Technical Documentation of Polynomial Calculator

Adrian-Ioan Tuns

Department of Computer Science
Faculty of Mathematics and Informatics
West University of Timișoara
Email: adrian.tuns99@e-uvt.ro

January 2, 2020

Contents

1 Introduction

1.1 The Idea

The idea for the Polynomial Calculator project appeared as a need for a cross-platform, easy to use calculator for polynomials processing and computing. The calculator works on polynomials of integer coefficients with one variable. The application assumes that the input in correctly introduced by the user and that the polynomials won't have negative-coefficient terms.

1.2 The Scope

The purpose of this document is to present the technical details regarding the application, how it is used and what functions presents. This document targets the end users and the academic audience, mainly students or teachers in the science fields.

2 Resources

2.1 Software Used

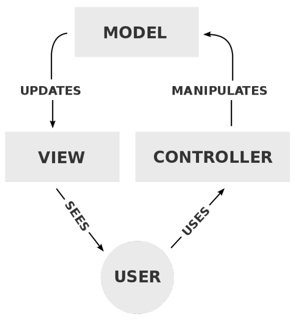
The project source code was written using Java programming language and making use of the software: IntelliJ IDEA, developed by JetBrains.

2.2 Project Development

2.2.1 Design

The application uses the MVC design pattern:

- Model: polynomial.
- View: a simple Graphical User Interface.
- Controller: a class which reads and processes the input from the text fields in the "View" and sends the output back to the "View".



The project contains 4 packages: one for each of the MVC unit: "model", "view", "controller"; and one for the Main function, which combines the other three packages: "mvc".

The "model" package has a "Monomial" class, which is the core of the entire model. It's then divided into "IntegerMonomials" and "RealMonomials", which are part of the "Polynomial" class.

The "view" package consists of a single class, "UserInterface", which creates the Graphical User Interface.

The "controller" package consists of a single class, "Controller", which contains a "UserInterface" object, that reads the input data and process it, parsing the "String" input into "Polynomial" objects.

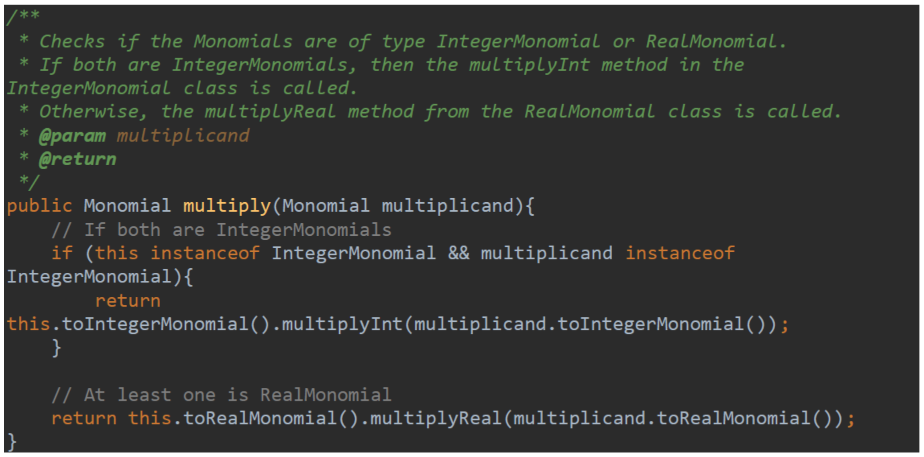
2.3 Documentation of the Implementation

This section will undergo a deep analysis of each part of the application:

1. Package: model

1.1. Monomial

- This is an abstract class, having two children: ”IntegerMonomial” and ”RealMonomial”. This design was chosen because even though the application is meant to work with integers, the division and integration can return real values and it this way the two possible outcomes are separated and solved.
- Two methods are implemented into ”Monomial”, that downcast the class into its two children. A ”compareTo” method is also implemented, for sorting the terms by their coefficients in a polynomial.
- The add, subtract, multiply and divide methods from this class check if the monomials are instances of each of its children and call the correct method for each case (add will either call ”addInt” from ”IntegerMonomial” or ”addReal” from ”RealMonomial”, and so on).
- Has a ”toString” method which checks all the combinations of coefficient and exponent and returns the correct ”String” representation.
- Example:

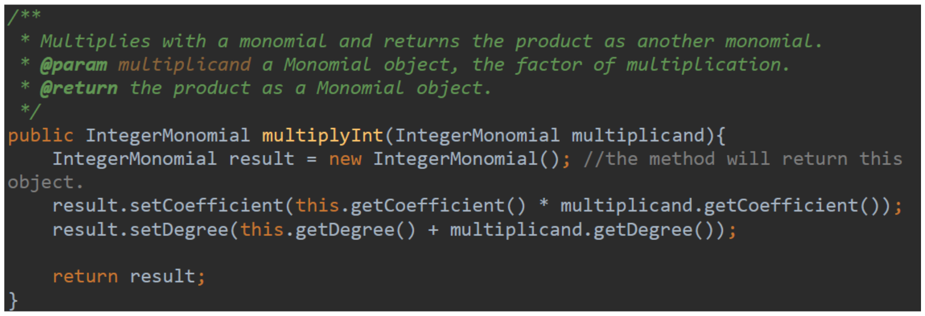


```
/**
 * Checks if the Monomials are of type IntegerMonomial or RealMonomial.
 * If both are IntegerMonomials, then the multiplyInt method in the
IntegerMonomial class is called.
 * Otherwise, the multiplyReal method from the RealMonomial class is called.
 * @param multiplicand
 * @return
 */
public Monomial multiply(Monomial multiplicand){
    // If both are IntegerMonomials
    if (this instanceof IntegerMonomial && multiplicand instanceof
IntegerMonomial){
        return
this.toIntegerMonomial().multiplyInt(multiplicand.toIntegerMonomial());
    }

    // At least one is RealMonomial
    return this.toRealMonomial().multiplyReal(multiplicand.toRealMonomial());
}
```

1.2. IntegerMonomial

- The coefficient saved as ”Number” data type in the parent class is used here as ”Integer” data type.
- All 6 operations are implemented on ”IntegerMonomials”.
- Example:


```
/**
 * Multiplies with a monomial and returns the product as another monomial.
 * @param multiplicand a Monomial object, the factor of multiplication.
 * @return the product as a Monomial object.
 */
public IntegerMonomial multiplyInt(IntegerMonomial multiplicand){
    IntegerMonomial result = new IntegerMonomial(); //the method will return this
object.
    result.setCoefficient(this.getCoefficient() * multiplicand.getCoefficient());
    result.setDegree(this.getDegree() + multiplicand.getDegree());

    return result;
}
```

1.3. RealMonomial

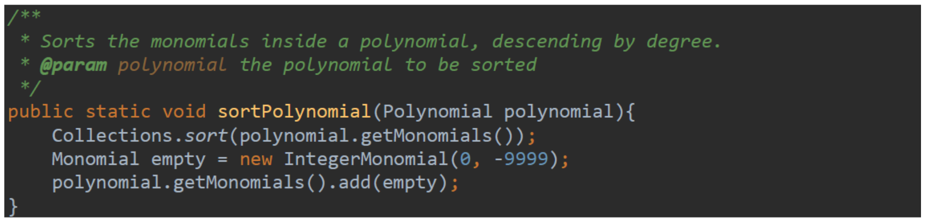
- The coefficient saved as "Number" data type in the parent class is used here as "Double" data type.
- Only four operations are used on "RealMonomials", therefore integration and differentiation were not implemented here.

1.4. Polynomial

- Is the model of the project design and contains a list of monomials implemented as an "ArrayList".
- The operations on polynomials are defined here.
- Has a "toString" method which returns the correct "String" representation.

1.5. PolynomialUtilities

- A class which contains 3 useful methods for polynomials:
- "introduceMonomial(Polynomial, Monomial)": introduces "Monomial" in the "Polynomial" list of monomials.
- "simplify(Polynomial)": used before doing any operation on polynomials, checks if the "Polynomial" has two or more terms of the same degree and adds them together into a single one.
- "SortPolynomial(Polynomial)": sorts the "Polynomial" using the "compareTo" method from the "Monomial". After this method, the polynomial will have its terms in descending order by degree. This method also calls "introduceMonomial" to add a new monomial of degree -9999 (chosen in such a way that there won't be another term of this power). This term helps traversing the list of the "Polynomial" list with Java Iterators. These terms are avoided by the "toString" method and cannot be seen by the user.


```java
/**
 * Sorts the monomials inside a polynomial, descending by degree.
 * @param polynomial the polynomial to be sorted
 */
public static void sortPolynomial(Polynomial polynomial){
    Collections.sort(polynomial.getMonomials());
    Monomial empty = new IntegerMonomial(0, -9999);
    polynomial.getMonomials().add(empty);
}
```

2. Package: view

   2.1. UserInterface

   - This class uses 3 "JPanels" with a "GridLayout(3, 1)" to display the input section, operations and output section.
   - Uses JLabels followed by "JTextField" for introducing or displaying polynomials and "JButtons" for each operation.
   - The private method "createComponents()" adds all the components into a "JFrame".
   - Has getters for input fields and setters for output fields which are processed by the "Controller" part.
   - Has a method for adding listeners(coming from the "Controller") to the buttons and a method for displaying error messages.
   - It implements the "Runnable" interface, having a "run()" method where the "JFrame" is initialized, the default close operation is set, the components are created by calling "createComponents()", and the frame is set to visible.

3. Package: controller

   3.1. Controller

   - This class receives data from the "Model" and from the "View" and sends data back to the "View".
   - It has a method called "toPolynomial(String)" which receives a "String" (from the UI input fields) and processes it using Regular Expressions (regex), making use of Java Pattern and Matcher classes, in order to build a "Polynomial" object. This method is of great importance as it converts the inputs to objects so the polynomial operations can be performed on them. The method is fairly complex, transforming the "String" into 6 substrings(groups) which give all the needed details about the form of one monomial at a time, which is then added to a polynomial that is returned in the end. This method is heavily commented in order to help identify the groups and monomial cases.
   - The private method, setListeners(), is self-called in the constructor of the "Controller" and gives readability to the code. Instead of creating new objects and overriding the method "ActionPerformed", lambda functions are used, making the code much shorter and readable.

3 Testing

The testing is made using JUnit4, inside a special "Test" class. This class provides four @Test methods, for polynomial operations and string-to-polynomial conversion and one @Before method, which initializes two polynomials in order to do operations on them.

What is tested	Input data	Expected result	Pass/Fail
String to Polynomial conversion	"3xˆ4+2xˆ3+x"	"3xˆ4+2xˆ3+x"	Pass
Polynomial addition	"3xˆ4+2xˆ3+x" "5xˆ2+2x+4"	"3xˆ4+2xˆ3+5xˆ2+3x+4"	Pass
Polynomial difference	"3xˆ4 + 2xˆ3 + x" "5xˆ2 + 2x + 4"	"3xˆ4+2xˆ3-5xˆ2-x-4"	Pass
Polynomial multiplication	"3xˆ4 + 2xˆ3 + x" "5xˆ2 + 2x + 4"	"15xˆ6+16xˆ5+16xˆ4+ +13xˆ3+2xˆ2+4x"	Pass

4 Constraints

The application assumes that the input is introduced correctly, having any number of terms of the form Cx^D, Cx, x^D, x, $-x$ or C, where C is an integer number representing the coefficient, D is a non-negative integer representing the degree(exponent) of the term, x is a character representing the polynomial and ˆ is also a character which is translated as "to the power of". The terms are separated by $+$ or $-$, with or without spaces in between.

If the input is not properly introduced, instead of giving an error, the application will search for valid characters and ignore the rest. For example, if the introduced input would be xˆabc+3#, the read polynomial would be $x+3$. However, this is not completely stable at the moment, because the application will not always give the correct result, as the foreign characters could make the application ignore some of the actual valid characters.

If the input has the negative degree "-1" for one of the terms(e.g. x^{-1}), and the desired operation is integration then the result will be wrong, the specific term being considered 0. The result for the term in this specific case should be a logarithm, but logarithms are outside the scope of this calculator application.

5 End-User License Agreement

End-User License Agreement (”Agreement”)
==

Last updated: January 02, 2020

Please read this End-User License Agreement (”Agreement”) carefully before clicking the ”I Agree” button, downloading or using Polynomial Calculator (”Application”).

By clicking the ”I Agree” button, downloading or using the Application, you are agreeing to be bound by the terms and conditions of this Agreement.

This Agreement is a legal agreement between you (either an individual or a single entity) and Polynomial Calculator and it governs your use of the Application made available to you by Polynomial Calculator.

If you do not agree to the terms of this Agreement, do not click on the ”I Agree” button and do not download or use the Application.

The Application is licensed, not sold, to you by Polynomial Calculator for use strictly in accordance with the terms of this Agreement.

License
- - - - - - -

Polynomial Calculator grants you a revocable, non-exclusive, non-transferable, limited license to download, install and use the Application solely for your personal, non-commercial purposes strictly in accordance with the terms of this Agreement.

Third-Party Services
- - - - - - - - - - - - - - - - - - - -

The Application may display, include or make available third-party content (including data, information, applications and other products services) or provide links to third-party websites or services (”Third-Party Services”).

You acknowledge and agree that Polynomial Calculator shall not be responsible for any Third-Party Services, including their accuracy, completeness, timeliness, validity, copyright compliance, legality, decency, quality or any other aspect thereof. Polynomial Calculator does not assume and shall not have any liability or responsibility to you or any other person or entity for any Third-Party Services.

Third-Party Services and links thereto are provided solely as a convenience to you and you access and use them entirely at your own risk and subject to such third parties’ terms and conditions.

Term and Termination
- - - - - - - - - - - - - - - - - - - -

This Agreement shall remain in effect until terminated by you or Polynomial Calculator.

Polynomial Calculator may, in its sole discretion, at any time and for any or no reason, suspend or terminate this Agreement with or without prior notice.

This Agreement will terminate immediately, without prior notice from Polynomial Calculator, in the event that you fail to comply with any provision of this

Agreement. You may also terminate this Agreement by deleting the Application and all copies thereof from your mobile device or from your computer.

Upon termination of this Agreement, you shall cease all use of the Application and delete all copies of the Application from your mobile device or from your computer.

Termination of this Agreement will not limit any of Polynomial Calculator's rights or remedies at law or in equity in case of breach by you (during the term of this Agreement) of any of your obligations under the present Agreement.

Amendments to this Agreement

- -

Polynomial Calculator reserves the right, at its sole discretion, to modify or replace this Agreement at any time. If a revision is material we will provide at least 30 days' notice prior to any new terms taking effect. What constitutes a material change will be determined at our sole discretion.

By continuing to access or use our Application after any revisions become effective, you agree to be bound by the revised terms. If you do not agree to the new terms, you are no longer authorized to use the Application.

Governing Law

- - - - - - - - - - - - -

The laws of Romania, excluding its conflicts of law rules, shall govern this Agreement and your use of the Application. Your use of the Application may also be subject to other local, state, national, or international laws.

Contact Information

- - - - - - - - - - - - - - - - - - -

If you have any questions about this Agreement, please contact us:
Faculty of Mathematics and Informatics
West University of Timișoara
Address: Bulevardul Vasile Pârvan 4, Timișoara 300223
Email: adrian.tuns99@e-uvt.ro

Entire Agreement

- - - - - - - - - - - - - - - -

The Agreement constitutes the entire agreement between you and Polynomial Calculator regarding your use of the Application and supersedes all prior and contemporaneous written or oral agreements between you and Polynomial Calculator.

You may be subject to additional terms and conditions that apply when you use or purchase other Polynomial Calculator's services, which Polynomial Calculator will provide to you at the time of such use or purchase.