Lab 06

This lab exercise must be submitted for a grade. You may work collaboratively with another student, but each student must submit his/her own solution using his/her own account. This exercise must be submitted by 11:00PM on Thursday 10/6. There is no extension for late submissions.

Checkout the project called 132Fall2016Lab06. It contains some public tests, but nothing else is provided. You will write a class called Triple. (Just leave it in the default package.)

After reading the following instructions, it might be useful to look over the public tests to get an idea of how your code should work.

Write a class called Triple with the following features. You may implement the class any way you choose – there are no restrictions, and there is no one "right" way to do it.

- The Triple class is similar to the Pair class from lecture, but a triple contains *three* objects (of the same type).
- You can't put just *any* kinds of things into a Triple; the elements that are stored in the Triple must be Comparable. You must use correct generic notation at the top of the Triple class to enforce this policy.
- The Triple class, itself, must also implement the Comparable interface.
    - In order to compare two Triple objects, you need to select the *largest* element of the first Triple and the *largest* element of the second triple. If these two elements match, return 0. If the element from the current object is larger than the element from the parameter, return something positive. If the element from the current object is smaller than the element from the parameter, return something negative.
- The Triple class must implement the Iterable interface. The Iterator should work as follows: Calls to next should return the three elements of the Triple, one by one, in increasing order. After three calls to next, hasNext should start returning false.

  For example, suppose the Triple contains the values 11, 35, and 2. In this case, the iterator should produce the sequence 2, 11, 35.

- Although they will not be graded, you should consider writing additional test cases to make sure that your code is working correctly!