

**POLYNOMIAL CALCULATOR**

Student: Francesca Dițulescu

Teacher: Viorica Chifu

Table of contents

Assignment objective . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Main objective . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

Sub-objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

Problem analysis, modeling, scenarios, use cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

Problem analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

Solution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Use cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

Overall System Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

Division into packages . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

Division into classes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

UML Class Diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

GUI Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .15

Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

ASSIGNMENT OBJECTIVE

MAIN OBJECTIVE

The main objective of this project ist o design and implement a polynomial calculator with a dedicated graphical user interface through which the user can perform different operations (addition, subtraction, multiplication, division, derivation and integration) on polynomials.

SUB-OBJECTIVES

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

PROBLEM ANALYSIS

The purpose of this project is to solve the following problem: performing polynomial operations on paper is difficult and time consuming when you have to deal with big polynomials and big coefficients.

SOLUTION

As a solution, this project provides a faster and more interactive application that can help the user to perform operations on polynomials easier.

REQUIREMENTS

Functional requirements:

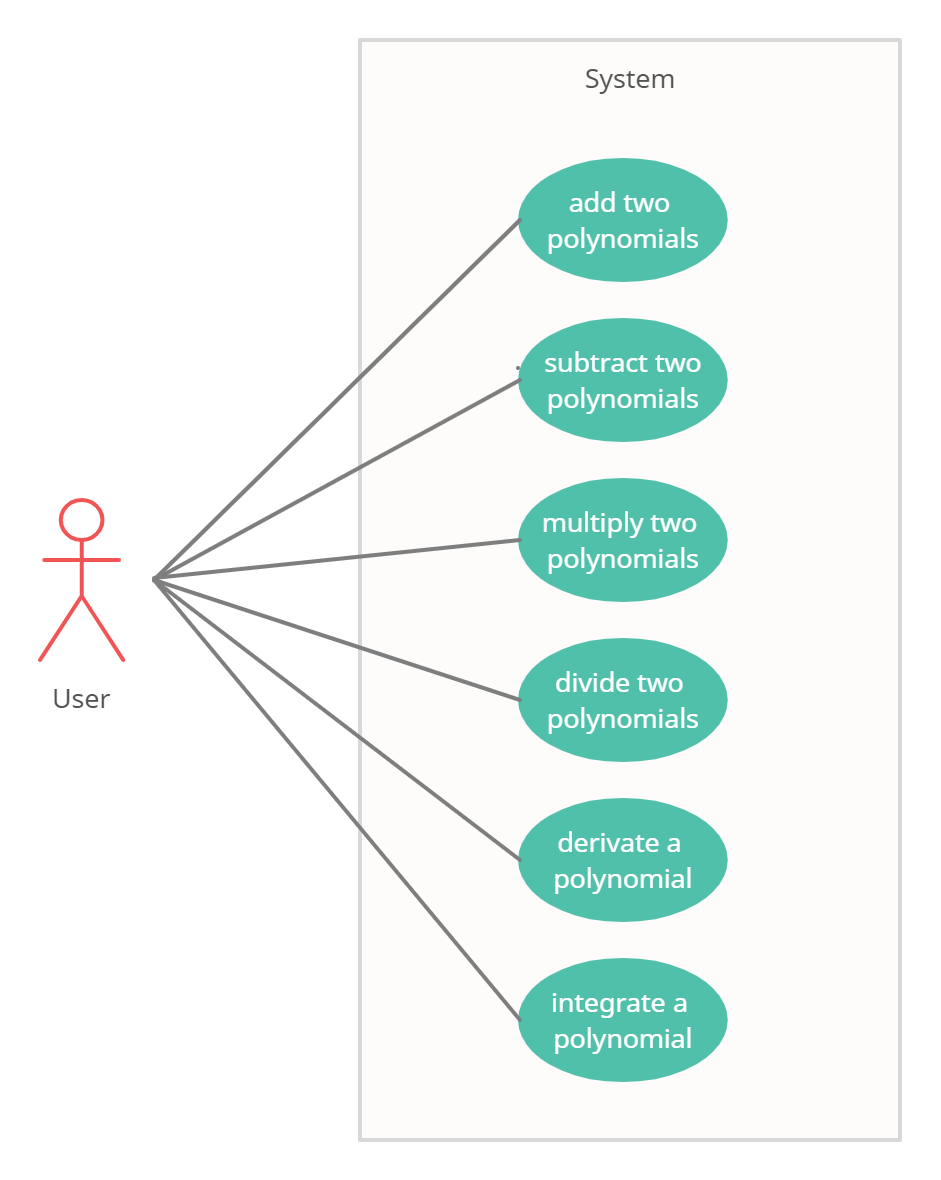
* The polynomial calculator should allow the user to insert data.
* The polynomial calculator should allow the user to select which operation he/she wants to perform.
* The polynomial calculator should be able to read a polynomial written in its algebric form.
* The polynomial calculator should be able to add two polynomials.
* The polynomial calculator should be able to subtract two polynomials.
* The polynomial calculator should be able to multiply two polynomials.
* The polynomial calculator should be able to divide two polynomials.
* The polynomial calculator should be able to derivate a polynomial.
* The polynomial calculator should be able to integrate a polynomial.

Non-functional requirements

* The polynomial calculator should be intuitive and easy to use.
* The polynomial calculator should have a nice and pleasant graphical user interface.

USE CASES

The use case diagram:



1. Add two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “+” (addition) button
3. The polynomial calculator performs the addition of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

2. Subtract two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “-” (subtraction) button
3. The polynomial calculator performs the subtraction of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

3. Multiply two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “\*” (multiplication) button
3. The polynomial calculator performs the multiplication of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

4. Divide two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “/” (division) button
3. The polynomial calculator performs the division of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

5. Derivate a polynomial

Primary actor: User

Main success scenario:

1. The user inserts one polynomial in the graphical user interface
2. The user selects the “ ‘ “ (derivate) button
3. The polynomial calculator performs the derivation of the polynomial and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

6. Integrate a polynomial

Primary actor: User

Main success scenario:

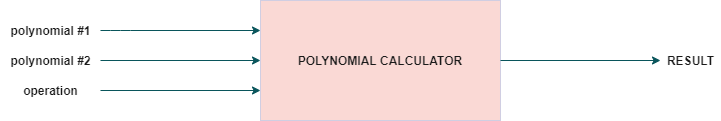
1. The user inserts one polynomial in the graphical user interface
2. The user selects the “ ∫ “ (integrate) button
3. The polynomial calculator performs the integral of the polynomial and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

DESIGN

Level 1: Overall System Design



Level 2: Division into sub-systems/packages

MODEL-VIEW-CONTROLLER ARCHITECTURE

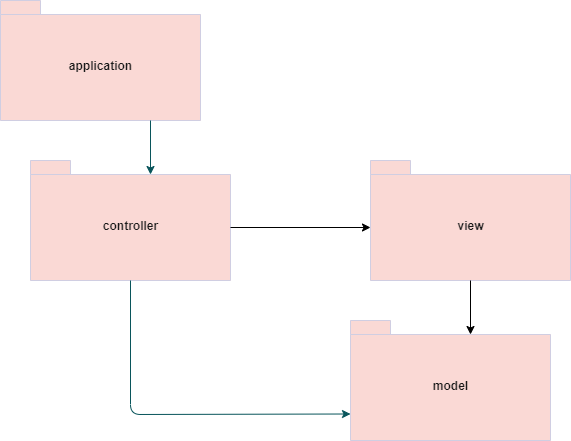
The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

The View component is used for all the UI logic of the application.

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.

My package diagram is the following:



Level 3: Division into classes

a) APPLICATION PACKAGE – contains the Main class with the main method which starts the application.

b) MODEL PACKAGE – contains the classes which model the application:

- Monomial: an element from a polynomial which is extracted from the introduced string in the graphical user interface. It comes with a method for being multiplied with a polynomial as a helper for the implementation of the multiplication operation.

- Polynomial: a list of monomials that has to be used as a term when performing the operations provided by the application. It comes with a few methods:

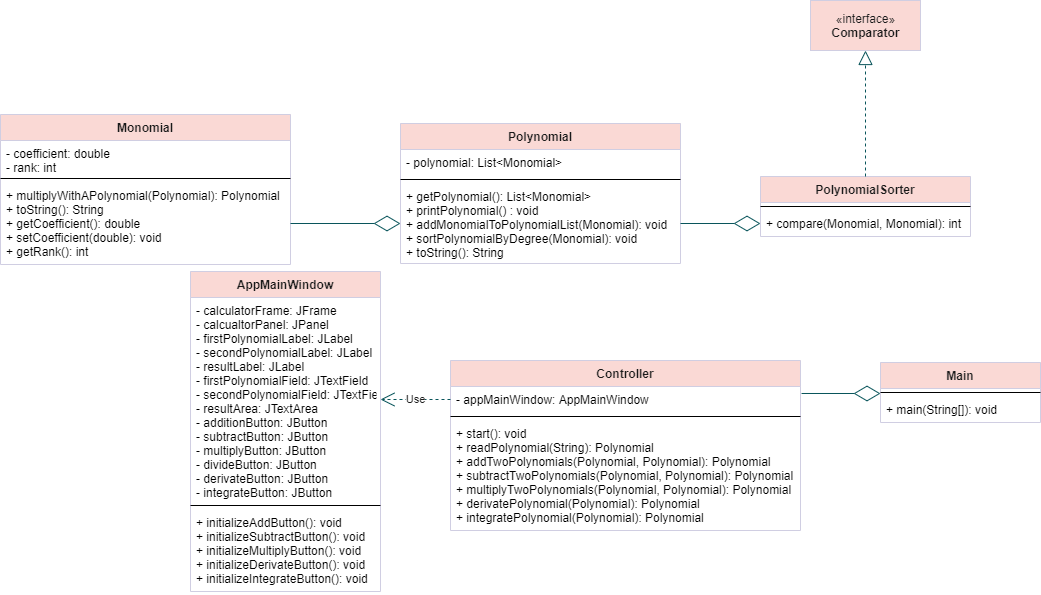
* addMonomialToPolynomialList – method used to add to the polynomial’s list a monomial (used to form the result polynomial after performing a specific operation).
* sortPolynomialByDegree – method used to sort all the monomials from the polynomial’s list in decreasing order by their ranks (degrees) in order to simulate a real mathematical representation of a polynomial.

c) CONTROLLER PACKAGE – deals with the business logic of the application and initializes the buttons from the graphical user interface. It contains the methods that implement all the operations that has to be performed by the application.

Here I also implemented a class PolynomialSorter that implements the Comparator interface in order to sort all the monomials from the polynomial’s list.

d) VIEW PACKAGE – contains the class which creates the graphical user interface (the visual part of the application)

UML CLASS DIAGRAM

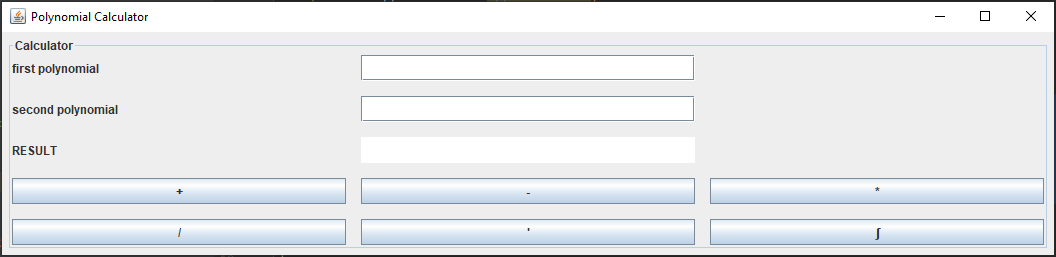


DATA STRUCTURES

For this assignment I have used only ArrayList in order to store the monomials of the polynomial. It was a more efficient approach instead of arrays in terms of the memory occupied. We could have used it for placing each monomial with its rank i to the corresponding ith cell from the array, but a lot of space would have been wasted for the monomials that had coefficient zero (were not in the polynomial) and it would have also been difficult to approximate a suitable size for the array (because it has a fixed one).

GUI DESIGN

The main frame contains three labels, two text fields in which the user can introduce data, one test area where the result of the operations selected is printed for the user to see. Moreover, the interface also provides six buttons, each representing an operation (addition +, subtraction -, multiply \*, divide /, derivate ‘, integrate ∫).



IMPLEMENTATION

In order to explain all the operations and how I thought of implementing, we will consider the two polynomials having the following form:

*P(x) = anxn + an-1xn-1 + … + a1x1 + a0*

*Q(x) = bnxn + bn-1xn-1 + … + b1x1 + b0*

The result RES(x) has been saved in all cases as a new Polynomial object.

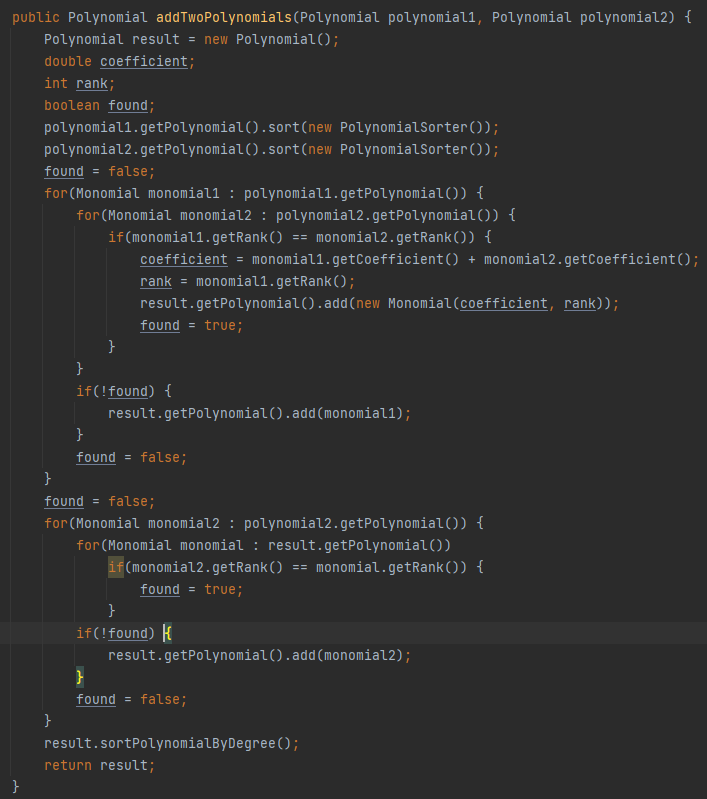
**1. addition**

When adding two polynomials, we have to add the coefficients from the monomials with the same rank:

*RES(x) = (an + bn) xn + (an-1 + bn-1) xn-1 + … + (a1 + b1)x1 + (a0 + b0)*

Now because the polynomials are implemented as a list of monomials sorted in descending order by their rank, the rank does not match the index (on the *i*th element we might not have the monomial of rank *i*). In case a monomial of rank *i* that is not in the first polynomial, then it is just simply added to the result polynomial.

When we find a monomial of rank *i* in both polynomials, we add their coefficients and place the sum in the result polynomial.



**2. subtraction**

When adding two polynomials, we have to add the coefficients from the monomials with the same rank:

*RES(x) = (an + bn) xn - (an-1 + bn-1) xn-1 - … - (a1 + b1)x1 - (a0 + b0)*

Now because the polynomials are implemented as a list of monomials sorted in descending order by their rank, the rank does not match the index (on the *i*th element we might not have the monomial of rank *i*). In case a monomial of rank *i* that is not in the second polynomial, then it is just simply added to the result polynomial (added in the list), but the coefficient multiplied by -1 (since they are subtracted).

When we find a monomial of rank *i* in both polynomials, we add their coefficients and place the sum in the result polynomial.

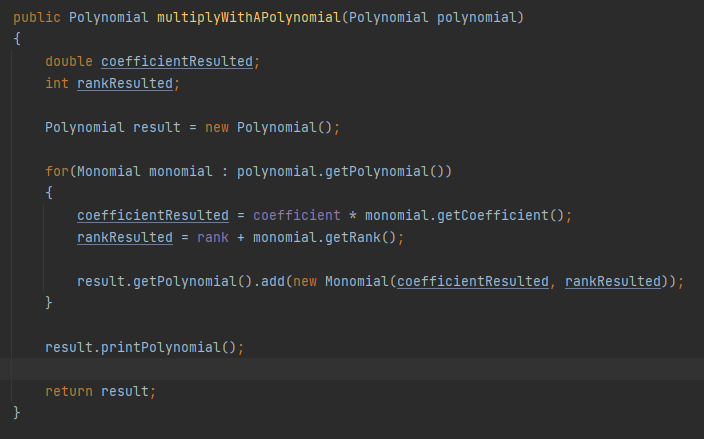


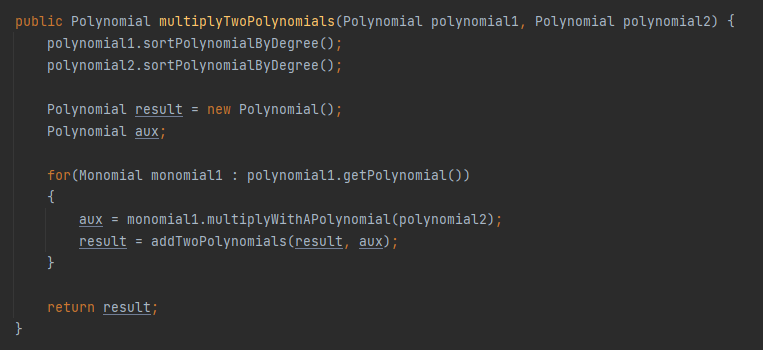
**3. multiplication**

For implementing the multiplication operation, I have considered multiplying each monomial from the first polynomial with the second polynomial and then add all the results in the final polynomial. The mathematical expression for what I mean is the following:

*RES(X) = anxn \* Q(x) + an-1xn-1 \* Q(x) + … + a0 \* Q(x)*

When multiplying a monomial with a polynomial, I simply multiplied each coefficient from the polynomial with the monomial’s coefficient and added to each rank the monomial’s rank (degree).



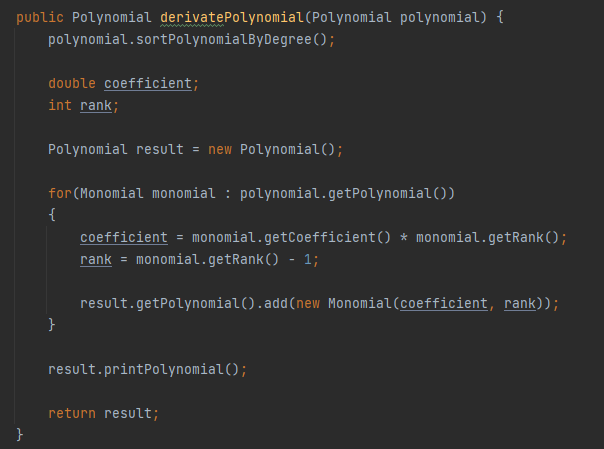


**4. derivation**

The derivation operation has the following mathematical form:

*P’(X) =*

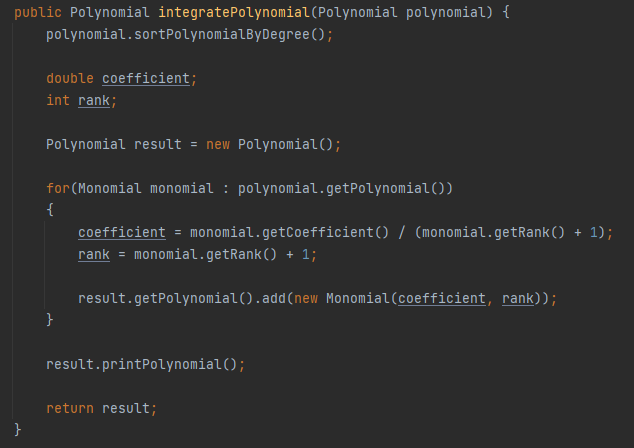
For each monomial, we multiply its coefficient with the current rank and update the monomial with its value decremented by 1.

X

**5. integration**

For this operation, for each monomial we have to divide the coefficient by rank + 1 and also update the monomial’s rank with its value incremented by 1.

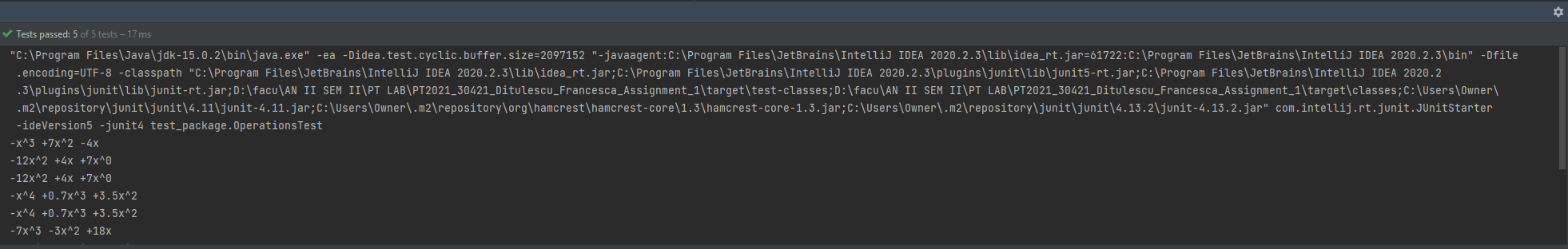
*∫P(x)dx =*



RESULTS – TESTING USING JUNIT

JUnit is a Java unit testing framework that’s one of the best test methods for regression testing. An open-source framework, it is used to write and run repeatable automated tests.

As with anything else, the JUnit testing framework has evolved over time. The major change to make note of is the introduction of annotations that came along with the release of JUnit 4, which provided an increase in organization and readability of JUnits. The rest of this blog post will be written from usages of Junit 4 and 5.



All the tests have passed for all the operations.

CONCLUSIONS

This assignment was a nice introduction to the MVC architecture and has helped me understand how to build a small application in Java while practicing my OOP skills.

BIBLIOGRAPHY

<https://howtodoinjava.com/java/collections/arraylist/arraylist-sort-objects-by-field/>

<https://topic.alibabacloud.com/a/use-a-regular-expression-font-colorredregexfont-to-match-polynomials-polynomial-regexpolynomial_1_31_32668196.html>

<https://stackoverflow.com/questions/34996128/maven-jetty-plugin-daemon-element-not-allowed-here>

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