Technical University of Cluj-Napoca

Faculty of Automation and Computer Science

2nd Semester 2016-2017

Programming Techniques Homework 1

Student: Andrei Branga

Group: 30422

Table of contents

1. Assignment objectives

(EN) Lab – Homework 1

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

2. Problem analysis, modelling, scenarios, use cases

2.1. Problem analysis

A system that takes as input two polynomials and with the aid of an intuitive GUI gives the user the possibility to apply specific operations on the input data.

I chose to give the user the possibility to enter the data as an input string.

The program will execute different operations on the data:

- Addition
- Multiplication
- Division
- Subtraction
- Derivative
- Integral

2.2. Modelling

For abstractisation I chose to implement the primordial data structure (The polynomial itself) as follows:

Firstly, a set of monoms is defined. Based on it, a special method (PolynomialFactory) will generate and instantiate the desired polynomial.

There are two types of polynomials and monoms:

- Real
- Integer

By our homework convention, the input polynomials are always integer, though they can also be processed as real.

2.3. Scenarios, use cases

The application I designed allows the user to enter two polynomials (p1, p2) and then choose the desired operation to be executed.

For example, if the user enters the following polynomials:

Polynomial_1 = 2*x^3+3*x^2-x+5   and

Polynomial_2 = x^2-x+1

the operations will result as follows:

Addition: **2*x^3+4*x^2-2*x^1+6*x^0**

Subtraction: **2*x^3+2*x^2+4*x^0**

Multiplication: **2*x^5+1*x^4-2*x^3+9*x^2-6*x^1+5*x^0**

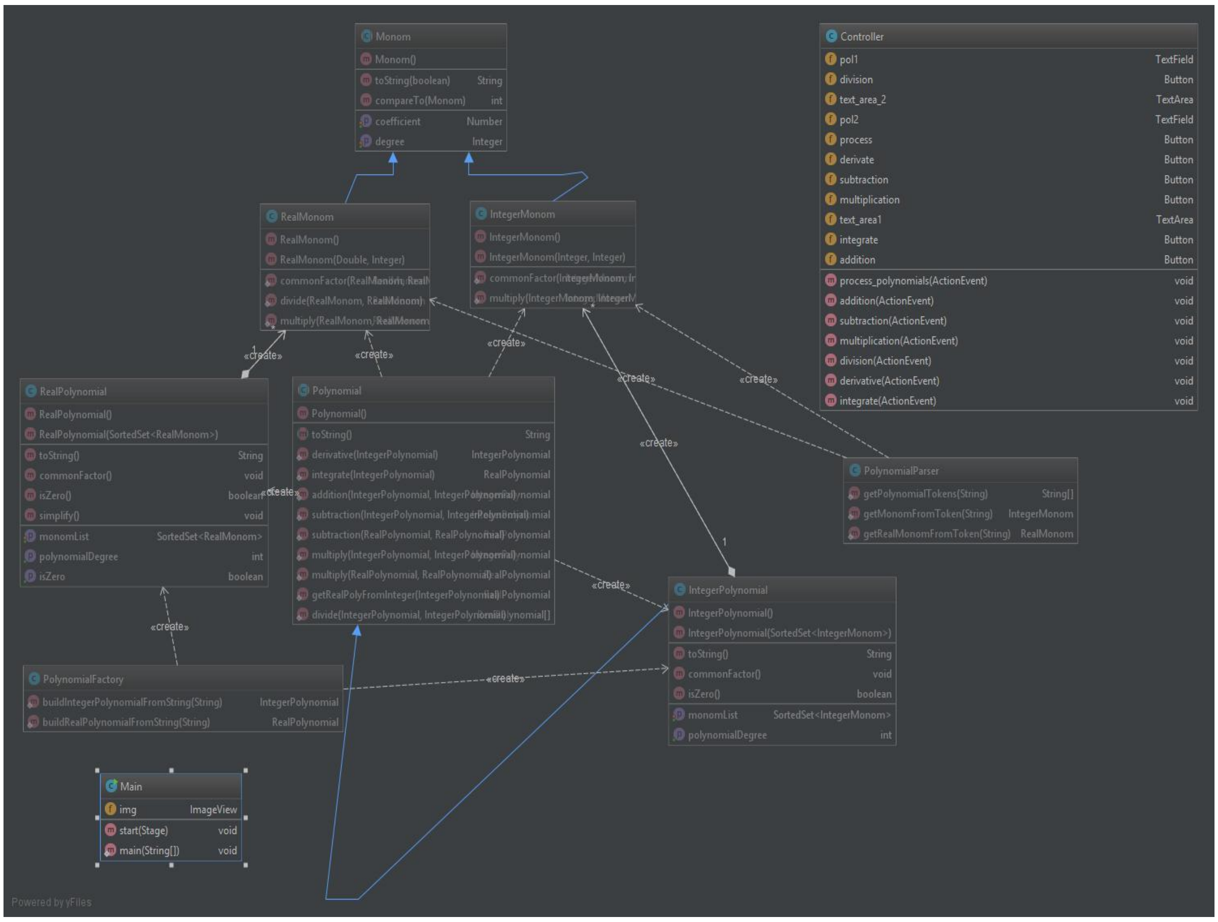
Derivative: **6*x^2+6*x^1-1*x^0 and 2*x^1-1*x^0**

Division): **2.0*x^1+5.0*x^0 ... rest: 2.0*x^1**

Integral: **0.5*x^4+1.0*x^3-0.5*x^2+5.0*x^1 and 0.3*x^3-0.5*x^2+1.0*x^1**

3. Design

3.1. UML diagram


Monom
Monom()
toString(boolean) String
compareTo(Monom) int
coefficient Number
degree Integer
Controller
pol1 TextField
division Button
text_area_2 TextArea
pol2 TextField
process Button
derivate Button
subtraction Button
multiplication Button
text_area1 TextArea
integrate Button
addition Button
process_polynomials(ActionEvent) void
addition(ActionEvent) void
subtraction(ActionEvent) void
multiplication(ActionEvent) void
division(ActionEvent) void
derivative(ActionEvent) void
integrate(ActionEvent) void
RealMonom
RealMonom()
RealMonom(Double, Integer)
commonFactor(RealMonom, RealMonom)
divide(RealMonom, RealMonom)
multiply(RealMonom, RealMonom)
IntegerMonom
IntegerMonom()
IntegerMonom(Integer, Integer)
commonFactor(IntegerMonom, IntegerMonom)
multiply(IntegerMonom, IntegerMonom)
«create»
RealPolynomial
RealPolynomial()
RealPolynomial(SortedSet<RealMonom>)
toString() String
commonFactor() void
isZero() boolean
simplify() void
monomList SortedSet<RealMonom>
polynomialDegree int
isZero boolean
Polynomial
Polynomial()
toString() String
derivative(IntegerPolynomial) IntegerPolynomial
integrate(IntegerPolynomial) RealPolynomial
addition(IntegerPolynomial, IntegerPolynomial) IntegerPolynomial
subtraction(IntegerPolynomial, IntegerPolynomial) IntegerPolynomial
subtraction(RealPolynomial, RealPolynomial) RealPolynomial
multiply(IntegerPolynomial, IntegerPolynomial) IntegerPolynomial
multiply(RealPolynomial, RealPolynomial) RealPolynomial
getRealPolyFromInteger(IntegerPolynomial) RealPolynomial
divide(IntegerPolynomial, IntegerPolynomial) RealPolynomial[]
PolynomialParser
getPolynomialTokens(String) String[]
getMonomFromToken(String) IntegerMonom
getRealMonomFromToken(String) RealMonom
1
IntegerPolynomial
IntegerPolynomial()
IntegerPolynomial(SortedSet<IntegerMonom>)
toString() String
commonFactor() void
isZero() boolean
monomList SortedSet<IntegerMonom>
polynomialDegree int
PolynomialFactory
buildIntegerPolynomialFromString(String) IntegerPolynomial
buildRealPolynomialFromString(String) RealPolynomial
Main
img ImageView
start(Stage) void
main(String[]) void
Powered by yFiles

3.2. Class design

As you can see in the UML Diagram, I chose to implement the polynomial using Polynomial class with its children and Monom class with its children.

PolynomialFactory will deal with constructing the polynomials from user's input.

PolynomialParser will parse the input string into data used by PolynomialFactory.

The Frame Class

The frame class hosts the Controller and Main classes.

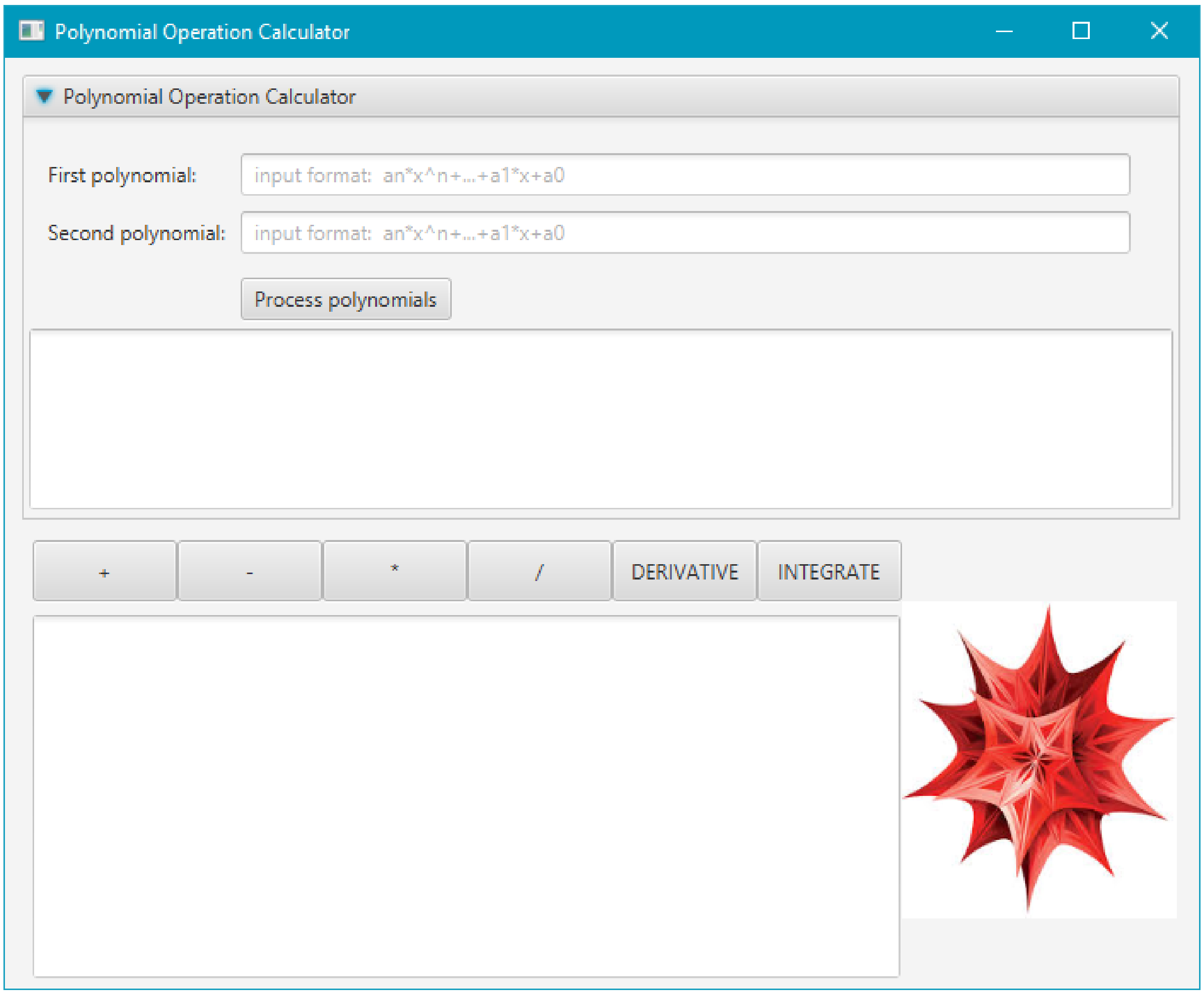
For the GUI I chose to implement it using javafx instead of swing. So, I created an fxml file holding the Frame design. In order for the GUI to work as intended, it needs a controller. This is what the Controller class does. It manages the GUI actions. The main class launches the frame and the Controller

3.3. Graphical User Interface (GUI)

The graphical user interface is designed to be easy to use, even for non-specialists. When running the application, a new window will open, where two polynomials will be written and one of the operations has to be chosen.

The input format is suggested inside the input fields.

It is very important to respect it!


Polynomial Operation Calculator
Polynomial Operation Calculator
First polynomial:
input format: an*x^n+...+a1*x+a0
Second polynomial:
input format: an*x^n+...+a1*x+a0
Process polynomials
+
-
*
/
DERIVATIVE
INTEGRATE

After choosing the desired operation, the input polynomial(s) along with the computed result will be shown in the lower part of the application window, in an easy to read format.

4. Conclusions

After this application's development I have figured out the following:

- Every task is simpler if you break it into smaller ones
- Every part needs to be tested before joining the final configuration
- Some difficulties may appear which may seem easy to deal with but are not easy at all (ex: input string parsing)

4.1. Things I learned

I am very happy that I have learned the polynomial division algorithm.

I have learned to work in the OOP-correct way.

I have learned to use javafx and SceneBuilder for developing better GUIs.

4.2. Future developments

There are lots of things that can be added to the application in the future, to improve the functionality, and also the graphical user interface.

Possible functionality improvements:

- catch all possible exceptions, and display a corresponding message (alert dialog, or result section).
- Allow more input formats so it would be easier for the user to enter the polynomials

5. Bibliography

http://stackoverflow.com/