Your task for this assignment is in three parts.

* (Part 1) You are to implement a class called *Benchmark*. Please see the skeleton class that I created in the repository. The API of this class is as follows:

**public class** Benchmark<T> {  
 **public** Benchmark (Function<T,Void> f)  
 **public double** run(T t, **int m**)  
}

The generic type *T* is that of the input to the function *f* which you will pass in to the constructor. The return type of your function *f* is *Void* (which means you will have to return *null* to close the lambda that you will write in your application).

Function *f* is the function whose timing you want to measure. For example, you might create a function which sorts an array with *n* elements.

The first parameter to the *run* function is the parameter that will in turn be passed to function *f*. In the example mentioned above, this parameter will be *n*, an *Integer* and the number of elements to be sorted.

The second parameter to the *run* function (*m)* is the number of times the function *f* will be called.

The return value from *run* is the average number of milliseconds taken for each run of function *f*.

* (Part 2) Implement two more classes: *SelectionSort* and *InsertionSort*. Each will define a method *sort* which takes as its input an *array<Comparable>[]* of some length (we referred to this as *n* above). See the stubs that I created in the repository. Use the implementations in the book (and lecture slides) or use your own implementation.
* (Part 3) Measure the running times of these two different sorts, with random arrays, ordered arrays and reverse-ordered arrays. I suggest that your arrays to be sorted are of type *Integer*. Use the doubling method for choosing *n*and test for at least five values of *n.*

As usual, the submission will be your entire project (*clean, i.e. without the target and project folders).*I will provide stubs and unit tests in the repository (they are not there just yet).

Report on your observations and show screen shots of (some of) the runs and also the unit tests.

Further notes: you should probably use the *System.nanos* method to get the clock time. This isn't guaranteed to be accurate which is one of the reason you should run the experiment several times for each value of *n*. Also, for each invocation of *run*, run the given function *f* 10 times to get the system "warmed up" before you start the timing proper.

Since you don't want to instantiate and populate a new array every time *f* is called, you should start out by creating a static array of sufficient length and each time the *f* function is called, use the *Integer* passed in to determine the number of elements to be sorted (i.e. not necessarily the whole array). Actually, you mnight have several of these static arrays: random, ordered, partially-ordered (?) and reverse-ordered.

If you need clarification, ask on Slack.

This assignment is a bit harder than I intended. The deadline has been extended to Saturday night.