Program #1
CSE 274
Polynomial ADT with *ArrayLists*

Implement and test an abstract data type for a polynomial. The ADT is specified below. You will find starter code for this assignment in the file *Polynomial.java* in the Canvas assignment.

Recall that polynomials look like this:

$f(x) = 3x^2 -2x + 4$ $\qquad$ $g(x) = -2x + 10x^{-3}$

Some of the operations that can be performed on polynomials include:

$f+g = 3x^2 - 4x + 4 + 10x^{-3}$
$f-g = 3x^2 + 4 - 10x^{-3}$
$f*g = -6x^3 + 4x^2 - 8x + 30x^{-1} - 20x^{-2} + 40x^{-3}$
$g(1) = 8$

Here is the specification of the Polynomial class.

```
public class Polynomial {
        private ArrayList<Node> terms;
        public Polynomial()
        public Polynomial(Polynomial poly)
        public Polynomial(ArrayList<Double> coef, ArrayList<Integer> expon)
        public boolean equals(Polynomial poly)
        public Polynomial add(Polynomial poly)
        public Polynomial subtract(Polynomial poly)
        public Polynomial multiply(Polynomial poly)
        public double evaluate(double value)
        public Polynomial derivative()
        public String toString()
}
```

You will notice that there are operations for adding subtracting and multiplying polynomials. We suggest that you implement the *add()* method first. Then the *subtract()* method is a fairly easy adaptation of the *add()* method. For the *multiply()* method, you will use the *add()* method to combine the partial products.

Note that we are specifying that this Polynomial ADT is to be implemented with an ArrayList. (If we were just specifying the Polynomial ADT we would not include that restriction.) You can see above the Polynomial class has one data member: *terms*. It is defined as an *ArrayList* of type *Node*. *Node* is a private internal class. We are providing some starter code that includes the needed code for the Node class.

```
private ArrayList<Node> terms;

private class Node implements Comparable<Node> {
        double coefficient;
        int exponent;

        public Node(double c, int e) {
                coefficient = c;
                exponent = e;
        }

        @Override
        public int compareTo( Node node ) {

                return node.exponent - this.exponent;
        }
}
```

This class includes two data members for the coefficient and the exponent. We include a simple constructor and implement the Comparable interface to allow *Node* objects to be sorted. With this code, you will be able to sort a *Polynomial* object to put the *ArrayList* into order by exponent with the method call: Collections.sort(terms);

The starter code that we have provided also gives the *toString()* method. It is important that you use this code without change to allow us to grade your work accurately.

Notes.

- Each coefficient of the polynomial will have type *double*; exponents will have type *int*.
- The string representation of the example polynomials above would be
    - 3.0x^2 – 2.0x + 4.0
    - -2.0x + 10x^-3
- The canonical string representation as displayed by *toString()* will have the following properties
    - Does not display terms with a 0.0 coefficient (unless the polynomial always evaluates to zero)
    - Note that exponents are indicated by ^. However, there is no need for you to store this "^". The *toString()* method takes care of this. Also, if the exponent is 1 this is represented as "x" rather than as "x^1". A constant term is not represented as "7.7x^0" but rather as "7.7. Again, the *toString()* handles this.

    - Terms with the same exponent should be combined. That is 5.0x^3, should not be displayed as 1.0x^3 + 4.0x^3.
- Use the exact spellings for the class and methods (case of letters included) given in the starter code.

When your Polynomial class code is complete, turn in the file Polynomial.java to Canvas. We will grade your program by running your code against a set of test cases. These test cases are very thorough. Thus, you should test your own code thoroughly including input error cases before submitting it. Your code will also be evaluated on its compliance to the department's Java coding standards.

Deleted: department's Java coding standards

We suggest that you start by implementing the constructor that takes in an *ArrayList* of exponents and a separate *ArrayList* of coefficients. We have provided the *toString()* method. So you can run test cases as soon as you have written this constructor. This will allow you to test incrementally and see your results. Thus, you will also have a Driver or Tester class to test your code. However, you will not turn that tester code in.