|  |  |
| --- | --- |
| **Project title** | **Polynomial Calculator** |
| **Author(s)** | **Pop Vlad-Iulian-Ioan** |
| **Group** | **30421** |

1. **Task Description**

The program’s goal is to implement a polynomial calculator which can perform different operations on them. The operations which can be performed are: adding, subtracting, dividing and multiplying two polynomial and also derivation or integration of one polynomial.

**Secondary requirements**

* Classes with maximum 300 lines (except the UI classes)
* Methods with maximum 30 lines
* Java naming conventions
* Object-oriented programming design (encapsulation, Polynomial and Monomial classes) • Lists instead of arrays
* foreach instead of for(int i=0…)
* Graphical user interface and pattern matching for extracting the polynomial coefficients
* Design according to the Model View Controller architectural pattern
* Regular expressions and pattern matching for extracting the polynomial coefficients
* Junit for testing

1. **Problem Analysis**

The input of the calculator must match a certain pattern, implemented with the help of the Regex package. The input for a monomial must have this form number\_x^number and each monomial must be separated by other through a withe space.

An example of a valid polynomial could be this one 3x^5 +4x^3 +2x^1.

Each monomial must obey this format implying that the user must specify each time the coefficient of the monomial and its grade including the time where the grade is equal to 0 or 1, for example x^0 or x^1, or the time where the coefficient is equal to 1, for example 1x^0 –this would mean that the free term is 1.

If the pattern isn’t respected a pop-up message will appear telling the user that the input is wrong and reminding the user the correct pattern that a polynomial must have.

For the operations to work the monomials must be written in decreasing order of their grade.

* Use Case: operations on two polynomials at a time
* Primary Actor: User
* Main Success Scenario:

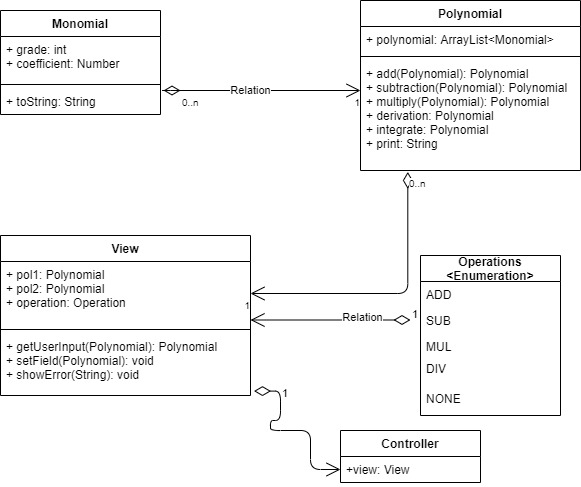
1. The user inserts a polynomial
2. The user selects an operation which requires two polynomials
3. The input is validated
4. The user inserts another polynomial
5. The user presses on the “=” button
6. The second polynomial is validated
7. The operation is computed
8. The result appears in the text box where the input has been inserted
9. The user press the “new” button which will erase the result and a new input can be inserted

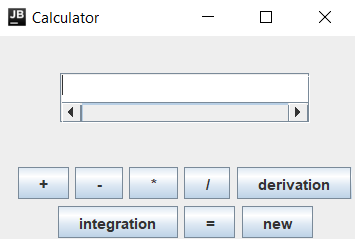
* Use Case: operations on one polynomial at a time
* Primary Actor: User
* Main Success Scenario:

1. The user inserts a polynomial
2. The user selects an operation which requires only one polynomial
3. The input is validated
4. The user presses on the “=” button
5. The operation is computed
6. The result appears in the text box where the input has been inserted
7. The user press the “new” button which will erase the result and a new input can be inserted

* Use Case: operations on one or two polynomials at a time
* Primary Actor: User
* Alternative Sequences:

1. The input does not respect the input pattern
2. The user inserts a polynomial
3. The user selects an operation
4. The input is not validated
5. The pop-up message appears
6. The use is requested another valid input
7. The user presses on the equal button “=” without an operation to be done
   * + 1. The pop-up message appears which will tell the user that no operation has been selected
       2. The user introduces a polynomial and selects an operation
8. The user inserts the monomials in non-decreasing order of their grade
   * + 1. The pop-up message appears which will tell the user that the monomial grades must be inserted in decreasing order
       2. The user inserts a valid polynomial

**3.Designing** ****

**This is how the interface looks like:**

The user inputs a valid input and then press the button that indicates which operation does he want to compute. After the pressing of one operation button the text area will be again blank making room for the second input. After the second polynomial press the “=” button to display the result on in the text area. There is a horizontal scroll bar for the cases that the input or the output is too large for the bounds of the given text area.

The button new can be press at any time to reset the input and the already chosen operation.

**4.Implementation**

**Monomial:**

The class represents a monomial. The coefficient and the grade fields are suppose to match a monomial of the form 2x^3. The field coefficient is of type Number because from the divison and from the integration operations the coefficient can result to be a double. So the purpose of using it is for flexibility, so we can have integer coefficients and double coefficient represented by the same field. The method toString is used for obtaining a good visual representation of the monomial.

**Polynomial:**

This class represent a list of Monomials which is commonly know as a polynomial.The list of Monomials is implemented with ArrayList.

The method of this class are meant to represent all the operations which have to be implemented in this calculator. All of them return the result of their specific operation in another object of type Polynomial.

The add and subtraction are implemented in a similar way. We do a simultaneous crossing of the two polynomials and if the grade of two monomials match we add them and then add the result to the new polynomial or if one the two grades is bigger we add it to the result. Such a crossing can be done because the monomials are inserted in decreasing way of their grades. After the two simultaneous crossings are done we check to see if there are any monomials of the two polynomials which haven’t been introduced in the result. The monomials which haven’t been introduced in the result are all the monomials of one of the two polynomials which have all their remaining polynomial grades smaller than the smalles monomial grade from the another polynomial.

The method multiply contains a for each for the first polynomial and another for each nested in the first one for the second polynomial multiplying each monomial of the first polynomial with each monomial of the second monomial.

The method derivation contains a for each and adds to the result each monomial with the original coefficient\* grade and the original grade-1.

The method integrate contains a for each and adds to the result each monomial with the coefficient, coefficient\*(grade+1) and the grade,grade+1.

The method print uses the method toString for each monomial of the polynomial.

**Operation:**

This class is an enumeration of the 5 five states that the View can have : NONE-no operation has been selected, ADD-add has been selected, SUB-subtraction has been selected, MUL-multiply has been selected, DIV- division has been selected.

**View:**

This class represents the view from the MCV pattern. It has as its fields the two polynomials that can be worked with and its state. All the graphical components are declared in this class.

The method getUserInput is use for retrieving data from the JTextArea and putting it in one of the two polynomials. The validation of the input is done in this method with the help of the Regex package.

The method setField is use for displaying the result of the operations to the user.

The method showError is use for the pop-up tabs which may appear if something goes wrong when the input is inserted.

Each listener of each button is declared in this class.

**Controller:**

This class represents the controller from the MVC pattern. The only field of this class is the view. In this class are implemented all the action listeners of each button.

The add, subtraction, multiply, derivation and integration buttons do almost the same thing. They fetch the input data with the getUserInput into the first polynomial named pol1, set the operation field of the view accordingly, and then put the empty string in the text area.

The equal button will fetch the input into the second polynomial and the will do the operation between the two polynomials that the operation field of the view suggests with a switch statement and the display the result in the text area.

The new button is used for convenience sake to erase the current data from the text area and to let the user insert a new input but it will also reset the operation field to NONE.

**5.Testing**

In the unit tests I wrote a test function for each of the operations one polynomials that I managed to implement namely : addition, subtraction, multiplication, derivation and integration. In each of the tests I used the same two polynomials constructed in the same functions that test a certain operation. I use Junit for the unit tests.

**6.Conclusions**

This assignment helped me get a little better at the Java programing language in general and it also helped at clarifying some object oriented programming paradigms such as: encapsulation, unit test, use cases, UML diagrams, types of relationships between classes etc.

It also made know to me some useful tools such as: the for each statements, the Regex package for pattern matching and the usefulness of lists instead of arrays.

**7.Bibliography**

The pdf. Support that the laboratory teacher provided to us.

Multiple sites from the World Wide Web.