Credit where credit is due:

These problems are adapted from Ilkka Kokkarinen's excellent "CCPS 209 Computer Science II Labs" problem set. Credit for these problems goes to him, and the full set of 99 can be found at his Github page:

https://github.com/ikokkari

Polynomial III: Comparisons

This lab continues modifying the source code for the Polynomial class from the first two labs to allow **equality** and **ordering** comparisons to take place between objects of that type. Modify your Polynomial class signature definition so that it implements Comparable<Polynomial>. Then write the following methods to implement the equality and ordering comparisons.

`@Override public boolean equals(Object other)`

Returns true if the other object is also a Polynomial of the exact same degree as this, and that the coefficients of this and other polynomials are pairwise equal. If the other object is anything else, this method should return false.

`public int compareTo(Polynomial other)`

Implements the **ordering comparison** between this and other polynomials, as required by the interface Comparable<Polynomial>, allowing the instances of Polynomial to be *sorted* or stored inside some instance of TreeSet<Polynomial>. This method should return +1 if this is greater than the other, -1 if the other is greater than this, and return a 0 if both polynomials are equal in the sense of the equals method. **A total ordering relation** between polynomials can be defined by many possible different rules. We shall use an ordering rule that says that **any polynomial of a higher degree is automatically greater than any polynomial of a lower degree**, regardless of their coefficients. For two polynomials whose degrees are equal, the result of the order comparison is determined by **the highest-order term for which the coefficients of the polynomials differ**, so that the polynomial with a larger such coefficient is considered to be greater in this ordering. Be careful to ensure that this method ignores the leading zeros of high order terms if you have them inside your polynomial coefficient array, and that **the ordering comparison criterion is precisely the one defined in the previous paragraph**.

A common bug in student code at this point is to loop through the coefficients from lowest to highest, instead of looping through them down starting from the highest. This bug is particularly tricksy to pinpoint, since looping through the coefficients in the wrong direction will still produce a perfectly legal total ordering between polynomials that the collections of type TreeSet<Polynomial> would be perfectly happy to use. It just would not be the total ordering that the JUnit test class is expecting to see, but how could the student know that if they missed it in the casual reading of the specification?