PT

Polynomial Operations

Homework 1

Chirodea Mihai – Cristian

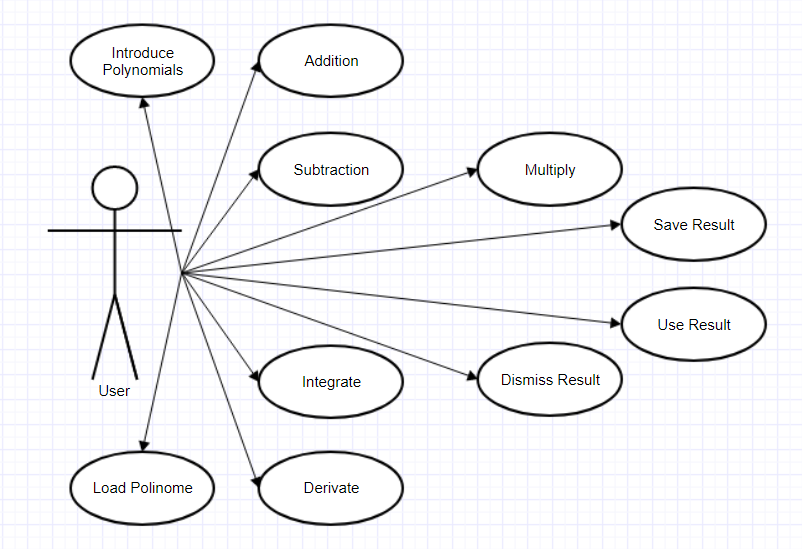
Group 30424

1. **Project Objective**

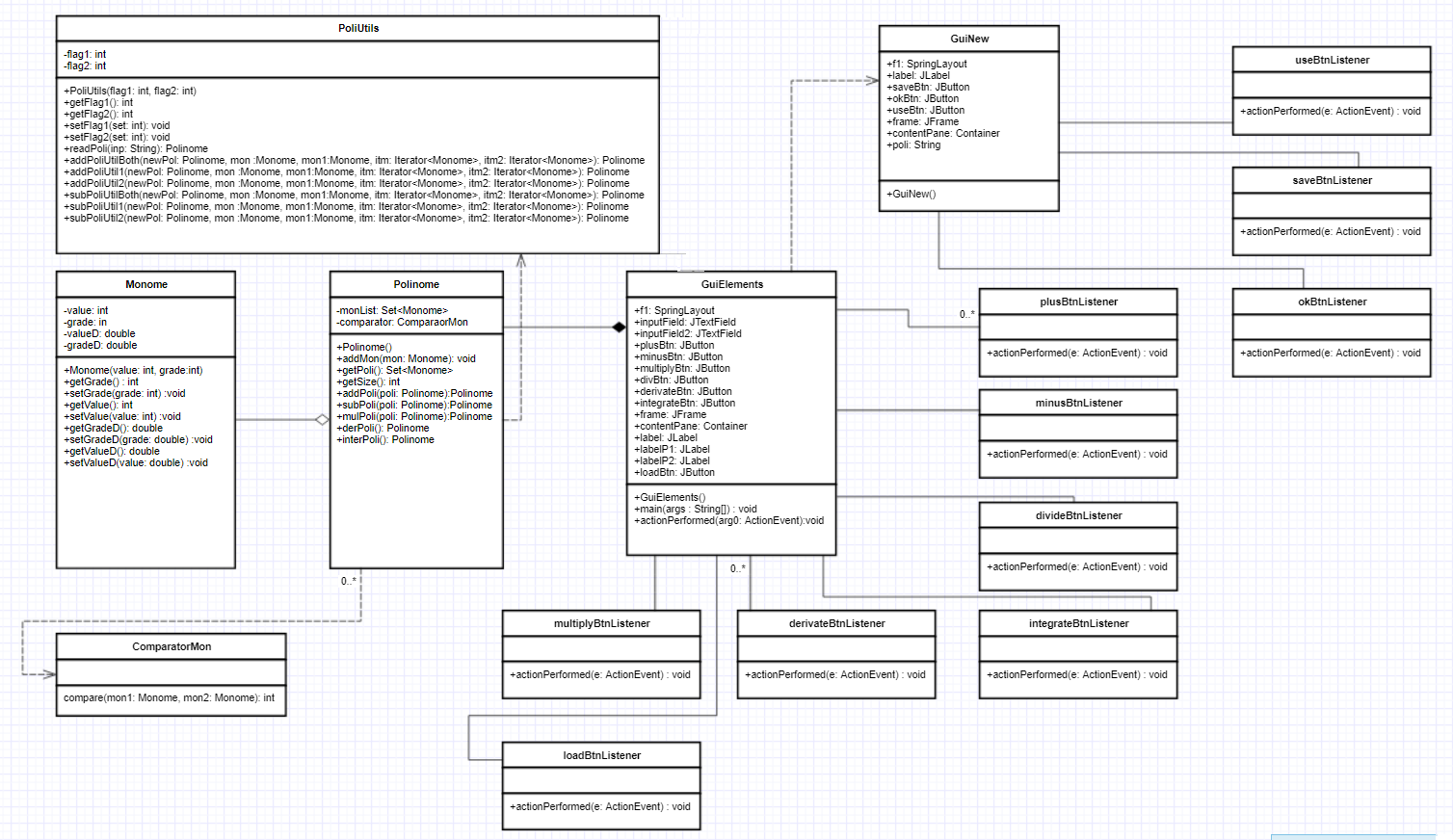
* The main objective of this project was to create a Java application capable of performing operations on 2 integer polynomials such as addition, subtraction and optionally division, multiplication, integration and derivation. The program also required an easy to use interface, created using Java’s Swing package, for the user to interact with.
* The project was realized in Eclipse and it’s able to perform the operations listed above, minus division. In order for the program to give correct results, a few steps have to be followed:
  + - **First Step**: The user has to introduce the 1-st polynomial in the first text field labeled accordingly, same for the 2-nd.
    - **Second Step**: The user has to verify that the polynomials introduced at step 1 have the following format: **NO** spaces between the monomials or between members of the monomial, **IF** a monomial is negative it needs to be written in the following manner “ (-coefficient)X^exponent” with a “+” in front, **ORDER** of the members doesn’t matter, the program automatically sorts them, **LASTLY**, if the exponent or coefficient of any monomial is 1, that ‘1’ has to be written (This does not apply in case of 0).
    - **Third Step:** After Step 1 and Step 2 are completed, the user can now, press any operation (minus division) and a new window will be presented to the user, containing the result and a few options: Option to save the result (Saves the result in a file, the result can then be used with the Load button on the main window. Another option is to use the result directly (Writes the result in the 1-st polynomial field). Lastly, there is the OK button, which, when pressed, closes the result window. **CAUSION**, pressing ‘x’ to close the window will close the program.
* If the steps above have been followed correctly, then the program should give the correct result. Here’s an example of a simple polynomial addition:
  + For the first polynomial we will have: **2x^8+7x^5+9x^10**
  + For the second polynomial we will have: **2x^8+(-20)x^20 (Notice that order doesn’t matter and that number of elements doesn’t matter, the program takes care of that)**
  + Addition result: **(-20)x^20 + 9x^10 + 4x^8 + 7x^5**.

1. **Problem Analysis**

* **Functional Requirements:**
  + In order to create a program that satisfies all the needs for the operations to occur, it’s important to know the components of a polynomial, to be more precise, the coefficients and exponents. Understanding that, we can decompose them into data structures and perform the needed operation by creating methods and classes to deal with said data structure.
  + So, first step is figuring what sort of structure to use, in order to store the polynomial. As the specification says, the user introduces the whole polynomial, so what the program has to do, is to extract the coefficients and exponents from it and represent the polynomial using them.
  + Although the requirements state that the inputs are integer values, for integration, the values have to be real. That is why I used integers for addition, multiplication and subtraction and for integration I converted said values into their real values.
  + In the unlikely case that the user forgets a monomial, he should be able to introduce it at the end of the polynomial, thus getting rid of the troubles of going through the polynomial and inserting it where it should be. For this I created a comparator, that automatically sorts the introduced polynomial.
* **Use Case:**

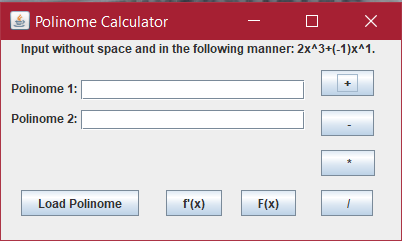


1. **Projection**

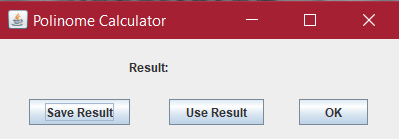
* **UML Diagram:**
* **Data Structures:**
  + In order to best represent the polynomial, I chose to create 2 Classes, first one containing the definition of a monomial and second containing the definition of a polynomial. The Polynomial contains a LinkedHashSet of Monomial type.
  + To best represent the 2 windows that the user will interact with I created 2 separate interfaces, one containing the first part where the user introduces the data and chooses the operation and the other containing the second part that displays the result.
* **Projection Decisions:**
  + A big decision I had to make was to either use an ArrayList or a HashSet to represent the polynomial, in the end I settled with a HashSet as I was familiar with it.
  + This decision made a huge impact on my program as it was a lot harder to do the addition and subtraction on different sized polynomials and it took a lot more lines of code to get it to work.
  + Another decision I made was to use text fields as input so that the user could introduce the whole polynomial. In my mind, this would have been more user friendly than simply making the user introduce the coefficients and exponents of each monomial.
  + And, again, this decision had a significant impact on the program, as I had to extract the values from the text, which in turn placed a few constraints on the user (Not being able to write with spaces and having to put negative coefficients in parenthesis). In time, the code required to extract the values got quite big and I made the decision to move it to the PoliUtils class, as the code was used by multiple buttons.
  + And in the end, I made the decision to put the result in a different window and give the user the option to do something with it (he could save it, use it, or dismiss it) and I also made the decision to split the subtraction and addition methods as they were too long.
  + This decision was easy to implement as it was straight forward a required a bit of work with files. And for the methods, it required a bit of thinking on how to split them so that the code wouldn’t be so big.

1. **Implementation**

* **Packages:**
  + In order to organize the classes, I split them into two packages:
    - **codes:** Contains the Polinome, Monome, PoliUtils, ComparatorMon
    - **guiInterface:** Contains the classes for the 2 intefaces.
* **Monomial Class:**
  + Its name is Monome, it contains 2 definitions of integers named value and grade and 2 definitions for real with the same name(Used for integration). Value stores the coefficient of the monomial and grade stores the exponent.
  + This class contains the following methods:
    - **The Constructor:** It initializes the value and grade.
    - **getGrade():** It’s an int method that returns the grade of the monomial.
    - **getValue():** It’s an int method that returns the coefficient of the monomial.
    - **setGrade(int val):** It’s a void method that sets the exponent of the monomial.
    - **setValue(int val):** It’s a void method that sets the coefficient of the monomial.
    - **getGradeD():** It’s a double method that returns the exponent of the monomial in its real representation.
    - **getValueD():** It’s a double method that returns the coefficient of the monomial in its real representation.
    - **setValueD(double val):** It’s a void method that sets the coefficient in its real representation.
    - **setGradeD(double val):** It’s a void method that sets the exponent in its real representation.
* **Polynomial Class:**
  + Its name is Polinome, it contains 2 stuctures. First one is named monList, and it’s a LinkedHashSet and the second one is named comparatorMon and it’s of ComparatorMon type.
  + monList stores the list of monomials in descending order, this is done by sorting using comparatorMon object.
  + This Class contains the following methods:
    - **The Constuctor:** Initializes monList.
    - **addMon(Monome mon):** adds “mon” to the LinkedHasSet.
    - **getPoli():** returns monList.
    - **getSize():** returns the size of monList //Not used in the program
    - **addPoli(Polinome poli):** returns a new polynomial created by adding monList from poli with monList from the current class.
    - **subPoli(Polinome poli):** returns a new polynomial created by subtracting monList in poli from monList in current class.
    - **mulPoli(Polinome poli):** returns a new polynomial created by multiplying monList from poli with monList from current class. The result is then verified in another class.
    - **derPoli():** returns a new polynomial that is created from deriving the monList.
    - **interPoli():** returns a new polynomial that is created from integrating the monList of the current class.
* **Comparator Class:**
  + Named ComparatorMon, it’s an auxiliary class that is used by Polinome to sort its monList. It implements the Comparator<Monome> interface.
  + It has only one method:
    - **compare(Monome mon1, Monome mon2):** Compares the exponents of the 2 monomials, and returns which one is bigger.
* **Polinome Utilities Class:**
  + Named PoliUtils, it serves as group of methods that other classes rely on. It has methods that help in special addition cases or special subtraction cases, methods that extract the values from the text.
  + It has the following methods:
    - **PoliUtils(int flag1, int flag2):** The constructor of the class that initializes the 2 variables that are used in some of the other methods.
    - **readPoli(String inp):** Takes the input string from the user and extracts the values of the polynomial, which then are added to monList in a newly created Polinome object.
    - **addPoliUtilBoth, addPoliUtil1, addPoliUtil2:** depending on the case, those mothods are used in the addition operation.
    - **subPoliUtilBoth, subPoliUtil1, subPoliUtil2:** depending on the case, those methods are used in the subtraction operation.
* **GUI Class 1:**
  + Named GuiElements, it contains the definitions of Java.Swing elements such as buttons, labels, textFields, containers and frames.
  + This class is the one responsible for creating the GUI with which the user interacts.
  + As methods it has the constructor that initializes the interface and its elements and that adds listeners to the buttons.
  + It uses Spring Layout which, because of it’s constraints, can be modified however the programmer wants.
  + It has 7 buttons (Addition, Subtraction, Multiplication, Division, Integration, Derivation, Load Polinome).
  + This class is also where the program is executed from.
  + The Interface for this class looks as follows:

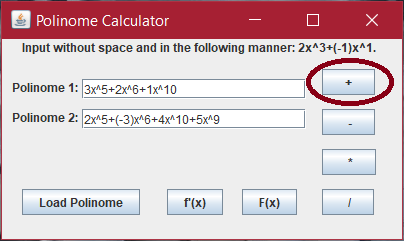
****

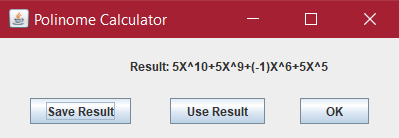
* **The Listener Classes:**
  + Each of those classes is used as a listener for each of the buttons, and when a button is pressed, the class corresponding to each button is called, executing the code in them.
  + As example, when pressing the addition button, the program jumps to the plusButtonListener class, which in turn calls the readPoli method in PoliUtils, and then calls the addPoli method in Polinome. After that, it takes the resulting polynomial and displays it on the second interface GuiNew.
* **GUI Class 2:**
  + Named GuiNew, it’s the interface where the user sees the result and gets to choose what to do with it.
  + It contains definitions of Java.Swing elements (JButton, JLabel, JFrame).
  + As methods it contains the constructor that initializes the interface and add the listeners to all the buttons.
  + It contains 3 buttons (Save, Use, OK) which have the following functions:
    - **Save Button:** Saves the result in a file, which can be accessed at a later date using Load Button in the main interface.
    - **Use Button:** Takes the result and copies it in the text field of polynomial 1, also resetting the text field from polynomial 2.
    - **OK Button:** Closes the Display GUI and returns to the main GUI.
    - **Caution! Closing any window with the ‘x’ button will close the program, not the window.**
  + The interface for this class looks as follows:

****

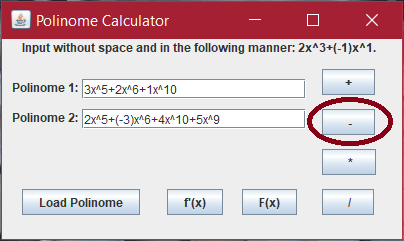
1. **Testing & Results**

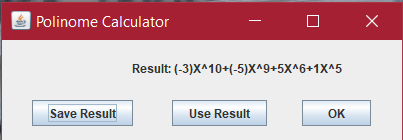
* In order to prove the correctness of the program, I have chosen the following cases and results:
  + **Test Input:** **3x^5+2x^6+1x^10 (Polynomial 1) 2x^5+(-3)x^6+4x^10+5x^9 (Polynomial 2)**
  + **Expected Results:** 
    - Addition: 5x^10+5x^9+(-1)x^6+5x^5

****

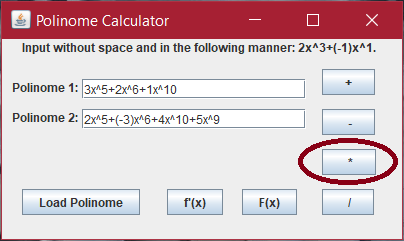
****

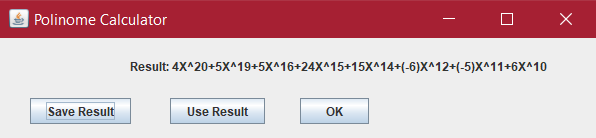
* + - Subtraction: (-3)x^10+(-5)x^9+5x^6+1x^5

****

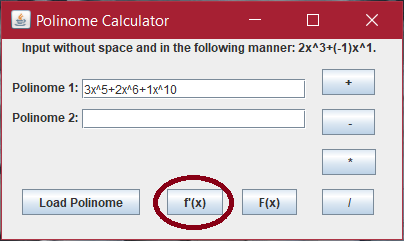


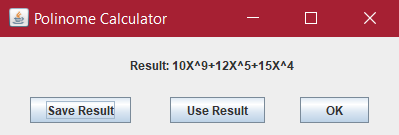
* + - Multiplication: 4x^20+5x^19+5x^16+24x^15+15x^14-6x^12+-5x^11+6x^10

****

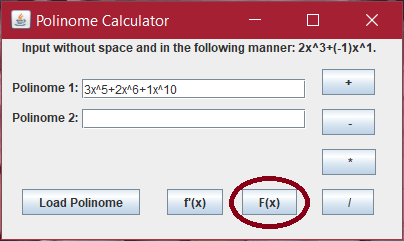
****

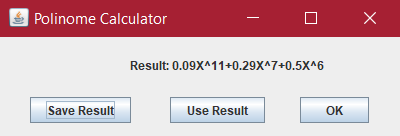
* + - Derivation: 10x^9+12x^5+15x^4

****

****

* + - Integration: 0.09x^11+0.29x^7+0.5x^6





1. **Conclusions**

* To sum it all up, I had to create a program that would do operations on polynomials and I created it using a LinkedHashSet to store the coefficients and exponents. Using methods to process the input data, the result is given accordingly.
* **What I learned:**
  + I learned to create GUI using only code, not drag & drop like I used to do.
  + I better familiarized myself with Hash Sets and OOP use of Objects.
  + I improved on working with the GUI.
  + I learned to extract information from a line of text using the split method in java.
  + I learned to create and use Button Listeners.
  + I learned how to link multiple interfaces.
* **Future improvements:**
  + Adding the division operation.
  + Making the program more flexible to user input errors.
  + Removing user constraints on the input.
  + Eventually changing the text input field with a better-looking style (i.e. when pressing “^” to create a small label above “x” where the user introduces the exponent.

1. **Bibliography**

* <https://stackoverflow.com/questions/3481828/how-to-split-a-string-in-java>
* <https://stackoverflow.com/questions/22313639/getting-polynomial-exponents-in-java>
* <https://docs.oracle.com/javase/tutorial/uiswing/components/panel.html>
* <https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html>
* <https://docs.oracle.com/javase/tutorial/uiswing/layout/spring.html>
* <https://stackoverflow.com/questions/5993779/use-string-split-with-multiple-delimiters>
* <https://stackoverflow.com/questions/22212412/sort-a-set-in-reverse-order>
* <https://stackoverflow.com/questions/17370909/sorting-a-linkedhashset>