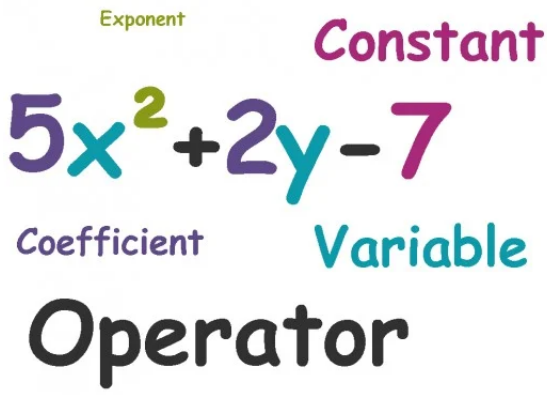
CSCI-605 Advanced Object-Oriented Programming Concepts

Homework 1: Polynomials



1 Introduction

We will use a *Native Array* to represent a single variable polynomial, whose coefficients are integers. The index of a coefficient in the array indicates the power of the term. The first coefficient in the array is a constant term (power of 0), the second coefficient in the array is a term with a power of 1, and so on. If there is no term of a particular power, less than the degree of the polynomial, the array will store a coefficient value of 0. For example:

Native Array	Polynomial	String
[1]	1	"1"
[3, -1]	-x + 3	"-x + 3"
[0, 3]	3x	"3x + 0"
[2, -1, -2, 1]	$x^3 - 2x^2 - x + 2$	"x^3 + -2x^2 + -x + 2"
[-5, 0, 0, 3, 3, 1]	$x^5 + 3x^4 + 3x^3 - 5$	"x^5 + 3x^4 + 3x^3 + -5"

1.1 Goals

Students will gain experience working with

- Basic built-in data types
  - Primitive types
  - The `String` class
  - Arrays
- Control flow statements

1.2 Provided Files

1. **problems.zip** contains the ten java programs you must analyze for Activity 1.
2. **src.zip** contains the starter code for the polynomials program (Activity 2).

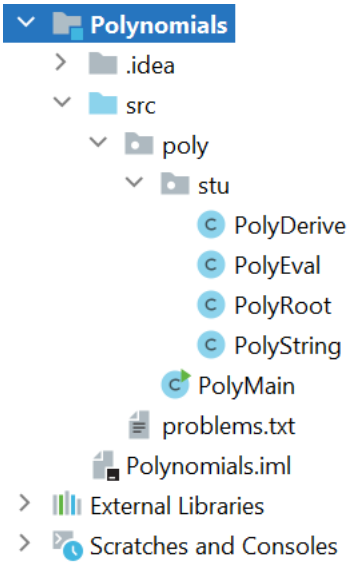
2 Implementation

2.1 Project Structure

Follow these steps to set up a new IntelliJ project for this homework and configure it with the starter code.

1. Create a new Java Project in IntelliJ for this assigment. Each week's assignment should be its own separate project - do not try and group them together.
2. Unzip the starter code on your computer. It should create a **src** directory, with Java source files underneath it.
3. Either copy/paste, or drag and drop the src folder to the root of your project in IntelliJ and allow it to copy or move the entire contents. **Make sure you do not copy/move over the zip file by accident!**
4. Make sure you are running Java 17 or higher.

Your project's structure should look as follows:



2.2 Activity 1: Problems

Before you start implementing the polynomials program, you will analyze ten Java programs with error/s that either prevent the code from compiling correctly or running correctly.

- Unzip the **problems.zip** file and you should see the ten Java programs.
- In your project **src** folder is a text file, `problems.txt`. You should fill this file out with your answers. Make sure that you indicate for each whether the problem is compile-time or run-time, and a brief description of what the problem is. You must use your own words to explain the problem, do not copy the error description from the console.

2.3 Activity 2: Polynomials

You are being given a complete main program, `PolyMain`. You can use this to test the functionality of the four polynomial activities you will implement:

1. `PolyString`: Convert polynomials into strings to be displayed.
2. `PolyEval`: Evaluating a polynomial for a value of x.
3. `PolyDerive`: Compute the derivative of a polynomial.
4. `PolyRoot`: Compute the root of a polynomial.

The main program tests all four components mentioned above. In order to run it, you need to enter the polynomial on the command line:

- Right click in the source code and select **Run PolyMain**.
- The program should run and display a usage message that you need to enter the polynomial on the command line.
- Look for the run configuration that just got created for `PolyMain`. Pull down and select **Edit Configurations...**
- In the window that comes up, enter the polynomial under **Program arguments**, for example -5 -1 2 0 3. Click **OK** after entering it.
- Run the main program again. The program must:
  1. reads the polynomial from the command line and displays it in the console (Activity 2.1).
  2. prompts the user for a value of x. This is what will be used to evaluate the polynomial you just entered (Activity 2.2).
  3. computes/displays the derivative of the polynomial (Activity 2.3).
  4. computes/displays the root of the polynomial, using Newton's method (Activity 2.4).

You are now ready to work on the four polynomial activities. There are TO DO comments in the code that you should replace with your own code.

2.4 Activity 2.1: PolyString

First you will convert a polynomial to its string representation to be displayed. For example:

Native Array	Polynomial	String
[1]	1	"1"
[3, -1]	-x + 3	"-x + 3"
[0, 3]	3x	"3x + 0"
[2, -1, -2, 1]	$x^3 - 2x^2 - x + 2$	"x^3 + -2x^2 + -x + 2"
[-5, 0, 0, 3, 3, 1]	$x^5 + 3x^4 + 3x^3 - 5$	"x^5 + 3x^4 + 3x^3 + -5"

Implement the `getString` method. Note that this method is going to be using a lot of string concatenation, which is fairly costly using the "+" operator. If you are interested in using something more efficient, refer to the StringBuilder class and this post.

2.5 Activity 2.2: PolyEval

For this part, you will evaluate a polynomial given a supplied `double` value for the variable `x`. For example:

Cmd args	Native Array	Polynomial	x	Evaluation
1	[1]	1	0.0	1.0
3 -1	[3, -1]	-x + 3	4.5	-1.5
0 3	[0, 3]	3x	-2.0	-6.0
2 -1 -2 1	[2, -1, -2, 1]	$x^3 - 2x^2 - x + 2$	2.0	0.0
-5 0 0 3 3 1	[-5, 0, 0, 3, 3, 1]	$x^5 + 3x^4 + 3x^3 - 5$	-3.9	-391.16669

Implement the `evaluate` method. You can use the examples from the table above to test the correctness of your method.

In addition to evaluating a polynomial, you will also implement a boolean function, `isZero`, that takes a polynomial and returns whether it is a zero polynomial or not. For example:

Cmd args	Native Array	Polynomial	Zero Polynomial?
0	[0]	0	true
1	[1]	1	false
0 3	[0, 3]	3x	false
0 0 0 4	[0, 0, 0, 4]	$4x^3$	false

2.6 Activity 2.3: PolyDerive

For this part, you will compute the derivative of a polynomial by implementing the `computeDerivative` method. The starter code allocates an array of size 0. This must be generalized. For example:

Cmd args	Native Array	Polynomial	Derivative as Polynomial
1	[1]	1	0
3 -1	[3, -1]	-x + 3	-1
0 3	[0, 3]	3x	3
2 -1 -2 1	[2, -1, -2, 1]	$x^3 - 2x^2 - x + 2$	$3x^2 - 4x - 1$
-5 0 0 3 3 1	[-5, 0, 0, 3, 3, 1]	$x^5 + 3x^4 + 3x^3 - 5$	$5x^4 + 12x^3 + 9x^2$

2.7 Activity 2.4: PolyRoot

This is the final activity with polynomials. You will compute one estimated root of a polynomial using Newton's Method.

The main program will only call this method if the polynomial is non-zero. For example:

Cmd args	Native Array	x	Root
3 -1	[3, -1]	4.5	3.0
0 3	[0, 3]	-2.0	-1.3877787807814457E-17
2 -1 -2 1	[2, -1, -2, 1]	2.0	2.0000000358875707
-5 0 0 3 3 1	[-5, 0, 0, 3, 3, 1]	-3.9	0.9128983495621411

You are now ready to implement Newton's Method to find an approximation for a root of a polynomial. First, recall the steps in the algorithm. We start with an initial guess of the root, `x0`.

```
1. Evaluate the polynomial at the given guess, x0:
                  result = f(x0)

2. If the maximum number of iterations was exceeded, or the
  absolute value of result is less than or equal to some
   (epsilon) of error, return the guess x0 as the root.

3. Otherwise, compute a new guess using the intersection of
 the tangent at the current guess with the x-axis:
  x1 = x0 - f(x0)/f'(x0)

4. Repeat steps 1-3 with new guess x1.
```

The public method, `computeRoot`, is already written for you. It makes the initial guess and calls the recursive method. You will implement the recursive algorithm in the private helper method `newtonsMethod`. Remember, to implement Newton's Method you need access to the methods you already wrote in other classes.

This class also has the constants from the table below. You must use these in your implementation.

Constant	Type	Value	Description
EPSILON	double	0.000001	The acceptable error difference for the result.
INITIAL_GUESS	double	0.1	The initial guess for the root, x0.
MAX_ITERATIONS	int	100	The maximum number of iterations to perform when estimating the root.

3 Submission

Submit all your source code to the MyCourses assignment before the due date.

- Right click on your source directory, **src**, which is inside your project, and make a zip file (not .7z or .rar!), named **hw1.zip**.
- Go to MyCourses assignment.
- Upload the zip file to submit your work into Homework 1 dropbox.
- Check that your uploaded submission worked. If you forget a step, your work may not be submitted, and you will lose credits.

Double check by going to the dropbox and refreshing. Also, you should receive an email when you successfully submit.

4 Grading

The grade breakdown for this homework is as follows:

- Implementation:
  - Activity 1: 10%
  - Activity 2: 80%
    - PolyString: 25%
    - PolyEval: 20%
    - PolyDerive: 15%
    - PolyRoot: 20%
- Code Style & Documentation: 10%.
  Your code must follow the style guidelines of the course, including javadoc for each function and author's full name in the file comment block at the top of the file. For a full style example refer to here.
  Your output should be formatted to look like the output in the examples. This will make testing and grading easier. There will be a 5% reduction in score for failing to meet this condition.

4.1 Deductions

The following deductions will apply:

- Completeness: up to 100% if a part of the writeup is not observed.
- Correctness/Testing: up to 50% or affected part if the solution is not correct.
- Quality/Style: up to 50% if solution is poorly designed.
- Explanation: up to 100% if the student is not able to respond grader's questions.