Data Abstraction: Lecture Lab

Check out the project named `IntegerSetLecLab` from the lectures repository – do not use the `IntegerSet` project that we've been using in class!

1. Add a specification for the following method to the `IntegerSet` interface: The method `intersection` returns a new set that is the intersection of this set and another set passed as a parameter. The intersection of sets A and B is the set that contains all the elements that are common to both A and B.

2. Having added the specification of this new method to the `IntegerSet` interface, notice that the two classes that implement this interface no longer compile. What error message are you given?

3. Add a stub for the `intersection` method to the `LinkedListIntegerSet` and `HashMapIntegerSet` classes. Notice that these two classes now compile.

4. Having written a specification for the `intersection` method, you now know how it can be used. Complete the following segment of code so that the intersection of `setA` and `setB` is assigned to `intersectAB`. Note that there's more than one way to do this!

   ```
   IntegerSet setA = new LinkedListIntegerSet();
   IntegerSet setB = new LinkedListIntegerSet();

   // insert some elements to setA and setB (omitted)
   IntegerSet intersectAB =
   ```

5. Add unit tests to the `LinkedListIntegerSetTest` class that test the `intersection` method. Note that a set is data of arbitrary size. What did you learn in CPSC 110 about designing tests on this kind of data? It applies here too!

6. Finally, provide an implementation for the `intersection` method in the `LinkedListIntegerSet` class. Note that you are not expected to implement this method in the `HashMapIntegerSet` class. Run the tests on the `LinkedListIntegerSet` class by right-clicking `LinkedListIntegerSetTest.java` and choosing *Run LinkedListIntegerSetTest* from the pop-up menu. Debug until all tests pass!