CSC205AB
MinilabArrayList - Polynomial

The assignment: You have already written a class called Term, which holds a Mathematical Term. For this Minilab, you are to write a class called Polynomial, which uses an ArrayList (or Vector if you want) to hold a number of Terms, which form a polynomial.

Changing your Term class: Please be sure that your Term class has data that is **private** since other classes should not be able to access the data directly. To make the Polynomial class work correctly, your Term class should also contain another method called:

```
//isZero – returns true if this instance of a Term is 0 (has a 0 coefficient)
public boolean isZero()
```

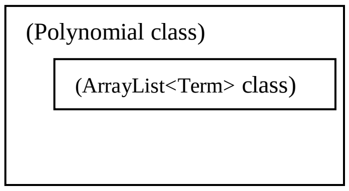
You should understand:

- how your Term class works, especially the following methods:
  - isZero
  - degree
  - addIn
  - multiplyBy
  - evaluate
- how an ArrayList works, especially the following methods:
  - add (both versions)
  - get
  - remove
  - isEmpty
  - size
  - contains
  - toString

Using the Term class in your Polynomial class ("has-a" vs. "is-a"): Your Polynomial class should use an ArrayList (or Vector) of Terms to hold its "data." It can either:

1) Have an ArrayList<Term> as its data, similar to our Queue demo. In that case, it would be a "has-a" relationship since the Polynomial "has a" ArrayList. The data and relationship are shown below:

```
public class Polynomial
{
        //------- data
        private ArrayList<Term> theAL;
```

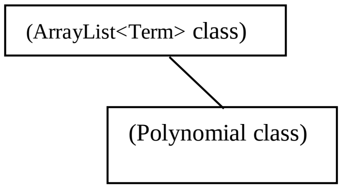


And every time a method wanted to manipulate the data, it would access the ArrayList's methods with a call such as:

```
theAL.add(i, aTerm);
```

2) Or…have an ArrayList<Term>) as parent class, so it would be an ArrayList itself and inherit ArrayList's methods. In that case, it would be an "is-a" relationship since the Polynomial "is a" ArrayList. The data and relationship are shown below:

```
public class Polynomial extends ArrayList<Term>
{
        //------- data
        (nothing, since it already is an ArrayList)
```



And every time a method wanted to manipulate the data, it would access its own (inherited) methods with a call such as:

```
this.add(i, aTerm);     //calls its own method that was inherited from ArrayList
```

However, do not use both "has-a" and "is-a" at the same time. Java allows it, but it is very easy to mix up which ArrayList is being manipulated.

Implementing PolynomialInterface: Your Polynomial class should implement PolynomialInterface. If you choose to use both "implements" and "extends" then they are written like this:

```
public class Polynomial extends ArrayList<Term> implements PolynomialInterface
```

You can have additional methods besides those specifically listed in the PolynomialInterface. You will have to do this to add the extra credit.

Inserting a Term into the ArrayList: The insert(Term aTerm) method is the most challenging, as the Terms should be in descending order by degree. The insert method will have to traverse the ArrayList and decide when and where to tell the ArrayList to add it or tell an existing Term to do an addIn. The methods of individual elements can be accessed like this:

```
for (int i=0; i<theAL.size(); i++)
        (for example) theAL.get(i).addIn(…
```

Or, if you decide to use a variable, like this:

```
{
  Term myTerm = theAL.get(i);
  myTerm.addIn(…
}
```

changing myTerm will also change the element in the ArrayList, since they are both references to the same Term.

Exact requirements and extra credit: The requirements for both the Minilab and the extra credit are described in the PolynomialInterface comments. Please do not change the PolynomialInterface, even if you do the extra credit.

Please submit: your Polynomial.java and updated Term.java (updated with isZero() method) via HyperGrade.