March, 2016

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

Laboratory Work – Assignment 1

Polynomial Processing

Name: *Popa Florin*

Group: *30425*

Table of Contents

1. Introduction

1.1 Problem specification

The problem is defined as follows: “Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.”

1.2 Personal approach

The program will have a graphic interface, through which the user can enter data (the two polynomials) and select the desired operation. The interaction will be done through the mouse pointer and the application’s text fields and buttons. A unique button is assigned to each of the following operations:

- ADDITION
- SUBTRACTION
- MULTIPLICATION
- DERIVATION
- INTEGRATION

The processed result will then be displayed in a unique “result” field.

2. Design

2.1 UML diagrams

a) Use Case Diagram

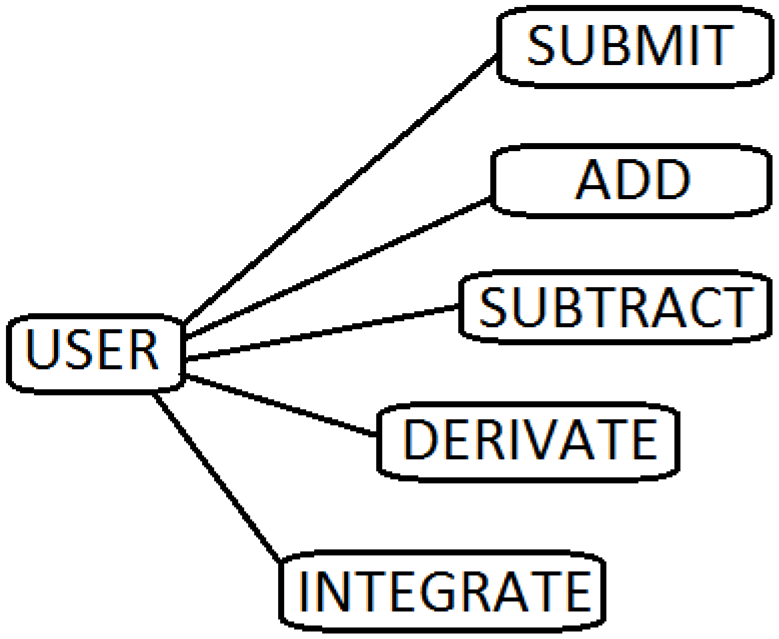


Illustration 1: UseCase Diagram

The use case diagram presents the user that interacts with the application. He can perform several actions such as adding, subtracting, multiplying, derivate, integrate two polynomials and submit the operands with the submit command.

a) Class Diagram

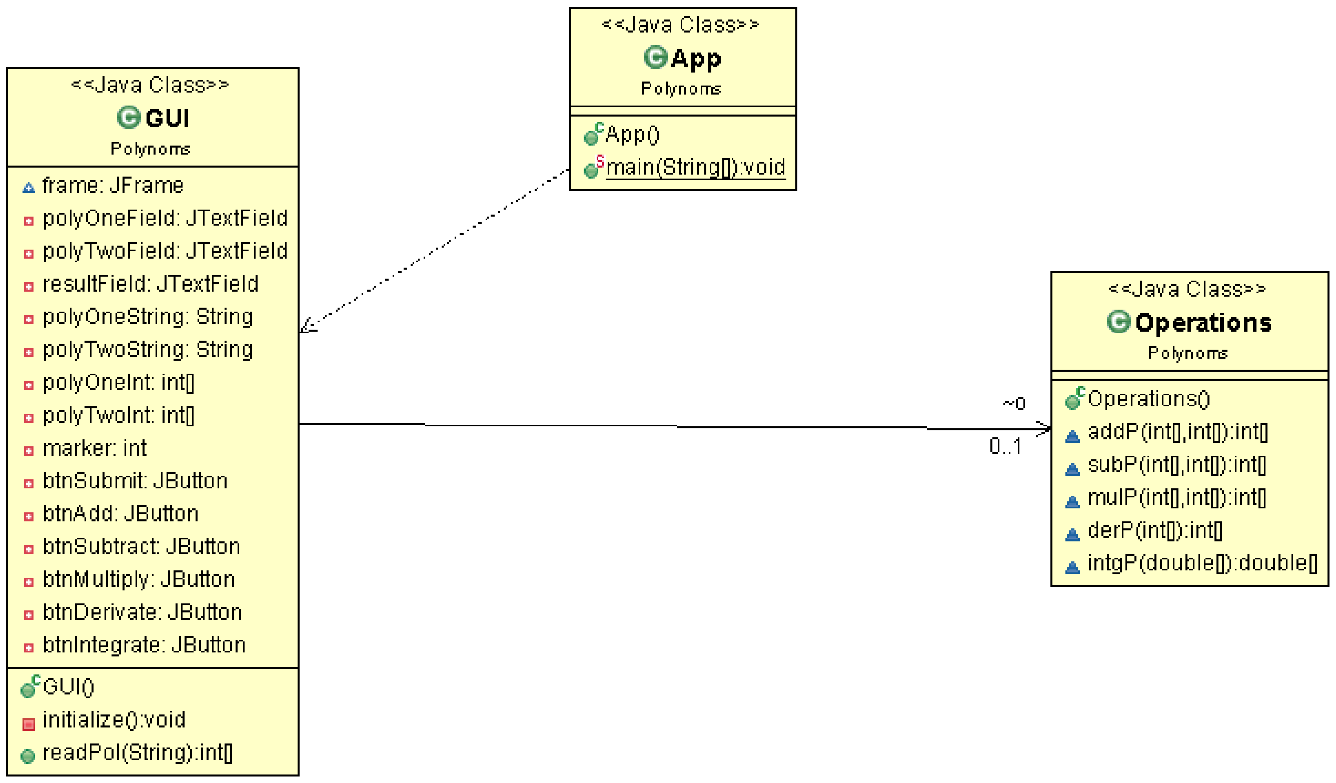


Illustration 2: Class Diagram

2.2 Classes Design

a) GUI Class

This class is designed to create a graphical user interface so the application would be easier to use. The frame and all the graphic models were created using Window Builder, which is an Eclipse IDE add-on that helps the programmer design the frame visually and then translate it into code.
The GUI's constructor calls the ***initialize()*** method, which is the engine of the class.

The methods of the GUI class:

- ***initialize()***

  In this method, the frame is created, and all the interactive elements are added. The frame's layout is ***null***, therefore all the elements are added at a particular set coordinate. Also, here are added the action listeners for each button and the corresponding event after interaction.

- ***readPol(String s)***

  This method is used to read the user's input from the related text fields and store each integer number in an array. This is done by converting the given string to a whole integer number and then store each integer digit, one by one, in a result array. This method is used to transform the string that holds the information from the JTextField and transforms it into an array of integers. The array will hold the coefficients of the polynom and the position will be the power of X.

  More specifically, for the string 4 0 3 5 results the following array:

  Array [0] =5 => x^0 coefficient is 5
  Array [1] =3 => x^1 coefficient is 3
  Array [2] =0 => x^2 coefficient is 0
  Array [3] =4 => x^3 coefficient is 5

  The algorithm works based on a string parsing function. This function divides the string in different parts depending on the indexes we want. In this case, we consider the information between the index where there is a "   ". In this way every coefficient is put at the right place in the array.

The attributes of the GUI class:

```
JFrame frame; //The frame object

private JTextField polyOneField; //The text field of the first polynom

private JTextField polyTwoField; //The text field of the second polynom

private JTextField resultField; //The text field of the result
private String polyOneString; //The variable that will be used to store the string entered in the first polynom text field

private String polyTwoString; //The variable that will be used to store the string entered in the second polynom text field

private int[] polyOneInt = new int[100]; //The array that will be used to store the first polynom coefficients (max 100)

private int[] polyTwoInt = new int[100]; //The array that will be used to store the second polynom coefficients (max 100)

private int marker = 0; //A marker used to implement error messages

private JButton btnSubmit; //The SUBMIT button

private JButton btnAdd; //The ADD button

private JButton btnSubtract; //The SUBTRACT button

private JButton btnMultiply; //The MULTIPLY button

private JButton btnDerivate; //The DERIVATE button

private JButton btnIntegrate; //The INTEGRATE button
```

The interaction:

The submit and addition button will be exemplified, since all the others are similar.

The action listener for the submit button will signal the moment it has to look in the text fields and retrieve the data. The data retrieved will be in String format and it will be transformed into an array of integers. Most importantly, if a text field contains characters other than numbers, no data will be retrieved from that text field.

After the submit button has been pushed and the data has been saved, it is safe to press the addition button. When pressed, the program will find which of the arrays is longer and initialize a new array of that length and through an object of type operations, the "addP" method with 2 parameters will be called and it will calculate the sum and return an array. The array will then be transformed into a string through the formula :

```
temp = temp + " " + Integer.toString(resultInt[i]) +  "X^" + i + " ";
```

The result will have the following form: "5X^3+2X^2+0X^1+3X^0".

While the addition, subtraction and multiplication methods will need information from both text fields, the derivation and integration methods will only need information from the first text field.

b) Operations Class

All the methods from this class are called from GUI and have the purpose to execute the demanded operation with the two polynomials given. These methods receive as parameters one or two polynomials and can return, depending on the case, a third polynomial which will be displayed in the result text field.

The methods of the Operations class:

- Addition, subtraction, multiplication (Dependent on two polynomials)

They receive as parameters two polynomials and depending on the case, they compute the specified operation; these methods return a third polynomial which will be displayed.

- Integration and derivation (Dependent only on the first polynomial)

They receive as parameters one polynomials and they return a second polynomial which will be displayed.

c) App Class

The class of the main method. In the main method, a GUI object is created, therefore the constructor is called and the window is created. There are also called some predefined functions of the JFrame to set the window visible, place it in the middle of the screen, deny resizing, and set the title.

3. Packages and Interfaces

A Java package is a mechanism for organizing Java classes into namespaces. Java packages can be stored in compressed files called JAR files, allowing classes to download faster as a group rather than one at a time. Programmers also typically use packages to organize classes belonging to the same category or providing similar functionality. A package provides a unique namespace for the types it contains. Classes in the same package can access each other's package-access members.

A package allows a developer to group classes (and interfaces) together. These classes will all be related in some way – they might all have to do with a specific application or perform a specific set of tasks.

For this application the following packages are imported, each of them having a certain role for the proper working of the application. We import them in the GUI Class (most of them relate to the user interface properties):

1) import java.awt: Contains all of the classes for creating user interfaces and for painting graphics and images. A user interface object such as a button or a scrollbar is called, in AWT terminology, a component. The Component class is the root of all AWT components.

a. java.awt.Font: The Font class represents fonts, which are used to render text in a visible way. A font provides the information needed to map sequences of characters to sequences of glyphs and to render sequences of glyphs on Graphics and Component objects.

2) import java.awt.event : Used in interacting with input devices such as the mouse and keyboard

a) java.awt.event.ActionEvent: A semantic event which indicates that a component-defined action occurred. This high-level event is generated by a component (such as a Button) when the component-specific action occurs (such as being pressed). The event is passed to every ActionListener object that registered to receive such events using the component's addActionListener method.

b) java.awt.event.ActionListener: The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

3) import javax.swing: Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

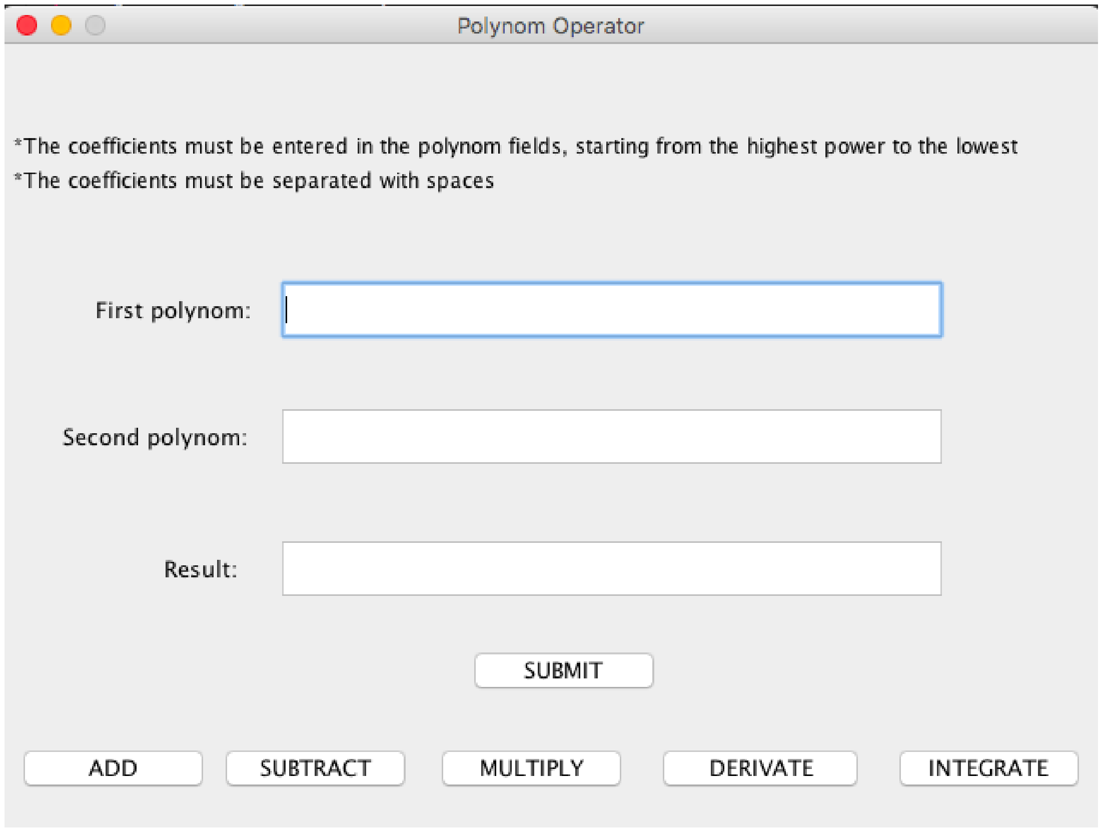
a) javax.swing.JButton: Buttons can be configured, and to some degree controlled, by Actions. Using an Action with a button has many benefits beyond directly configuring a button.

b) javax.swing.JFrame: An extended version of java.awt.Frame that adds support for the JFC/Swing component architecture.

c) javax.swing.JLabel: A display area for a short text string or an image, or both. A label does not react to input events. As a result, it cannot get the keyboard focus. A label can, however, display a keyboard alternative as a convenience for a nearby component that has a keyboard alternative but can't display it.

A JLabel object can display either text, an image, or both. You can specify where in the label's display area the label's contents are aligned by setting the vertical and horizontal alignment. By default, labels are vertically centered in their display area. Text-only labels are leading edge aligned, by default; image-only labels are horizontally centered, by default.

4. User Interface



*The coefficients must be entered in the polynom fields, starting from the highest power to the lowest
*The coefficients must be separated with spaces

Illustration 3: Application Window

When running the application, the "Polynom Operator" window will open and it will provide to the user the possibility of giving inputs and choosing what operation he or she likes to be executed. This window is constructed in the GUI class using some predefined classes and instructions.

The user interface is based on the properties of the above mentioned packages. All the objects we need are declared as attributes of the GUI class and they are initialized in the constructor of this class.

The listener interface is for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked. In this case the only events that occur are when the user clicks on one of the operation buttons from the graphical interface.

The actionListener for the Submit button has an extra try and catch block for verifying if all the characters in the text field are numbers. If there is a character that is not a number, then that polynomial will not be saved and an error message will be displayed in the result field. In addition, if an operation button is pressed before submitting any input data, the result field will display an error message as well.

5. Conclusions

Developing this application was an interesting challenge. It was a very good practice as a laboratory assignment. Of course, many features could be implemented in future versions, such as mathematical display of the polynomials, other operations, input data from a file, etc. Overall, this assignment taught me how to manage time better and how to work with WindowBuilder.

6. Bibliography

- https://docs.oracle.com/
- http://stackoverflow.com
- https://wiki.eclipse.org