void (+) display(String pol): void … getters and setters

5.1. Description

I designed the problem based on the Model – Controller – View (MVC) architectural pattern. It is usually used for developing user interfaces that divides an application into three parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

- The model is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface. It directly manages the data, logic and rules of the application. Thus, my model consists of the Polynomial class and the Monomial class (with its children: IntMonomial and RealMonomial).

- The view consists of the Graphical User Interface (GUI) which will be described later in this document.

- The controller accepts input and converts it to commands for the model or view. So it basically links the model and the view.

I chose the MVC architectural pattern because it enables the logical grouping of related actions on a controller together, it creates components that are independent of one another, thus enabling the reusability of the code, it has low coupling meaning that its components has, or makes use of, little or no knowledge of the definitions of other separate components and it is good practice for future group projects I will be a part of because multiple developers will be able to work simultaneously on the mode, controller and views. Also, even though I have not used it for this project, the MVC architecture can have multiple views which can be very useful in a lot of projects.

Although it has many advantages, I noticed that the MVC has disadvantages too. The framework navigation can be complex because it introduces new layers of abstraction and requires users to adapt to the decomposition criteria. It also has a steep learning curve although I would consider this useful for developing my skills as a programmer.

6. Class Description

• View (GUI)

This class represents the GUI. The private attributes that also have getters and setters are the ones that are needed to perform the required task.

The init() method is a pretty straight forward window building method. It uses JTextFields, JButtons, JLabels and a JFrame.

- ➢ The input text fields expect to receive a string of coefficients and powers that would correspond to a polynomial. The order of the coefficient – power tuples can be random although the string must have the following format:
  C0 P0, C1 P1, C2 P2, C3 P3 …, CN PN
- ➢ If the format is not respected the program will throw an error and it will exit. An error message is displayed in the console.

The View class is contained in a package with the same name.

• Monomial

This class has two attributes: coeff (Number type) and power (int type). The coeff attribute is of Number type because we will be needing two kinds of monomials: real monomials and integer monomials. The Number type encapsulates both cases. This will be implemented by creating two subclasses that will be described after this.

This class' only methods are its constructor which assigns the two parameters to its attributes, and the getters and setters for coeff and power.

• IntMonomial

This class extends the monomial class meaning that it is a subclass. Thus, it only has a constructor with two parameters. In contrast with its superclass, here the coeff parameter is of Integer type.

• RealMonomial

Similarly to the IntMonomial class, this class extends the monomial class meaning that it is a subclass. Thus, it only has a constructor with two parameters. In contrast with its superclass, here the coeff parameter is of Double type. This makes it possible to execute the subtraction and integration operations. The result of those operations will be a polynomial that, instead of having integer coefficients, will have float (double) coefficients.

• Polynomial

The polynomial object is an ArrayList type that consists of an array of Monomials.

Methods:

- ➢ addition(Polynomial pol)
  - ~ this method detects the polynomial that has the maximum coefficient of the two operands (the parameter and the one that calls the method) and adds the coefficients of the other one to a copy of the first. This results in a third polynomial that is the sum of the other two and that will be returned.
- ➢ subtraction(Polynomial pol)
  - ~ similarly to addition(Polynomial pol), this method detects the polynomial that has the maximum coefficient of the two operands (the parameter and the one that calls the method) and subtracts the coefficients of the other one to a copy of

the first. This results in a third polynomial that is the sum of the other two and that will be returned.

- multiplication(Polynomial pol)
  - this method firstly calculates the degree of the result by adding the degree (maximum power) of the two operands. The resulting operand is then initiated with IntMonomials that have the coefficient 0. Finally, each of the first polynomial's monomials is multiplied with each of the second's and the resulting polynomial is updated and then returned.
- division(Polynomial pol)
  - this method is not yet implemented…

- differentiation()
  - firstly the resulting auxiliary polynomial is initiated with as many IntMonomials as the polynomial to be integrated -1. Then each monomial is updated: every ith monomial's coefficient will be the product of the ith original polynomial's monomial coefficient and power – 1 - i. The resulting polynomial is finally returned.

- Integration()
  - It is very similar to the differentiation() method. The degree of the resulting polynomial will be n + 1 (n being the original polynomial's degree) and every ith coefficient will be replace with the ith coefficient of the original devided by the power + 1. Also, the resulting polynomial will consist of RealMonomials because of the aforementioned division.

- ➢ toString(Polynomial pol)
  - ~ this is a simple method that generates a string from a given polynomial for it to be displayed. This method is implemented by using a StringBuilder. Thus, it is pretty straight forward, for each monomial the coefficient, a "x^", the power and a '+' will be appended to the StringBuilder. At the end every "+-" will be replaced with a '-' (this is because a '+' is added even if the next coeff is negative) and the last '+' is deleted. The resulting string is returned.

- ➢ Getter and setter for the Monomial list.
  - ~ Pretty straight forward.

These 4 classes is what the model package is made of.

The controller package has one single class:

• Controller

The controller accepts input and converts it to commands for the model or view. The class had as attributes two Polynomials 'firstPol' and 'secondPol' and a View. Thus, it is through the controller that the model and the view are linked. This class' constructor has calls the init() method of the view and all the operation methods.

- ➢ read()
  - ~ this method, as its name would suggest, takes the strings from the two text fields in the view and through it the two Polynomial attributes are updated with the correct data. It is surrounded with a try catch block in case the String in the text fields doesn't match the format required.

- initPols(String[] s1, String[] s2)
    - the two parameters represent two string arrays, each element being a coefficient and a power separated by a space. The method calculates the max power for each string and initializes the two Polynomial lists, resulting in the two being the size of the maximum power corresponding to their input string and each coefficient being 0.

- addMonom(Polynomial pol, String[] s)
    - a simple method that has as parameters a Polynomial and a string containing a coefficient (let's call it c) and a power (let's call it p). It sets the pth monomial's coefficient to c.

- the next 6 methods correspond to each of the 6 operations and function the same way. First the read() method is called to update the polynomial, then the operation method is called through the 'firstPol' and then the result is displayed using the display(String pol) method.
- display(String pol)
    - this method simply updates the result label from the view to display the result after the operation

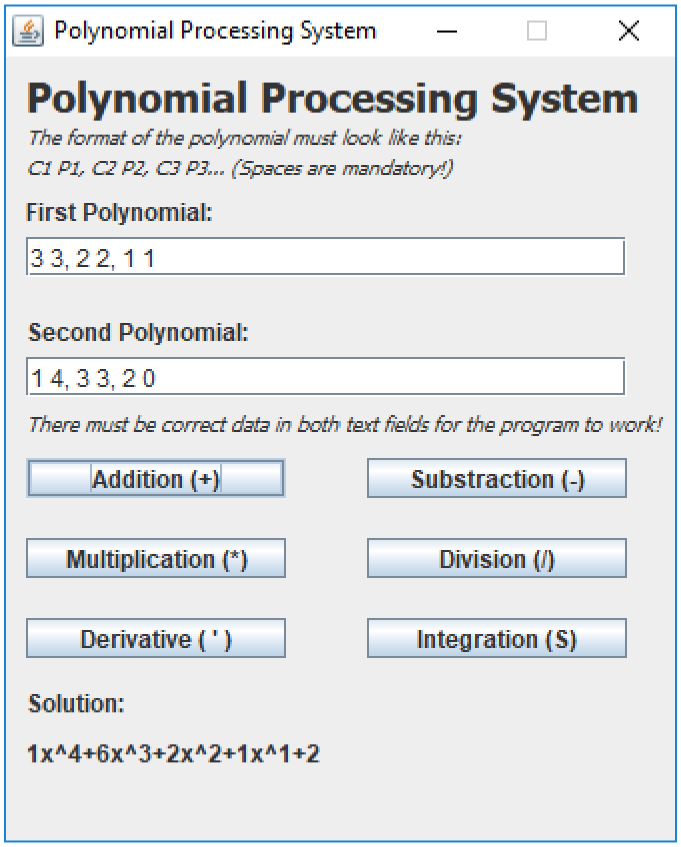
- getters and setters

7. Testing

I used JUnit to test all of the cases that would apply to this problem. As expected the tests came back positive meaning that the expected result for the given inputs is correct. The tests are under the Test class.

8. Results

Here are some examples of inputs and the outputs each of the operations.


Polynomial Processing System
Polynomial Processing System
The format of the polynomial must look like this:
C1 P1, C2 P2, C3 P3... (Spaces are mandatory!)
There must be correct data in both text fields for the program to work!
Addition (+)
Substraction (-)
Multiplication (*)
Division (/)
Derivative (')
Integration (S)

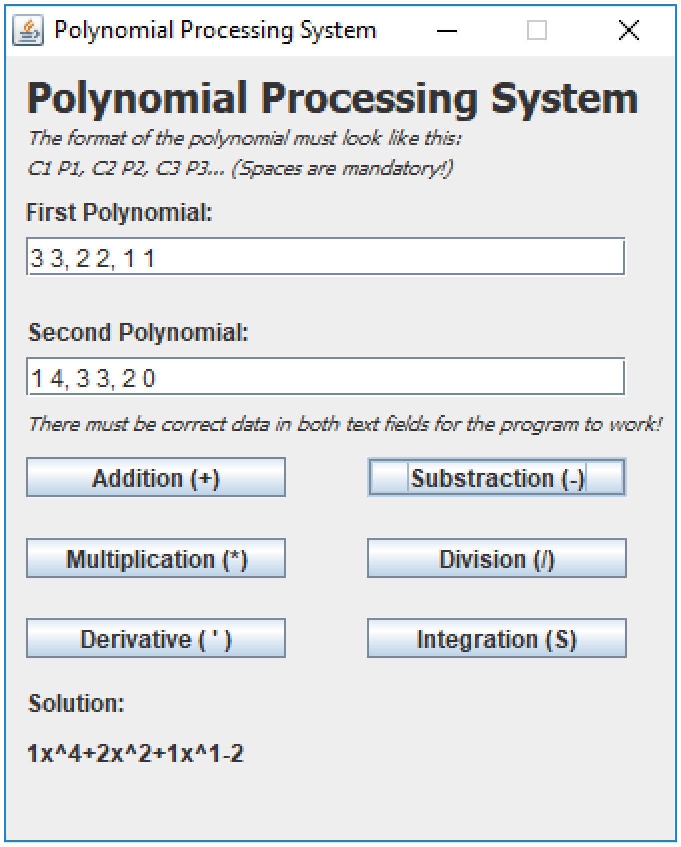
First Polynomial: 3 3, 2 2, 1 1

Second Polynomial: 1 4, 3 3, 2 0

Solution:

1x^4+6x^3+2x^2+1x^1+2

Addition


Polynomial Processing System
Polynomial Processing System
The format of the polynomial must look like this:
C1 P1, C2 P2, C3 P3... (Spaces are mandatory!)
There must be correct data in both text fields for the program to work!
Addition (+)
Substraction (-)
Multiplication (*)
Division (/)
Derivative (')
Integration (S)

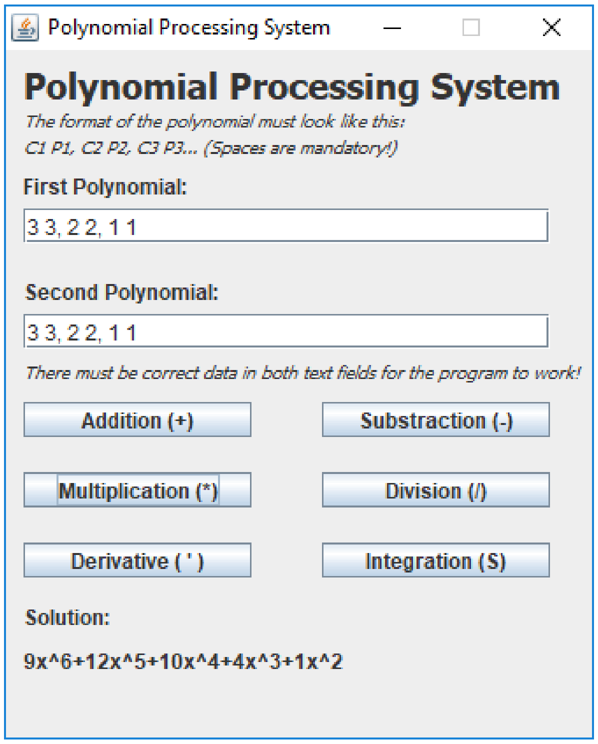
First Polynomial: 3 3, 2 2, 1 1

Second Polynomial: 1 4, 3 3, 2 0

Solution:

1x^4+2x^2+1x^1-2

Subtraction



Polynomial Processing System

The format of the polynomial must look like this:
C1 P1, C2 P2, C3 P3... (Spaces are mandatory!)

First Polynomial: 3 3, 2 2, 1 1

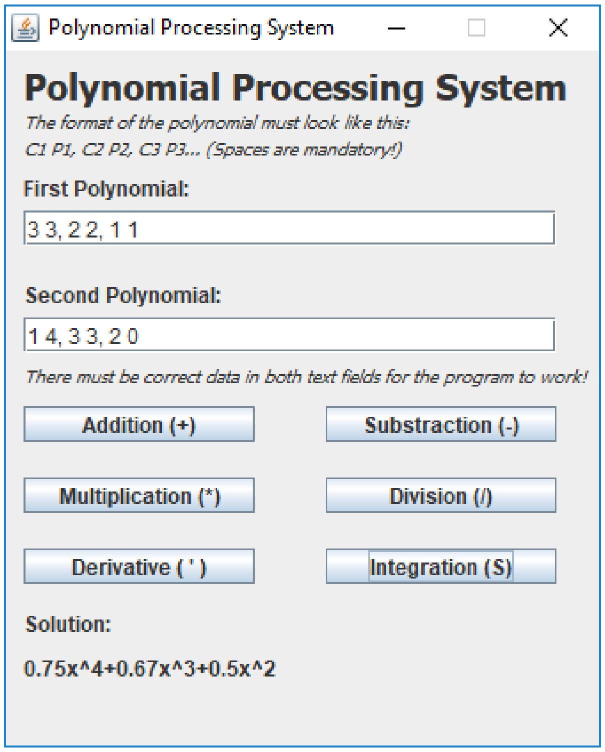
Second Polynomial: 3 3, 2 2, 1 1

There must be correct data in both text fields for the program to work!

Solution:

9x^6+12x^5+10x^4+4x^3+1x^2

<Multiplication



Polynomial Processing System

The format of the polynomial must look like this:
C1 P1, C2 P2, C3 P3... (Spaces are mandatory!)

First Polynomial: 3 3, 2 2, 1 1

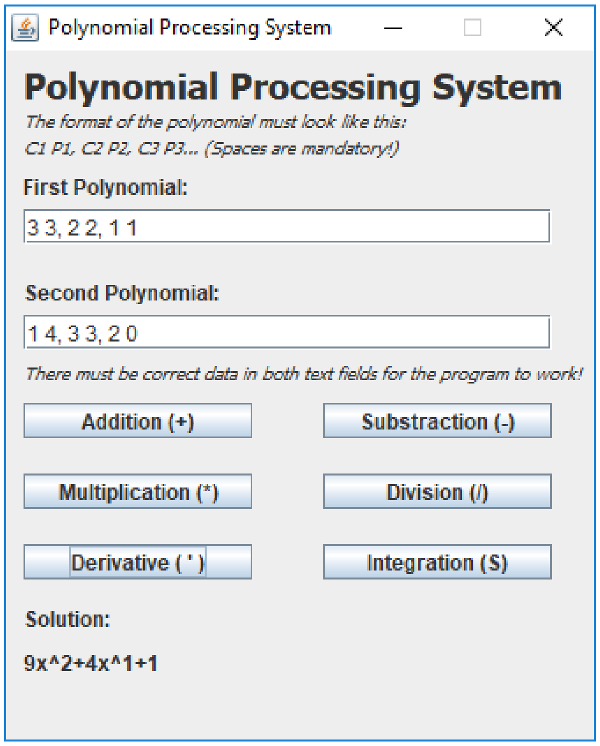
Second Polynomial: 1 4, 3 3, 2 0

There must be correct data in both text fields for the program to work!

Solution:

0.75x^4+0.67x^3+0.5x^2

Integration>



Polynomial Processing System

The format of the polynomial must look like this:
C1 P1, C2 P2, C3 P3... (Spaces are mandatory!)

First Polynomial: 3 3, 2 2, 1 1

Second Polynomial: 1 4, 3 3, 2 0

There must be correct data in both text fields for the program to work!

Solution:

9x^2+4x^1+1

<Differentiation

It can be observed that the results are correct, so the algorithms are correct although that doesn't mean that the implementation is optimal.

9. Conclusion

In my opinion, this first project was a very good choice to be the first one. It is very well designed for testing the OOP skills acquired in the first semester and to get us up to speed with the curriculum of this semester.

Further improvements to the program would be:

- ~ Have a read() method that takes as input any known format for the polynomials
- ~ Have the result of the integration to be displayed as a ratio
- ~ Have more operation (i.e. finding the roots)
- ~ Better implementation of the algorithms
- ~ Not displaying the ^1 for monomials with power 1

10. Bibliography

https://en.wikipedia.org

http://www.coned.utcluj.ro/~salomie/PT_Lic/

https://softwareengineering.stackexchange.com

https://www.google.ro