Assignment 1 Documentation

Polynomial Calculator

Student: Moisa Oana Miruna

Group: 30422

Polynomial Calculator Documentation

1. Assignment objective

The objective of this assignment is to develop a Java application, namely a calculator for polynomials of one variable and integer coefficients. The secondary objectives of this project are: understanding the requirements and how the application is supposed to work, designing the calculator, implementing the calculator (both the actual functionality and the graphical user interface) and finally testing the application using unit tests. Each of these sub-objectives will be described in detail in the following paragraphs.

2. Problem analysis, modeling, scenarios, use cases

The functional requirements for this assignment are the following:

- The users should be allowed to insert polynomials.
- The users should be allowed to choose the operation they want.
- The polynomial calculator should be able to perform the following operations: addition, subtraction, multiplication, division, compute the derivative and the integral.
- The polynomial calculator should display the result.
- The users should be allowed to use the calculator as many times as they want and change the input polynomials.
- The application should handle bad inputs from the user.

The non-functional requirements of this problem are:

- The application should be easy to use, with a friendly interface.
- Except for the UI classes, they should contain at most 300 lines.
- The methods must be at most 30 lines long.
- The code should use lists instead of arrays and foreach instead of for.

Use cases:

- **Use Case 1:** add/subtract/multiply two polynomials

Primary Actor: user

Main Success Scenario:

-The user inserts the two polynomials in the two text fields from the graphical user interface.

-The user clicks on the “ADD”/ “SUBTRACT”/ “MULTIPLY” button

- The result is computed and displayed in the corresponding text field

Alternative Sequence 1: Incorrect polynomials

-The user inserts one or two incorrect polynomials

-The user is warned about the mistake

-The process starts again

Alternative Sequence 2: One/two polynomials missing

-The user lets at least one of the two text fields empty

-The user is warned about the mistake

-The process starts again

Use Case 2: divide two polynomials

Primary Actor: user

Main Success Scenario:

-The user inserts the two polynomials in the two text fields from the graphical user interface.

-The user clicks on the "DIVIDE" button

- The result is computed and displayed in the corresponding text field

Alternative Sequence: Incorrect polynomials

-The user inserts one or two incorrect polynomials

-The user is warned about the mistake

-The process starts again

Alternative Sequence 2: One/two polynomials missing

-The user lets at least one of the two text fields empty

-The user is warned about the mistake

-The process starts again

Alternative Sequence 3: Polynomial zero

-The user inserts a 0 polynomial

-The user is warned about the mistake

-The process starts again

- **Use Case 3:** differentiate/ integrate one polynomial

Primary Actor: user

Main Success Scenario:

-The user inserts the one polynomial in the first text field from the graphical user interface.

-The user clicks on the "DERIVATIVE"/"INTEGRATION" button

- The result is computed and displayed in the corresponding text field

Alternative Sequence: Incorrect polynomial

-The user inserts an incorrect polynomial

-The user is warned about the mistake

-The process starts again

Alternative Sequence 2: Missing polynomial

-The user lets the first text field empty

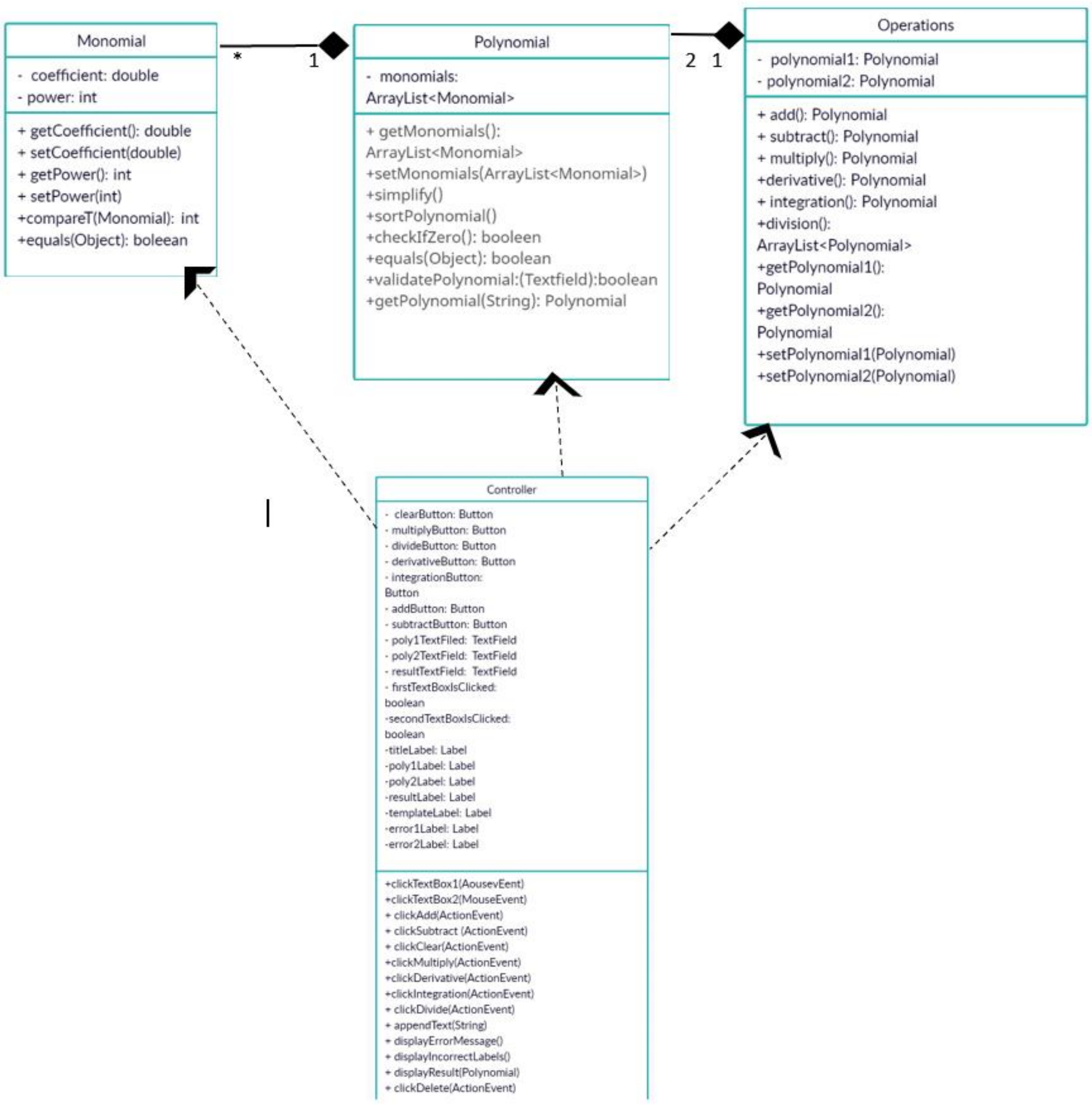
-The user is warned about the mistake

-The process starts again

3. **<u>Design (design decisions, UML diagrams, data structures, class design, interfaces, relationships, packages, algorithms, user interfaces)</u>**

Viewed as a black box, the application has 3 inputs (2 polynomials and 1 operation) and an output (the result).

I have designed my project using the Model View Controller pattern. I have created a package called "Models", which contains 3 classes used for the main functionality and data. These classes are: Polynomial, Monomial and Operations. The "base" class of the project is the Monomial, which has two fields: the coefficient and the power, because every polynomial is composed by multiple monomials. The class Polynomial has a single field which is a list of monomials. The class Operations has two polynomials as the fields and 6 methods which describe the functionality of each operation. I also have a View.fxml, which contains all the code related to the graphical user interface and a class Controller, which is an intermediate between the View and the Models. It is the class that takes data from the user interface and also sends data to the user interface, mainly by the means of events.



Monomial
- coefficient: double
- power: int
+ getCoefficient(): double
+ setCoefficient(double)
+ getPower(): int
+ setPower(int)
+compareT(Monomial): int
+equals(Object): boleean
*
1
Polynomial
- monomials: ArrayList<Monomial>
+ getMonomials(): ArrayList<Monomial>
+setMonomials(ArrayList<Monomial>)
+simplify()
+sortPolynomial()
+checkIfZero(): booleen
+equals(Object): boolean
+validatePolynomial:(Textfield):boolean
+getPolynomial(String): Polynomial
2
1
Operations
- polynomial1: Polynomial
- polynomial2: Polynomial
+ add(): Polynomial
+ subtract(): Polynomial
+ multiply(): Polynomial
+derivative(): Polynomial
+ integration(): Polynomial
+division(): ArrayList<Polynomial>
+getPolynomial1(): Polynomial
+getPolynomial2(): Polynomial
+setPolynomial1(Polynomial)
+setPolynomial2(Polynomial)
Controller
- clearButton: Button
- multiplyButton: Button
- divideButton: Button
- derivativeButton: Button
- integrationButton: Button
- addButton: Button
- subtractButton: Button
- poly1TextFiled: TextField
- poly2TextField: TextField
- resultTextField: TextField
- firstTextBoxIsClicked: boolean
-secondTextBoxIsClicked: boolean
-titleLabel: Label
-poly1Label: Label
-poly2Label: Label
-resultLabel: Label
-templateLabel: Label
-error1Label: Label
-error2Label: Label
+clickTextBox1(AousevEent)
+clickTextBox2(MouseEvent)
+ clickAdd(ActionEvent)
+ clickSubtract (ActionEvent)
+ clickClear(ActionEvent)
+clickMultiply(ActionEvent)
+clickDerivative(ActionEvent)
+clickIntegration(ActionEvent)
+ clickDivide(ActionEvent)
+ appendText(String)
+ displayErrorMessage()
+ displayIncorrectLabels()
+ displayResult(Polynomial)
+ clickDelete(ActionEvent)

As for the class relationships, between the Monomial Class and Polynomial class there is a relation of composition: one polynomial can have multiple monomials, the monomials are a part of the polynomial. Between the class Operations and the class Polynomial it is once again a relation of composition because an operation has two polynomials on which it operates. The class Controller is dependent on all these three classes, because it uses methods from all of them: Monomial, Polynomial, Operations. The Main Class is the one that launches the application.

The user interface is simple and easy to use. The two text fields for the operands are editable and must be used by the user. The result text field is used for displaying the result of the operations and is not editable by the user. The user can use the keyboard with both numbers and symbols in order to construct the polynomials. When the user clicks one such button, a character will be appended to the text field that was previously selected by clicking on it. I have used several labels in order to make the interface as detailed as possible. We have labels that indicate where the user should write the polynomials, where the user should look for the result and what the correct template for a polynomial is. These labels are visible at all times. I also have some labels that change visibility depending on the actions of the user. For example, the "Incorrect format" or "Required!" labels, that warn the user about the mistakes made.

4. Implementation

Monomial Class

It has 2 fields: a double coefficient and an integer power. For instantiating the monomial objects, I have used a constructor with these 2 parameters. The methods in this class are mainly getters and setters and 2 overridden methods. I have overridden the method compareTo(), which compares the powers of 2 monomials and which I have used for sorting the monomials of a polynomial. I have also overridden the equals() method, which I have used for testing.

Polynomial Class

It has only 1 field, which is an ArrayList of monomials. For this class, I have used a parameterless constructor. Besides the getters and setters, I have overridden the equals() method, also used for testing. Some other methods from this class are one that sorts the polynomial in descending order of the power of the monomials, one that checks if the polynomial is zero and one that removes the monomials with coefficient zero from the polynomial. Another 2 important static methods from this class are validatePolynomial(), which makes the application accept only polynomials that follow a certain pattern, and getPolynomial(), which transforms a String into an object of class Polynomial. Both these methods are based on regular expressions.

Operations Class

The operations class has two polynomials as fields and 6 methods for the 6 operations implemented: addition, subtraction, multiplication, division, derivation and integration.

add() method: This method returns the result polynomial and operates on the two fields of the Operations class. Firstly, I put the first polynomial into the result. Then, for each monomial from the second polynomial I search a "pair" in the result, so a monomial with the same power. If there exists one, the coefficient of the monomial in the result is set to the new value (the sum of the two coefficients). If the two coefficients sum to 0, the monomial is removed from the result. When the polynomials enter this method, I suppose they are ordered, so I can stop the search after I meet a smaller power.

subtract() method: Similar to add(), only that the coefficients are subtracted.

multiply() method: Multiply each monomial from the first polynomial with each monomial from the second polynomial and then add them to the result, so that the equal powers are simplified.

derivative() method: Returns a new polynomial composed of monomials with the coefficient equal to the old coefficient multiplied by the power and the power subtracted by one.

integrate() method: Returns a new polynomial composed of monomials with the coefficient equal to the old coefficient divided by the power+1 and the power is incremented by one.

divide() method: Returns an arraylist with the quotient and the remainder of the division. Always divide the polynomial with the higher grade to the polynomial with the smaller grade. I divide the first monomial of the dividend to the first monomial of the divisor, then I multiply the first element of the quotient with the divisor and subtract the result from the dividend, obtaining the remainder. This sequence is repeated until the degree of the divisor is greater than the degree of the remainder.

Controller Class

The user interface has 3 text fields for the polynomials, 2 of which can be edited by the user, 6 buttons for executing the corresponding operations, 16 buttons with numbers and symbols to allow the user to write in the text fields, one button for deleting from the text fields (one character at a time) and one Clear button for making all 3 text fields empty. The controller class is mainly composed of events for the buttons. All numbers and symbols buttons append text in the text field that is selected at that moment of time (which is decided with 2 boolean variables). The methods for the operation buttons first verify if any of the required text fields are empty, and if so, a message is displayed. Then, it is checked if the polynomials are valid, syntax wise. If not, another alert is displayed to the user. Then the polynomials are converted from strings into objects of type Polynomial and the corresponding method from the Operations class is called for them.

Regular Expressions & Inputs Accepted: I have based my implementation a lot on regular expressions. Firstly, I have used one for validating the input. The ideal structure of a polynomial inserted by the user is this: $P(\mathrm{x})=a*\mathrm{x}^\wedge n+\mathrm{b}*\mathrm{x}^\wedge(n-1)+\ldots+\mathrm{c}*\mathrm{x}+$ d, or without "*". If the user inserts a polynomial whose monomials are not sorted in descending order of the power, then the input is still valid, because it is automatically sorted in the method. It also works if the user decides to write something like this, which is odd and not necessary but accepted: "0*x", "1*x","x^0", "x^1". Any variation aside from the template and these exceptions are not accepted and the user will have to correct the mistakes (eg. "xx", "x++3", "x-","x^^2").

The regular expression that I have used for this validation is: "^(([+-]{1}|^[+-]?)([0-9]+[*]?[x]{1}|[0-9]+|[x]{1})([\\^]{1}[0-9]+)?)++$". This is the template for a monomial, which has to be valid throughout the whole polynomial. The signs +- must appear once, unless if it is the start of the expression, when they are optional. They are followed by 3 possibilities: coefficient, just the variable x, or both coefficient and x. Then the power is optional. For parsing the input I used a simpler expression in which I have defined three groups: one for extracting the coefficient, one for extracting the variable and one for extracting the power: "([+-]?[0-9]*)[*]?([x])?[\\^]?([0-9]*)". Then I have also added some additional code for correctly creating the polynomial using these groups.

5. Results

I have performed Parametrized Junit tests for all methods in the Operations class, in order to run a test multiple times with different parameters. I have provided String parameters for the input polynomials and the expected result, which then I have converted into Polynomials using the getPolynomial() method. For each operation I have done 4-5 tests, which covered both basic examples and the trickier or particular cases.

6. Conclusions

In conclusion, I think I learned a lot from this assignment, as it was both challenging and interesting to work on it. It was the first time that I've used an architectural pattern in my project and that has made me a lot more organized. Also, it was the first time that I have used unit tests and it made me realize the importance of testing in such a project. As for further developments, I think one would be to consider polynomials with negative powers also, maybe define an operation for definite integration, or even consider more input polynomials.

Application User Manual:

This is a calculator for executing operations on polynomials of integer coefficients and one variable. For the addition, subtraction, multiplication and division operations insert the polynomials in the first two text fields, push the button for the desired operation and the result will appear in the "result" text field. For the derivation and integration operations, insert only one polynomial in the first text field, then push the button for one of these two operations and the result will automatically appear. If you leave the required text fields empty, you will be warned about this, and you won't be able to see a result until you fill them in. For inserting the polynomials, you can use the numbers, symbols and variable x buttons available, just select the text field you want to write into using the mouse. If you want to delete something from a text field, use the Delete button, which deletes one character at a time. If you want to start all over again, press the Clear button, which will clear all three text fields. If the polynomials are not inserted according to the template showed on the interface, you will also be warned and you will have to correct the mistakes. Insert the polynomial sorted in descending order of the powers and respect the format presented.

7. Bibliography

ASSIGNMENT_1_SUPPORT_PRESENTATION_v1.pptx