Department of Electrical and Computer Engineering
University of Puerto Rico
Mayagüez Campus

ICOM 4035 – Data Structures
Spring 2013

Project #1: Algebra and Calculus on Polynomials
Due Date: 11:59 PM-March 19, 2013

Objectives

1. Understand the design, implementation and use of the ArrayList container class.
2. Gain experience implementing abstract data types using already developed data structures.
3. Gain experience with object-oriented programming concepts.

Overview

You will implement and test a polynomial class, using the ArrayList container class as the data structure to store the terms in a polynomial. The general form of a polynomial is as follows:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

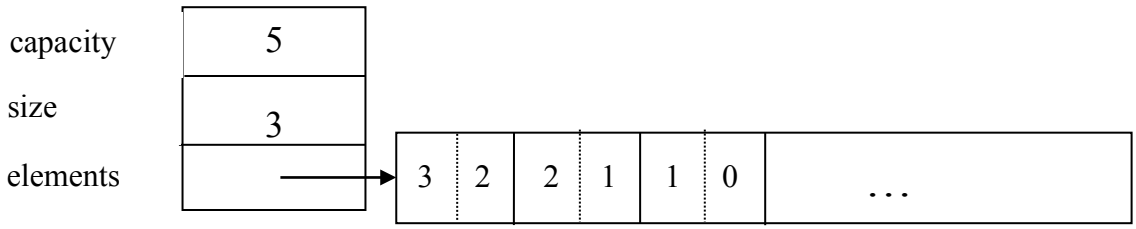
Here, each term has a coefficient, denoted as a_i, and a exponent *i*, which represent the power of the variable x. Your polynomial class must implement the following operations:

1. Addition – given two polynomials P_1 and P_2, compute the polynomial $P_3 = P_1 + P_2$.
2. Subtraction – given two polynomials P_1 and P_2, compute the polynomial $P_3 = P_1 - P_2$.
3. Multiplication – given two polynomials P_1 and P_2, compute the polynomial $P_3 = P_1 * P_2$.
4. Scalar Multiplication - given a polynomial P, multiply it by a constant *c*, and return it as a new polynomial.
5. Derivative – given a polynomial P, finds its derivative.
6. Indefinite integral – given a polynomial P, finds its indefinite integral (anti-derivative).
7. Definite integral – given a polynomial P, evaluate its definite integral over an interval [a,b].
8. Degree – given a polynomial P, find its degree (the largest exponent in any term).
9. Evaluate – given a polynomial P, evaluate it at value *x,* to compute $y = P(x)$.
10. String Converter – a given a polynomial *P*, return its string representation in the form:

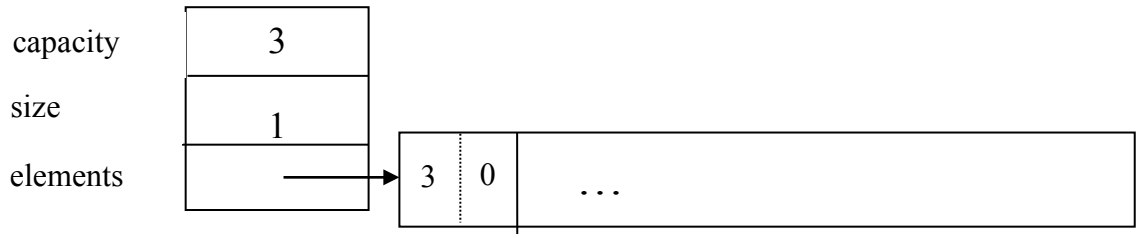
$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

You must use the ArrayList container class discussed in the lectures. The ArrayList container will store the terms in the polynomial, in decreasing order of exponent. Thus, each element in the

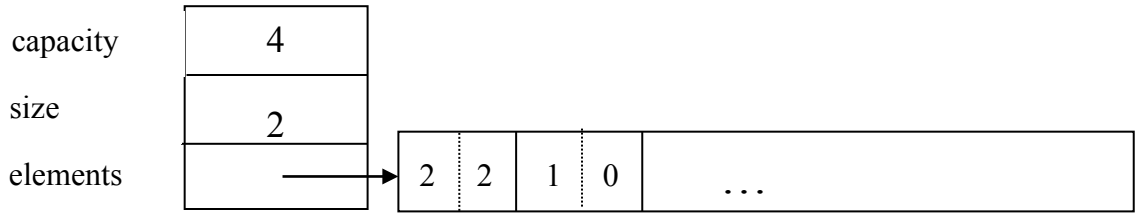
ArrayList represents a term in the polynomial. For example, if we need to represent the following polynomial: $3x^2 + 2x + 1$, then the organization of the ArrayList container associated with the polynomial class should look like this (assuming initial capacity of 5):



The dotted lines are meant to convey the fact that each element is a structure with two fields: the coefficient of the term and the exponent to which the variable x should be raised. In your implementation, you cannot store terms containing a coefficient equal to zero. The only exception is the case in which the polynomial correspond to P(x) = 0, meaning that the polynomial is just the number 0. When you implement your mathematical operations you must make sure you don't add terms to the polynomial that are zero. Again, the only exception is when the resulting polynomial is the value 0.
To clarify, this point consider the following expression: (2x + 1) – (2x – 2). In this case, the resulting polynomial will be 3, and the representation will be (assuming initial capacity of 3):



As you can see, the terms with variable x cancel out, and there is no need to represent 0x in the polynomial. Likewise, there is no need to represent the term corresponding to x raised to the first power in this polynomial: $2x^2 + 1$



In this case, the term corresponding to a*x* is not represented since the coefficient is zero.

You will be given two interfaces that specify what a polynomial and a term must do. These interfaces are called Polynomial.java and Term.java respectively. You implementation will consist of adding Java code to implement two modules: PolynomialImp.java (which implements

interface Polynomial) and TermImp.java (which implements Term.java). In addition you will receive three JUnit files that contain the test case for the project. The list of files is a follows:

1. Term.java – interface that defines operations in a polynomial term.
2. Polynomial.java – interface that defines operations in a polynomial.
3. TermImp.java – YOUR implementation of Term.java
4. PolynomialImp.java – YOUR implementation of Polynomial.java
5. Test1.java – JUnit Test Case 1: **Your implementation must pass this test case without errors to be considered a running program.**
6. Test2.java – JUnit Test Case 2: Second test case
7. Test3.java – JUnit Test Case 3: Third test case

By passing test cases Test1, Test2, and Test3 you will earn 70 pts in the project. The rest of the points will be awarded upon passing other test cases that the professor will use, plus verifying the documentation of the code.

You can go to the class web page and download a zip file containing all these files. Just access the link named Projects, and down load the sources files associated with the link: *Project #1 – Polynomials*.

PROJECT DUE DATE: **11:59 PM – March 19, 2013**.