**Programming Techniques**

**1st Homework**

**-Polynomial Processor-**

**1.Objectives**

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

This program is an application for polynomial processing .It consists of a GUI (user interface ) for an easier interaction with the application which performs different mathematical operations on two input polynomials. The operations are: Addition, Subtraction, Multiplication, Division , Integration and Differentiation. When the user wants to do a certain operation , he must input the respective polynomials in the indicated text fields and select the desired operation from the buttons on the window that will appear when he starts the application. The result will be displayed in a text field at the bottom of the window.

**2. Analysis, Modelling, Use Cases, Scenarios**

2.1. Analysis

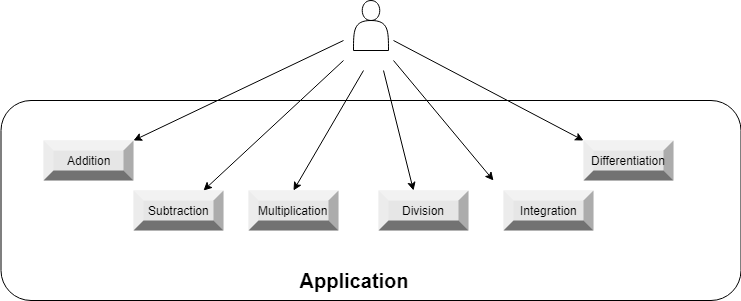
The application provides a comfortable environment for any kind of user which needs an efficient and fast way to compute some complex operations on polynomials which would otherwise take a significant amount of time to be done with pen and paper. The code for this application has been written in such a manner that each bit of code can be reutilised for other applications and new features can be easily added by other programmers since many of the actions have their own stand-alone class.

2.2. Modelling

Internally, the modelling of the polynomials relies on the main two classes: Polynomial and Monomial . A monomial consists of a coefficient ,a variable ‘x’ with unknowm value and an exponent and looks like this : “2x^1”. The polynomial is made up of two or more monomials with different exponents. Each of the two structures have their own functionality ,thus, they have their own class. Because each operation on the polynomial is done ultimately at the level of the monomial, Monomial objects have the purpose of supporting each basic arithmetical operations(such as addition, subtraction, multiplication,division) which is needed to compute more complex operations suffered by the polynomials. The polynomials are a collection of monomials so they have the function to keep themselves ordered in a descending order of the exponents and to be able to add new monomials to the list . The monomials are the backbone of the model.

2.3. Use cases

As per the use case definition, the control lies in the hands of the user once the application starts. The application has been created with the user’s experience in mind and provides a little guide for using every ability of the application correctly. The next diagram shows every function which the user is provided with:



2.4. Scenarios

Scenarios represent the tracking of steps taken by the user when uses the application. Their purpose is to predict any unwanted situation which can be caused by the user and to prevent them. Next, due to the similar behavior of all use cases, we’ll consider only three of them and step through the sequences.

*Use case #1-* Addition

Input : two polynomials

Output : sum of the two polynomials

Sequence of events :

1. User starts application
2. The user writes the two input polynomials in the “First polynomial” and “Second polynomial” fields
3. The user clicks on the button “Add” to trigger the desired operation to be performed
4. The system receives input and begins validation process
5. If the input is correctly introduced, the result will appear in the “Result” text field.
6. Else, a notification will pop up, signaling the error in input that has been found or if no input has been found.
7. Application reverts to step 2.

*Use case #2-* Division

Input : two polynomials

Output : the quotient and the remainder from the division and both come in polynomial form

Sequence of events :

1. User starts application
2. The user writes the two input polynomials in the “First polynomial” and “Second polynomial” fields
3. The user clicks on the button “Divide” to trigger the desired operation to be performed
4. The system receives input and begins validation process
5. If the input is correctly introduced, the result will appear in the “Result” text field and “Division remainder”
6. Else, a notification will pop up, signaling the error in input that has been found or if there was no input found.
7. Application reverts to step 2.

*Use case #3-* Differentiation

Input : one polynomial

Output : the derivative of the polynomial

Sequence of events :

1. User starts application
2. The user writes the input polynomial in the “First polynomial” or “Second polynomial” field.
3. The user clicks on the button “Derive” to the side of the field where is the input polynomial to be differentiated
4. The system receives input and begins validation process
5. If the input is correctly introduced, the result will appear in the “Result” text field.
6. Else, a notification will pop up, signaling the error in input that has been found.
7. Application reverts to step 2.

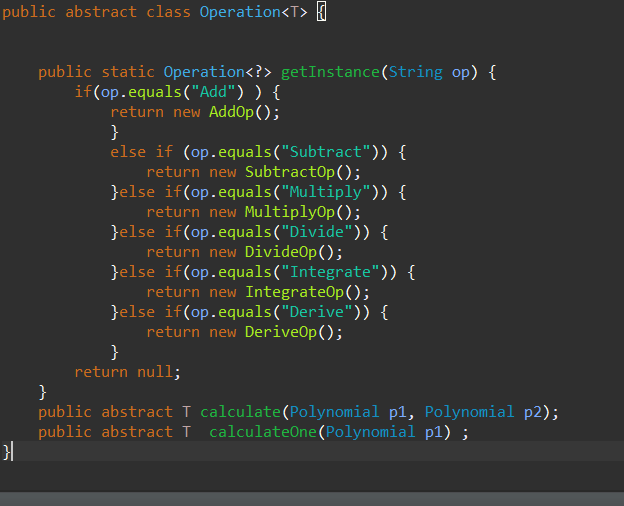
**3. Design**

3.1. Design decisions

Despite the low complexity of the application ,I decided to use the high-level architectural pattern MVC and the lower-level design pattern named Factory Pattern. This decision is based on my desire to learn everything about OOP efficient and organized programming. With the Factory Pattern I have fully employed the power of polymorphism creating a “virtual” constructor out of an abstract class where we delegate the instantiation of objects to subclasses and where we put the definition of factory methods. These are static methods overridden by each subclass. The client is totally decoupled from the implementation details of derived classes. Polymorphic creation is now possible.

The “Operation” class is the abstract factory class whose task is to fetch the instance

of the operation which is needed by the constructor and to provide the declaration of the header of the methods that will be overridden and implemented by the subclasses.

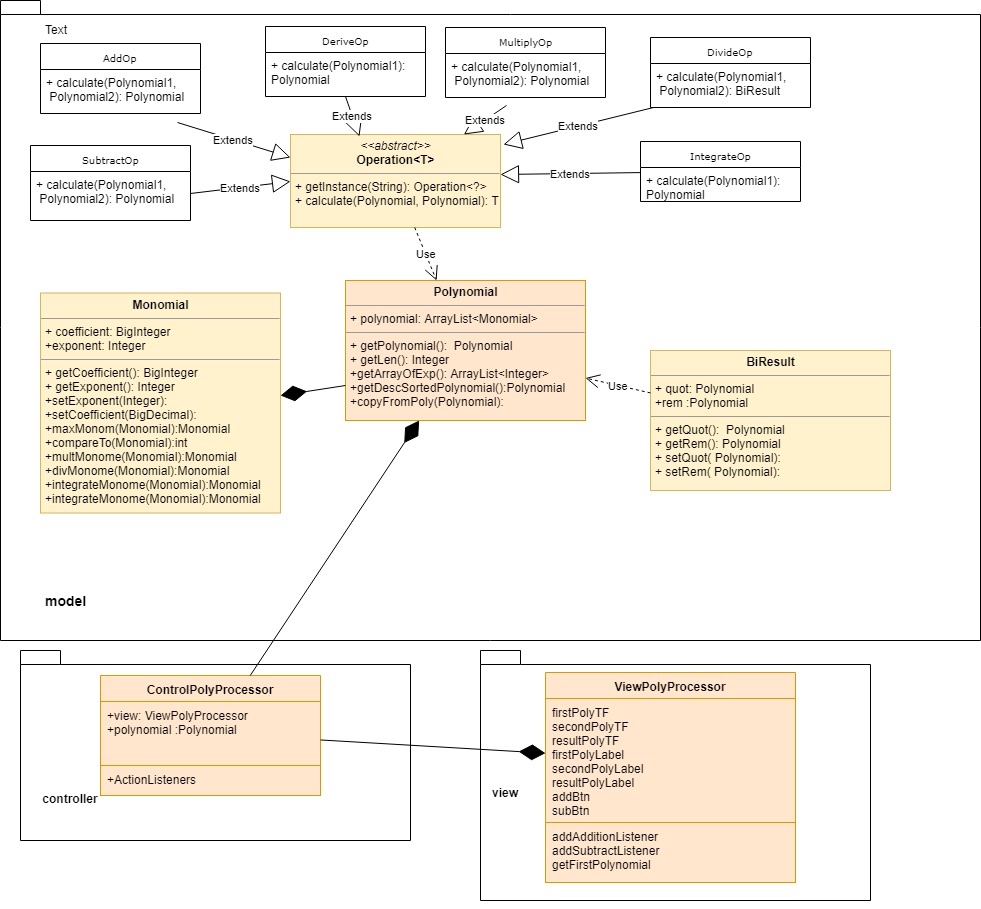


The MVC architectural pattern stands for Model-View-Controller and splits the application in 3 main parts, each with its own role.

Model 🡪 contains the representation of data in application .In this application is ensured by the Polynomial, Monomial , BiResult , Operation and all of its’ subclasses(AddOp, SubtractOp, MulitplyOp etc.). The BiResult is used only when we return the result from the division under the form of quotient and remainder, therefore it will return an object with two polynomials.

View 🡪 contains the implementation for the GUI. It’s used only for designing the interface.

Controller 🡪 is the one that connects the view and the model. It listens to the users’ commands from the interface and fetches the results and data needed to satisfy those commands.

3.2. UML Class Diagram

3.3. Data Structures

***List🡪*** is used through the implementation form of ArrayList which is a resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)

3.4. Class Design

The following are the classes used in the application:

* Monomial – is the backbone of the model, being the computational cell off the application. Every basic arithmetical function takes place here at an atomic level.
* Polynomial – used to instantiate polynomials objects which are stored in an ArrayList of Monomials.Objects of this class can sort themselves in a descending order and ,also, copy themselves into a new polynomial.
* BiResult –especially made for division because it returns an object composed of the quotient and remainder of the division
* Operation
* AddOp, SubtractOp, DivideOp,MultiplyOp, IntegrateOp, DeriveOp
* ControllerPolyProcessor
* ViewPolyProcessor
* Extractor –is the parser of the polynomials . It breaks down string inputs and extracts from it the coefficients and exponents from input.

3.5. Relationships

Due to the architectural pattern the main classes ,that represent the core of the application, are ControllerPolyProcessor, ViewPolyProcessor, Polynomial and Monomial. The controller listens for the operation requested by the user through the interface and uses the Operation class to compute and retrieve the result. Operation class fetches an instance of the operation classes AddOp, SubtractOp, DivideOp, IntegrateOp, MultiplyOp, DeriveOp. Each of these classes use objects of the Polynomial, Monomial and BiResult classes.

3.6. Packages

The application is structured in controller , model, view and exceptions packages.The model package is the biggest as it contains the most classes as in the Polynomial and Monomial, all operation classes. The controller contains ControllerPolyProcessor and the view contains ViewPolyProcessor. The exceptions package contains the class WrongFormatException class which are thrown by the Extractor class, contained in the model.

3.7. Algorithms

Addition : adding two polynomials is done by adding the coefficients of the monomials with the same exponent from the two polynomials. If there are monomials who doesn’t have a corresponding exponent in the other polynomial, it will simply be added in the resulting polynomial.

Subtraction: for subtraction the same steps are applied as in the addition, the only difference is that you subtract the coefficients from the second polynomial’s monomials with the same exponent and the resulting polynomial will contain the monomials which have not been subtract with a ‘-‘ sign.

Multiplication: has been done after the mathematical algorithm that multiplies each monomial from the first monomial with all of the monomials in the second polynomial.The result is added into a separate polynomial.

Division: the algorithm used is called Long Division and follows the next steps:

1. I make sure the polynomials are written in descending order.
2. Divide the term with the highest power in the divisor polynomial by the term with the highest power in the dividend polynomial. The result will be placed in the quot polynomial.
3. Multiply the quotient obtained in the previous step by the dividend polynomial and set it as the remainder.
4. Subtract the remainder from the divisor.
5. Repeat Steps 2, 3, and 4 until the exponent of the leading term in the remainder is less than the exponent of the leading term of the divisor polynomial.

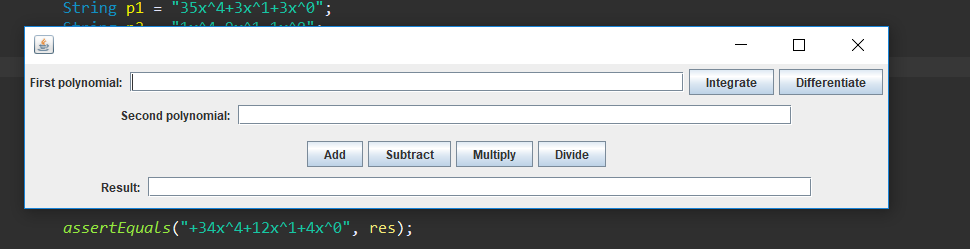
Integration: each monomial has been integrated after the formula:

IntegralOf(ax^n) = a/(n+1)x^(n+1)

Derivation: each monomial has been differentiated after the formula:

(a\*x^b)’ = a\*b\*x^(b-1)

3.8. User Interface



The user interface contains the fields for introducing the two polynomials to be computed. Due to design choices ,only the first polynomial introduced can be integrated and differentiated.

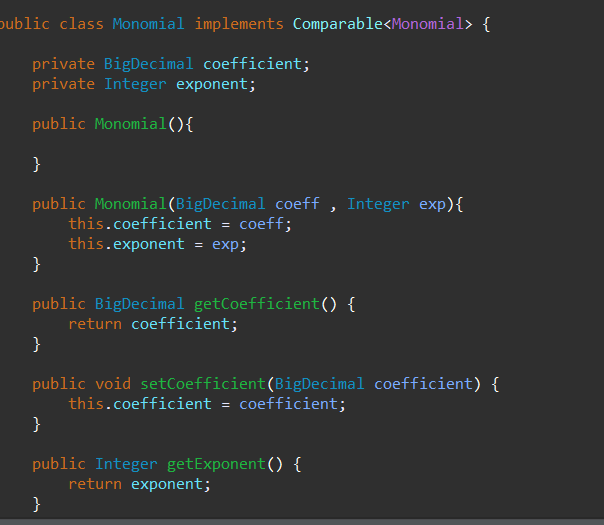
When the user makes a click on one of the buttons, after he input the polynomials, which has written on it the operation it performs ,the program will perform the arithmetical operation and display it in the result field.

One mention is that for the division the quotient and remainder will be displayed in the result field as follows : “quot : 1x^0 rem: “ in the case we have no remainder.

**4.Implementation and testing**

4.1 Implementation

Next are the description of implementation of some of the most important classes.

**Monomial class** – being the atoms of polynomials they contain coefficient as BigDecimal instances due to the need for them to accurately represent the coefficients of the polynomials result after division or integration and BigDecimal has very great precision of representation. This class has methods implemented for the operations performed on polynomials at monomial level.

**Polynomial class**- it contains an ArrayList of Monomials as its only field and has a constructor with empty parameters and another one with ArrayList parameters.

**Extractor class**- it contains the regex expression which I used to parse the polynomials: "(-? [\\b](file:///\\b) \\d+) x \\ ^( -?\\d+ [\\b)](file:///\\b))".This expression returns groups of characters which represent the coefficient and exponent of each polynomial and ads the newly formed monomials into the polynomial we need further.

4.2. Testing

**5.Results**

The resulting application is a robust and user-friendly „helper” for polynomial computations. The GUI is simple and intuitive, the buttons are pretty self-explanatory. The user is not required to write in the text fields the polynomials in the correct order, as the program does the ordering, but the user must respect a certain format for inputting polynomials. For example, „+1x^3-2x^1+1x^0” cannot be written as „x^3+1-1x-1x”, every coefficient and exponent must be clearly written out. The program ignores garbage input and selects it out of the input polynomial. The only allowed charcters are digits, „+-^” and „x”, all the others are considered garbage. Also, the only limit on the number of monomials is the memory allocated to the ArrayLists, so no hard-coded limit is involved.

**6.Conclusions, Lessons learned**

6.1. Conclusions

In building this program the two main difficulties were the mathematical implementation and the user interaction, and I discovered that both can be error prone due to the unexpected situation that can occur during the usage of such a complex application.can .I had to make a lot of debugging in order to provide a robust implementation.

6.2. Lessons learned

During the developement of this application I have learned how how to implement the Polynomial Long Division algorithm, also how regular expressions work and how useful they are for parsing Strings and how to take advantage of the functionalities provided by the ArrayList and BigDecimal class .

**7.Bibliography**

<https://codereview.stackexchange.com/questions/64250/showing-a-jdialog-whenever-there-is-an-exception>

<http://www.mesacc.edu/~scotz47781/mat120/notes/divide_poly/long_division/long_division.html>

<https://stackoverflow.com/questions/28859919/java-regex-separate-degree-coeff-of-polynomial>

<https://stackoverflow.com/questions/2784514/sort-arraylist-of-custom-objects-by-property>