Homework #2
due Monday, Feburary 4, 10:00 PM

In Homework #1, you implemented a few ADTs (in particular HexTile). For this assignment, you will build a Sequence ADT of hex tiles.

1 Concerning the HexTileSeq ADT

The HexTileSeq ADT is a variant of the Sequence ADT from the textbook (Section 3.3 on pages 145–158 (142–155, 3rd ed.)). Please read this section thoroughly! We use the same data structure in the same way. We give you a partial implementation to start with.

You are encouraged to write your own `toString` method which you will find useful when debugging the ADT, but you are not required to define such a method, nor are there any requirements for what the result should look like.

Unlike the textbook, we recommend that you do *not* use `System.arrayCopy`. While it does indeed run faster than doing a loop, it's not very clear what it is doing. Use a "for" loop for clarity. We also recommend putting the logic for determining whether there is sufficient capacity into `ensureCapacity` itself, rather than into all of its callers.

2 Concerning Invariants and Assertions

The data structure makes use of two integer fields and an array field. There are certain configurations that make no sense. Thus we will define and check object invariants, which are properties that should always hold true. If any of the invariant properties are violated, it means something is wrong with your implementation.

You will implement several class invariants within the `wellFormed` method in `HexTileSeq` class. Read the comments there for detailed instructions.

Recall that the beginning of every public method should have code as follows:

```
assert wellFormed() : "Invariant failed at start";
```

and at the end of any public method that changes any field, there must be the following line, right before the end:

```
assert wellFormed() : "Invariant failed at end";
```

We have placed these lines in the code in the skeleton file for your convenience. Do not remove them!

3 Concerning Clone

In Java, a `clone()` method is used for duplicating objects. The original object and the cloned object should have same values but different identities (i.e., they are `!=`). Therefore, modifying one of them will not affect the other (sometimes called the separation property). This is automatically guaranteed for primitive types. However, for reference types (including array), it usually takes more efforts (sometimes even impossible) to achieve this requirement. To which level does the separation property hold distinguish the concept of *deep clone* and *shallow clone*. For more information on deep clone, read book p. 313, exercise 5. Alternatively, Wikipedia has a few paragraphs on this topic: `https://en.wikipedia.org/wiki/Object_copying`. We leave a line of code for you to implement for this Homework.

4 Concerning the `Demo` Program

We provide most of a program that reads in a file of hex tile descriptions and shows the resulting game board graphically. Two sections of the code are omitted; these require you to use the HexTileSeq ADT that you are implementing.

To run it, put `test/sample.hex` as a "Program Argument" or type in text on the console in the same format as in `test/sample.hex`.

5 Random Testing

We provide random testing for this homework. Once you are passing all the normal tests, use random testing to find any bugs that might still linger in your code. For example, some students will add many cases to their code to satisfy tests without actually solving the true problem. Random will usually find an example where such code doesn't meet the specification. It runs your code with a random series of commands looking for behavior that deviates from the required behavior. If it finds a problem, it will print a JUnit test case tailored to that particular problem. You should take the output and copy it into a file and run it as a JUnit test to see the problem. The test are usually long and involved.

To run the random test, open `homework2.jar` in "Referenced Libraries," find `RandomTest` in the default package and run it from there.

6 Files

In the git repository for this assignment, we provide the following files (and others unnamed):

- `src/TestHexTileSeq.java` JUnit test cases. Do not modify this file.
- `src/TestInvariant.java` Access to the invariant tester: we have a few tests to check that your `wellFormed()` does the correct thing.
- `src/TestEfficiency.java` JUnit test cases that check whether you are using efficient techniques. Do *not* enable the invariant checker while running these tests. The tests should take at most a second or two. Anything longer may indicate that you are using inefficient techniques.
- `src/edu/uwm/cs351/HexTileSeq.java` A skeleton implementation of the Sequence ADT.
- `src/Demo.java` The GUI program, it will not work properly until all the other classes are completed.
- `test/sample.hex` An example file of hex tiles to use for `Demo.java`.

7 What you need to do

You need to complete the `HexTileSeq` and `Demo` classes. These classes must be pushed to your homework2.git repository (in AFS) before the deadline.