DOCUMENTATION

ASSIGNMENT *No1*

STUDENT NAME: Terente Ioana-Emilia
GROUP: 30422

CONTENTS

1. Assignment Objective

The main objective of the project is to design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select a mathematical operation to be performed and view the result.

The sub-objectives of the project are to:

- Analyze the problem and identify the requirements
- Design the polynomial calculator
- Implement the polynomial calculator
- Test the polynomial calculator

2. Problem Analysis, Modeling, Scenarios, Use Cases

Problem Analysis:

A polynomial is an expression that can be built from constants and symbols called indeterminates or variables by means of addition, multiplication and exponentiation to a non-negative integer power.

The polynomial consists of a list of certain terms, also called monomials, for example -3x^4 is a monomial. The coefficient is -3, the indeterminate is x and the degree is 4. Forming a sum of several terms produces a polynomial, for example: -3x^4+5x-2, which has three terms with different exponents: the first is degree four, the second is degree one and the third is degree zero.

An alternative representation for polynomials consists of a sequence of order pairs:

{ a0,0, a1,1,a2,2, …., (an,n)}.

Each order pair (ai,i) corresponds to the term ai x^i of the polynomial. An ordered pair is composed of the coefficient of the i-th term and its index representing the exponent i. For example, the polynomial -3x^4+5x-2 can be represented by the sequence {(4,-3), (3,0), (2,0), (1,5), (0,-2)}. The sequence contains the pairs of coefficient and degree of each monomial.

This way of representing the polynomial can be used to perform the following operations: addition, subtraction and multiplication of two polynomials as well as the derivative and the integral of a polynomial.

Primary Actor: user

Modelling

Main Success Scenario:

1. The user inserts 2 polynomials in the graphical interface, each polynomial has its own text box with a label in front of it: “First polynomial”, “Second polynomial”.
2. The user presses the button with the chosen operation: “Add”, “Subtract”, “Multiply”, “Integrate”, “Derivative”.
3. The polynomial calculator performs the correspondent operation and displays it in the third text box labeled as “Result of operation”.

Alternative Sequence: Polynomials introduced incorrectly.

1. The user inserts incorrect polynomials.
2. The polynomial calculator will not display any result, waiting for a correct polynomial.

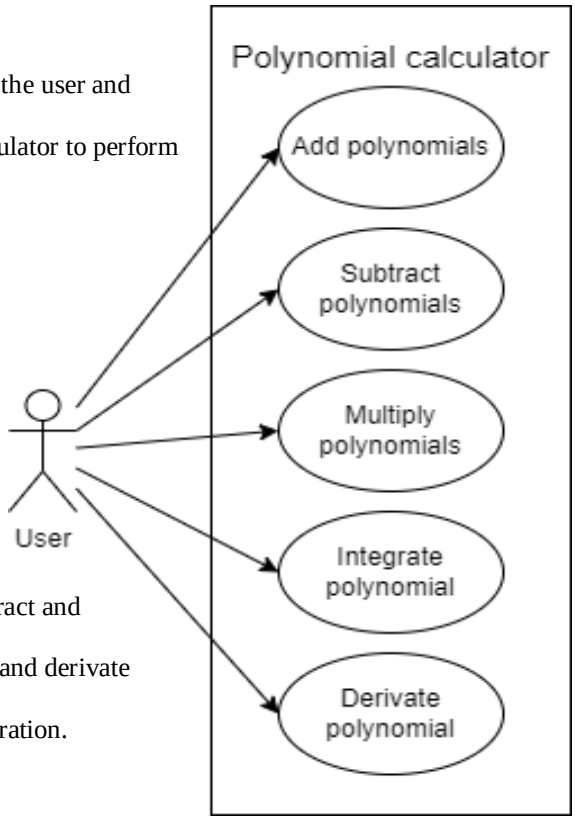
Scenarios:

The user introduces two polynomials in the correspondent text fields. The polynomial should have the degree specified if it greater than one and the sign as well. Here is an example of how the introduced polynomial should look like : +3-x+2x^1-4x^5. For some operations such as addition, subtraction and multiplication both text fields correspondent to the first and the second polynomial should have some values introduced or the result will not be displayed. For the derivative and the integral of the polynomial only the first text field correspondent to the first polynomial is used, so there should be the values introduced. The result will be displayed in the result text field as well.

Use Cases

The polynomial calculator will take the polynomials introduced by the user and perform the chosen operation.
The user should be able to use the functions of the polynomial calculator to perform the following operations by pressing the corresponding button:

- Add polynomials
- Subtract polynomials
- Multiply polynomials
- Integrate polynomial
- Derivate polynomial



Functional requirements:

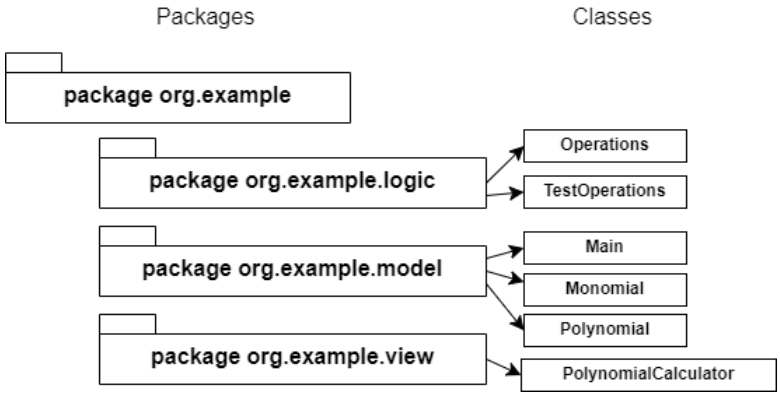
- The polynomial calculator should allow users to insert polynomials.
- The polynomial calculator should allow users to select the mathematical operation.
- The polynomial calculator should allow users to add, subtract and multiply two polynomials.
- The polynomial calculator should allow users to integrate and derivate polynomials.
- The user should be able to see the result of the chosen operation.

Non-Functional Requirements:

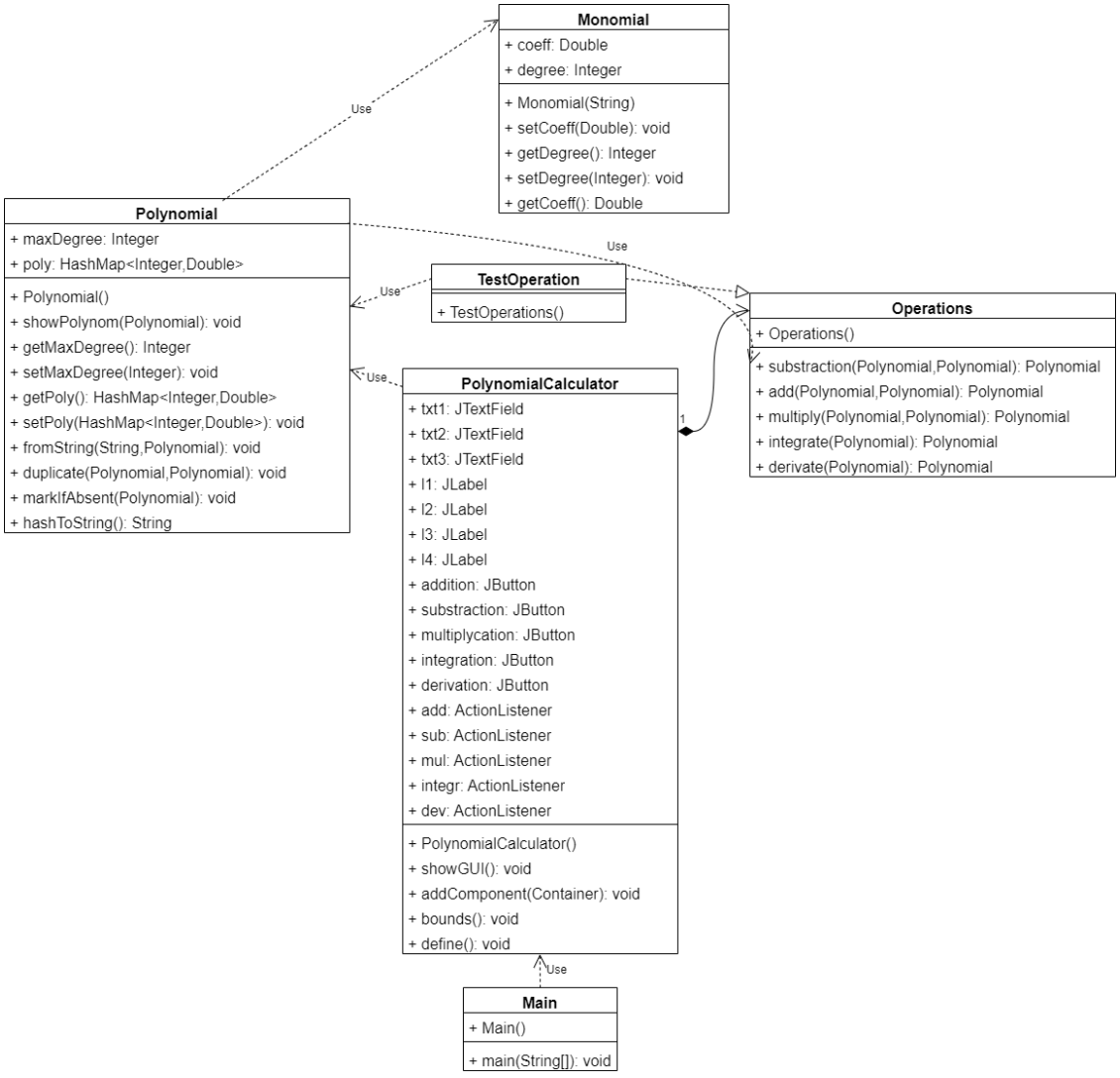
- Usability: The polynomial calculator should be intuitive and easy to use by the user, the user can easily choose any operation by pressing the button with the chosen operation.
- Scalability: The polynomial calculator should be designed to handle an increasing number of users and workloads, with the ability to scale up and down as needed.
- Performance: The polynomial calculator should be able to handle polynomials of large degree and size and should return result quickly and efficiently.
- Security: The polynomial calculator should be designed with security in mind, with appropriate measures taken to prevent unauthorized access or data loss.
- Liability

3. Design

UML Package Diagram

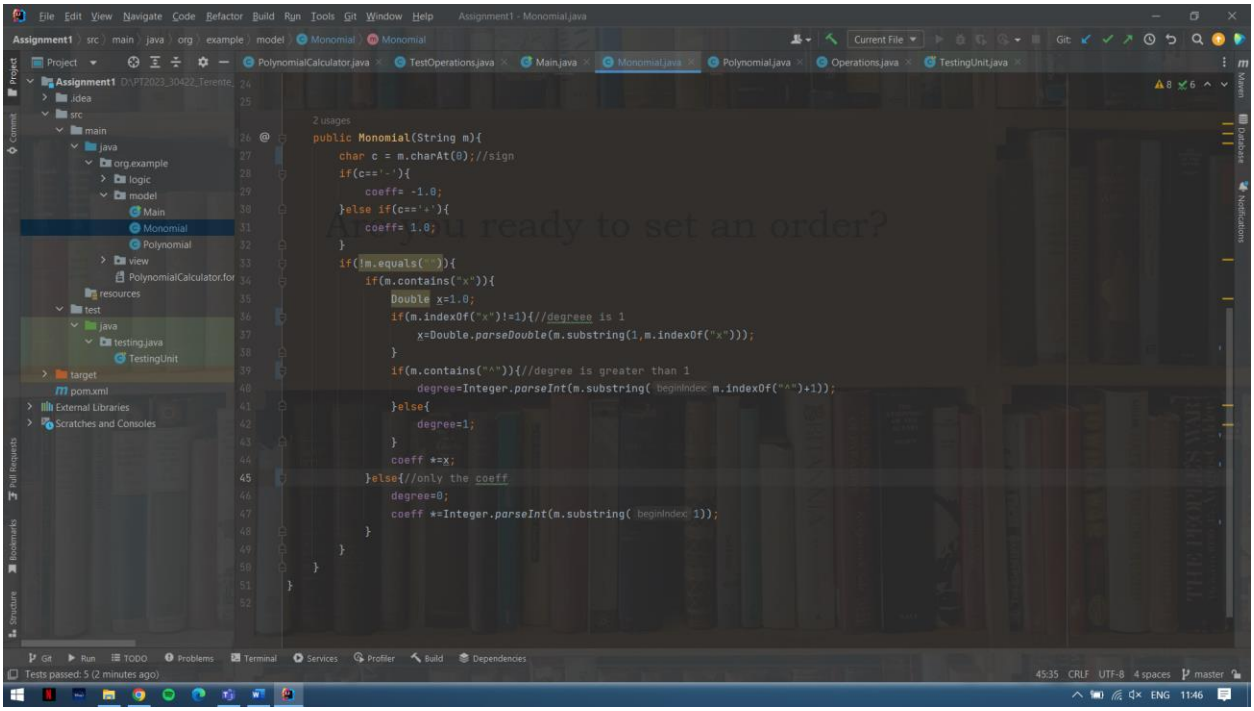


UML Class Diagram



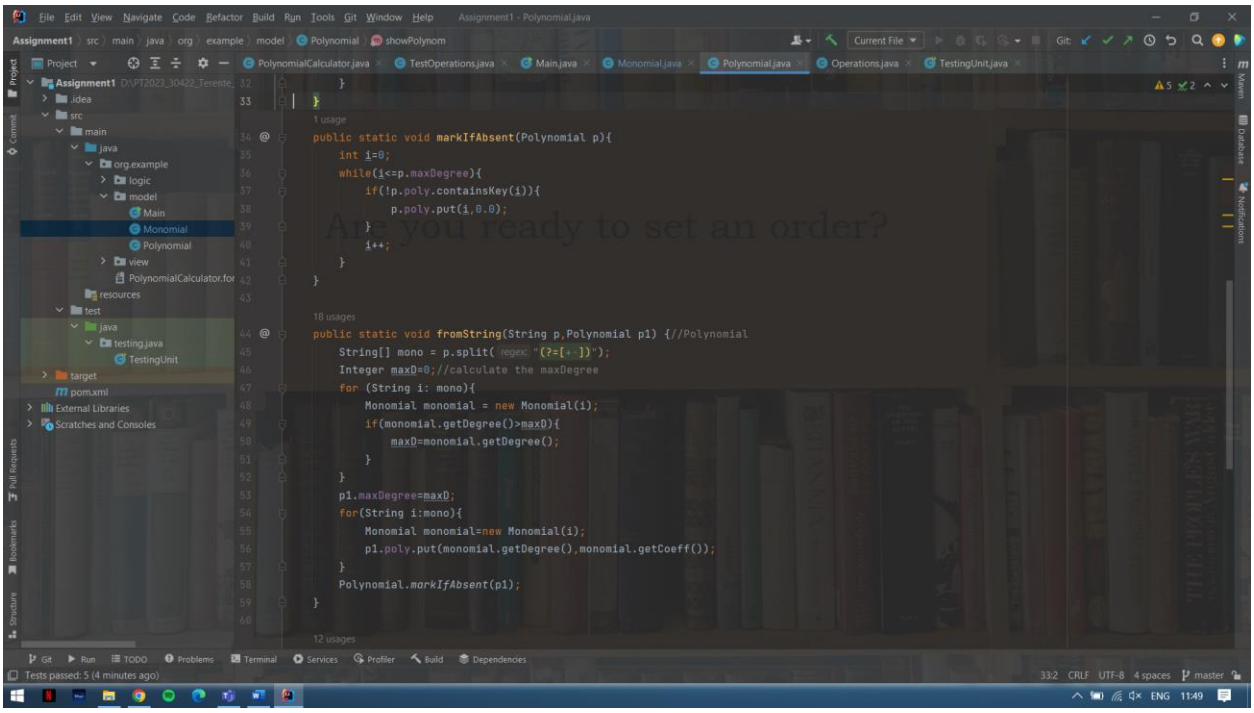
4. Implementation

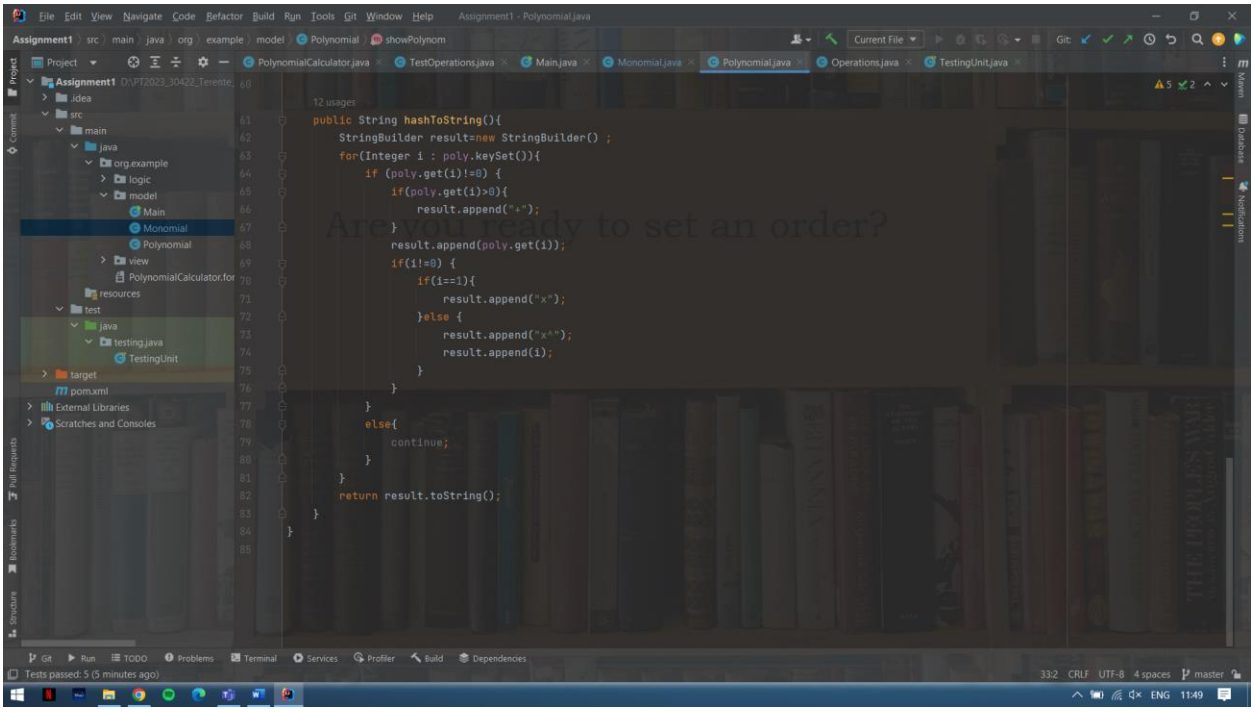
Monomial class is implemented in such a way that I use it for getting the coefficient and degree for each input given as a String monomial, with getters: *getDegree*() and *getCoeff*(). The only method from this class is *public Monomial(String m){}*. It verifies the character situated at the 0 index to find the sign of the monomial. Then separates into two cases: one in which the monomial has a degree greater than zero and contains the variable ”x” and a one in which the degree is zero and the monomial contains only the coefficient. When the monomial contains the variable “x” then the degree can be one and the string does not contain “^” this means only the coefficient needs to be computed, and the other case where I compute the degree, greater than 1, and the coefficient.



Polynomial class is implemented using the *HashMap<Integer,Double>* to store the degree(type Integer) and the coefficient(type Double) for the polynomial. The first variable in the HashMap is the degree used as a key starting from zero to the maxDegree of the polynomial and the second variable is the coefficient which is computed as a correspondent value of the key(degree).
In this class I have the following important methods (taken in order from the project):

- *markIfAbsent(Polynomial p){}* is used to fill with zeros the keys (degrees) in the HashMap where there is no coefficient. It checks every key until the maxDegree and for every one of it, if the key does not exist in the HashMap it puts zero at the correspondent degree.
- *fromString(String p,Polynomial p1){}* is used to get a polynomial string values stored into a HashMap .A string *p* is given as an input. First step, split *p* using regex +/- and for each monomial from the inputted polynomial string I compare to calculate the maxDegree. Next for loop is used to put in the HashMap each degree and coefficient by calling the getCoeff and getDegree from the Monomial class. The method markIfAbsent is used to fill in with zero positions where there are no coefficients.
- *hashToString()* is used to compute back the HashMap into a String to be displayed. For each key in the HashMap verify if the coefficient is different from zero. Here are a few cases :were the coefficient is greater than zero it puts a “+” in the string and after it the value at the correspondent key, if the degree is not zero and it is one than it put only “x” variable, else puts in the string “x^” and the key value. The returned value of the result is of String type.





Operations class has only methods of type Polynomial in which the result returned is correspondent to the called operation.

- *add(Polynomial polynomial1,Polynomial polynomial2){}* . For each key of the two polynomials given as parameter verify if the key are equal then add the coefficients, else if the addition polynomial has no coefficient at the verified key adds it.
- *substraction(Polynomial polynomial1,Polynomial polynomial2){}*. I have used the add method to return the coefficient but with ”-”.
- *multiply(Polynomial polynomial1,Polynomial polynomial2){}*.

- *integrate(Polynomial p){}* here I have used the mathematical formula of integration and implementing it in Java.
- *derivate(Polynomial p{}* here I have used the mathematical formula of derivation and implementing it in Java.

PolynomialCalculator is the class implemented the graphical user interface GUI. I have declared variables for each label: l1, l2, l3 and l4, buttons for each operations: add, subtract, multiply, integrate and derivative and text boxes for the 2 polynomial that need to be introduced and for the result that will be displayed. Every button has an ActionListener implemented that performs the correspondent operations by calling the *Operations* class.

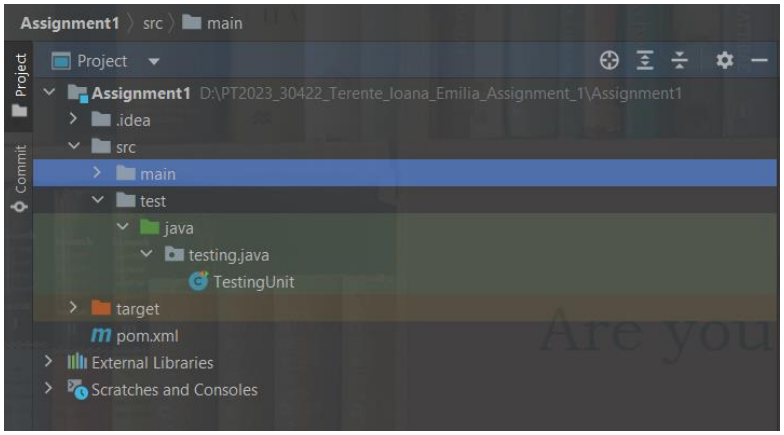
TestingUnit class is used to ensure that the polynomial calculator performs the expected operations and returns the expected result. It is used to test whether correctly adds, subtract, multiplies, derivates and integrates.

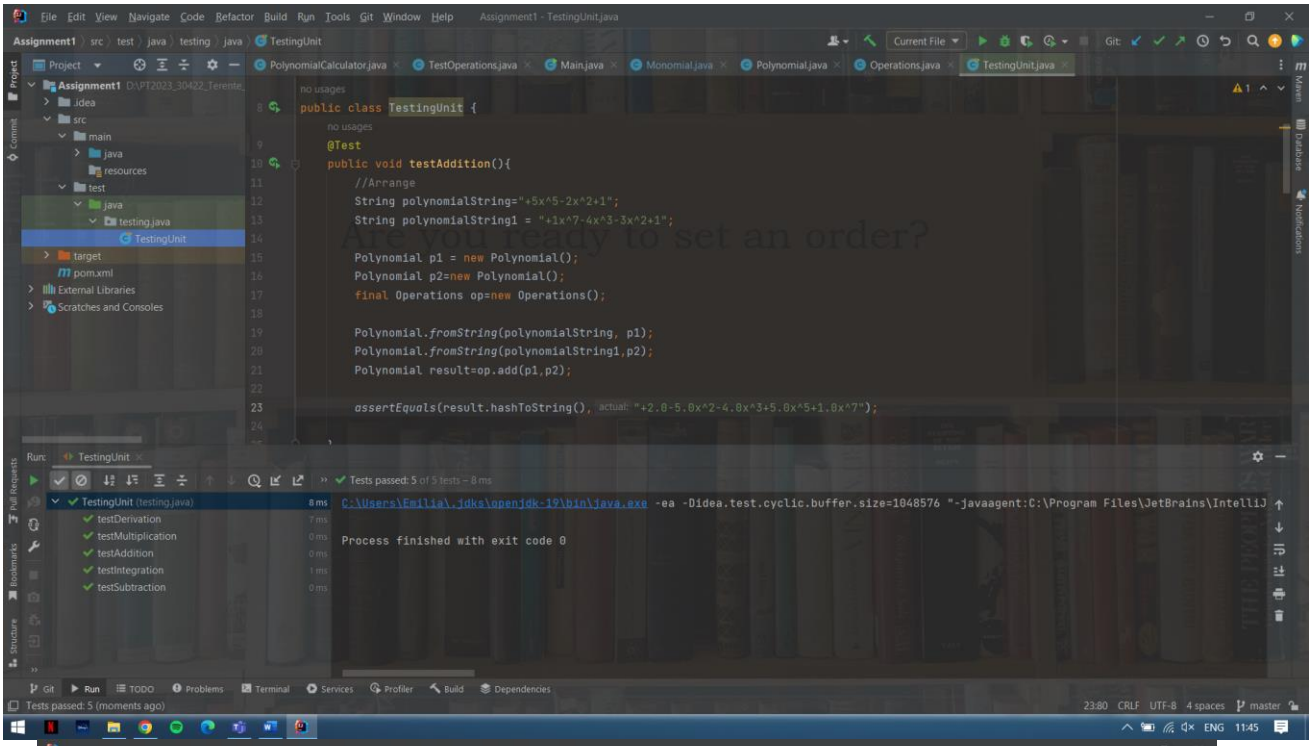
5. Results

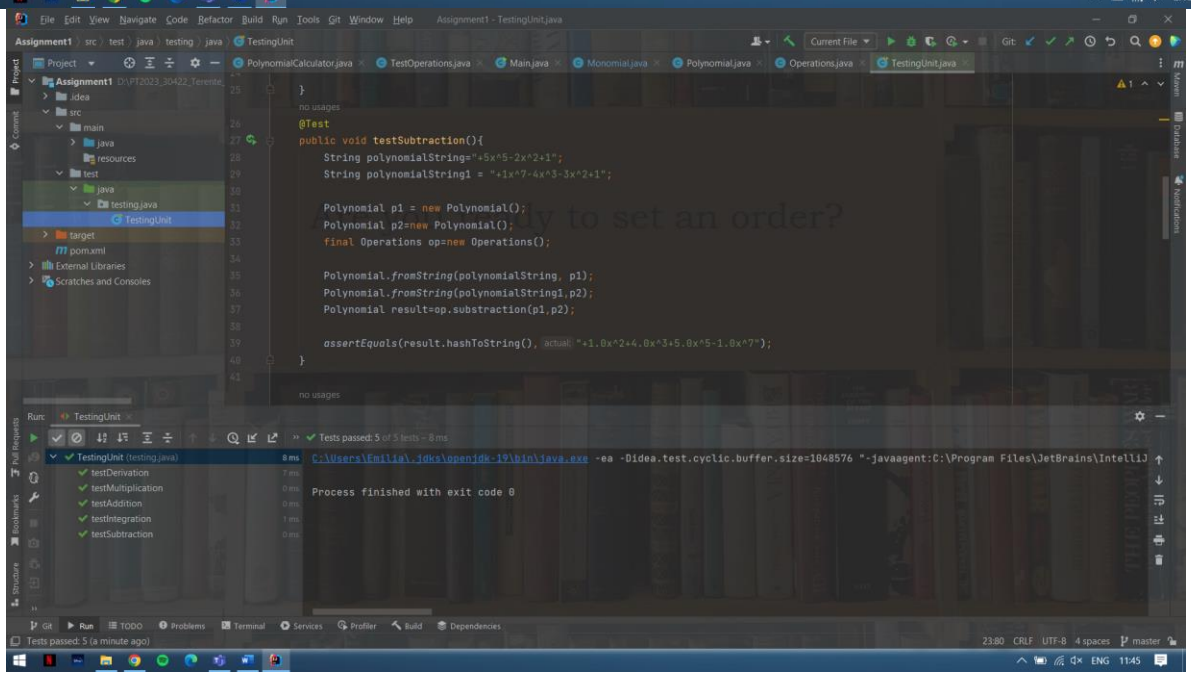
I have performed tests inside the JUnit called *TestingUnit* inside the *test* folder for the following operations: addition: method *testAddition()*, subtraction: method *testSubtraction*(), multiplication: method *testMultiplication*(), integration: method *testIntegration*() and derivation with the correspondent method for testing: *testDerivation*().

Inside each method user for testing I have declared a string, using the method *fromString(String,Polynomial*) to return a polynomial, performing the operations and calling the method *assertEquals*() giving as parameters the result from operation as a string applying *hashToString*() method that takes a polynom and returns a string, to verify if the result is correct.

After introducing all of the operations in the Junit and running the testing framework the output was the expected one resulting that the implemented operations are correct.







6. Conclusions

In conclusion, a polynomial calculator involves working with different objects, such as polynomials, monomials, terms and coefficients.

From the point of view of the OOP: implementing a polynomial calculator helped me learn how to create and manipulate objects, use packages, classes and interfaces, apply concepts such as inheritance and encapsulation.

As for data structures, a polynomial calculator requires efficient storage and retrieval of a polynomial expressions and terms. I have learned how to use HashMap< > to store and manipulate data, and how to optimize the use of memory and processing time.

When implementing a polynomial calculator , I have learned how to use parsing and how to evaluate user input to perform mathematical operations.

For testing and debugging, implementing a polynomial calculator in Java involves writing test cases, debugging code and verifying correctness and efficiency of the program. I have learned how to use testing frameworks such as Junit.

Implementing a polynomial calculator in Java has helped me to develop my programming skills, particularly in object-oriented programming, algorithms and data structures.

As for future developments, I have thinking about making the way the user should introduce the polynomial easier. To get the mandatory '+' sign from the start of the expression out. Other possible development is to integrate to the polynomial calculator other software applications, such as spreadsheets, graphing tools and mathematical modeling software useful for increasing productivity.

7. Bibliography

1. *What are Java classes? - www.tutorialspoint.com*
2. *Fundamental Programming Techniques https://dsrl.eu/courses/pt*
3. *Operations with Polynomials http://content.nroc.org/DevelopmentalMath/COURSE_TEXT2_RESOURCE/U11_L3_T2_text_final.html*
4. *Java HashMap https://www.w3schools.com/java/java_hashmap.asp*
5. *Java - String split() method https://www.tutorialspoint.com/java/lang/string_split.htm*