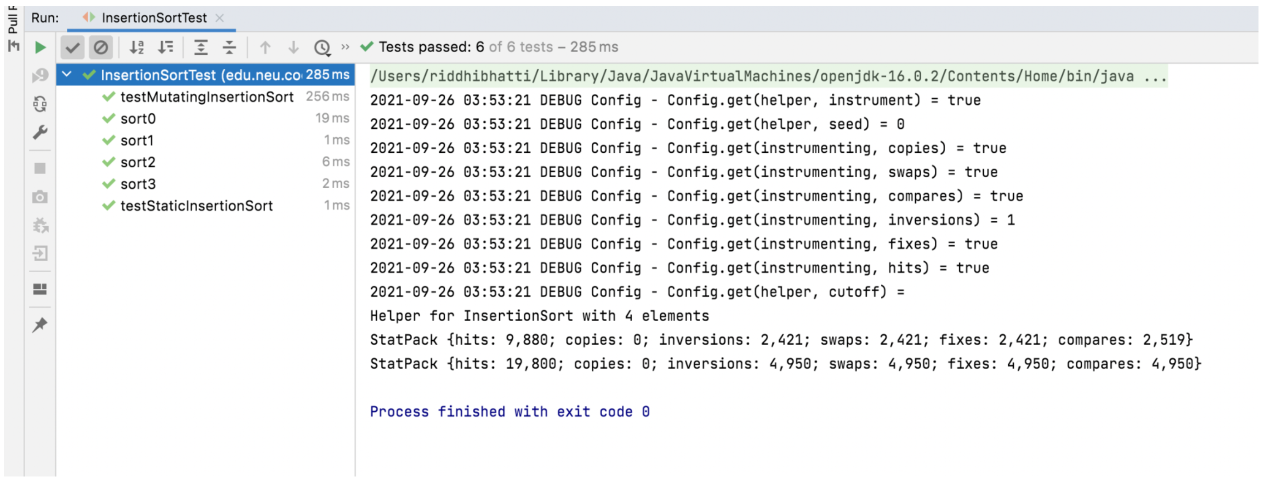
Riddhi Bhatti | NUID: 001502713

Program Structures & Algorithms
Fall 2021
Assignment 2

1. **Tasks Performed in the Assignment :**
    a. **Implemented the code for methods methods mentioned in Timer and InsertionSort class and created a new class for testing insertion sort for different types of array**
    b. **Ran the experiment using a doubling method for 5 different values of n where n is the number of elements in the array.**
    c. **Ensured that all the test cases ran successfully**

2. Relationship conclusion:

3. Evidence to support conclusion

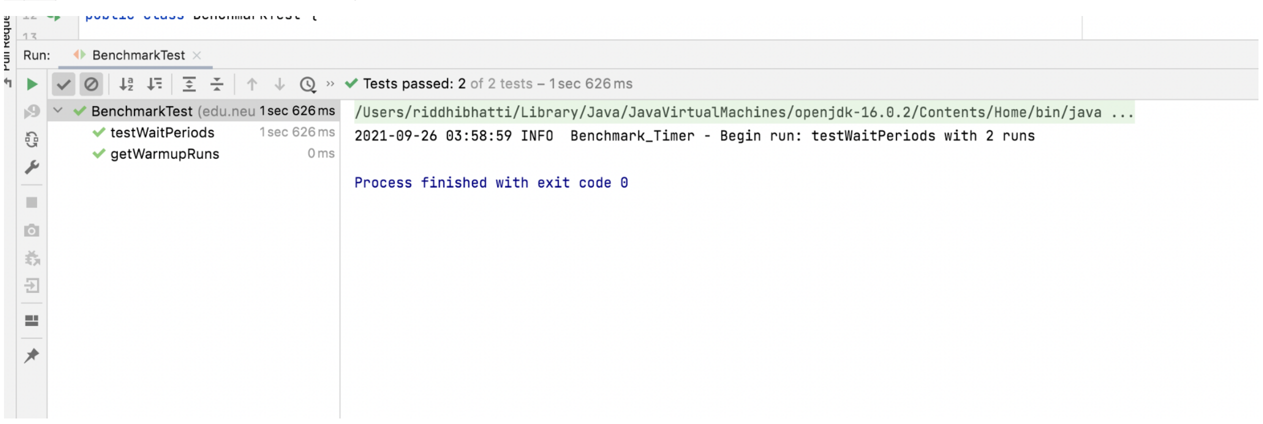


```
Run:  InsertionSortTest
✓ Tests passed: 6 of 6 tests – 285 ms

✓ InsertionSortTest (edu.neu.co 285 ms
   ✓ testMutatingInsertionSort  256 ms
   ✓ sort0                       19 ms
   ✓ sort1                        1 ms
   ✓ sort2                        6 ms
   ✓ sort3                        2 ms
   ✓ testStaticInsertionSort      1 ms

/Users/riddhibhatti/Library/Java/JavaVirtualMachines/openjdk-16.0.2/Contents/Home/bin/java ...
2021-09-26 03:53:21 DEBUG Config - Config.get(helper, instrument) = true
2021-09-26 03:53:21 DEBUG Config - Config.get(helper, seed) = 0
2021-09-26 03:53:21 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-26 03:53:21 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-26 03:53:21 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-26 03:53:21 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-26 03:53:21 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-26 03:53:21 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-26 03:53:21 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,880; copies: 0; inversions: 2,421; swaps: 2,421; fixes: 2,421; compares: 2,519}
StatPack {hits: 19,800; copies: 0; inversions: 4,950; swaps: 4,950; fixes: 4,950; compares: 4,950}

Process finished with exit code 0
```

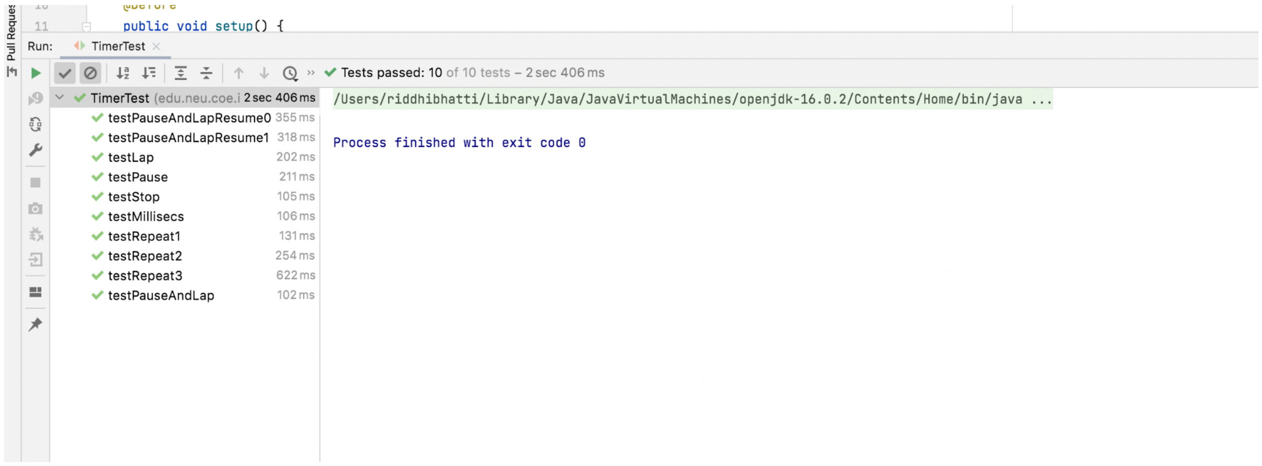


```
Run:  BenchmarkTest
✓ Tests passed: 2 of 2 tests – 1 sec 626 ms

✓ BenchmarkTest (edu.neu 1 sec 626 ms
   ✓ testWaitPeriods    1 sec 626 ms
   ✓ getWarmupRuns             0 ms

/Users/riddhibhatti/Library/Java/JavaVirtualMachines/openjdk-16.0.2/Contents/Home/bin/java ...
2021-09-26 03:58:59 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs

Process finished with exit code 0
```



```
public void setup() {
```

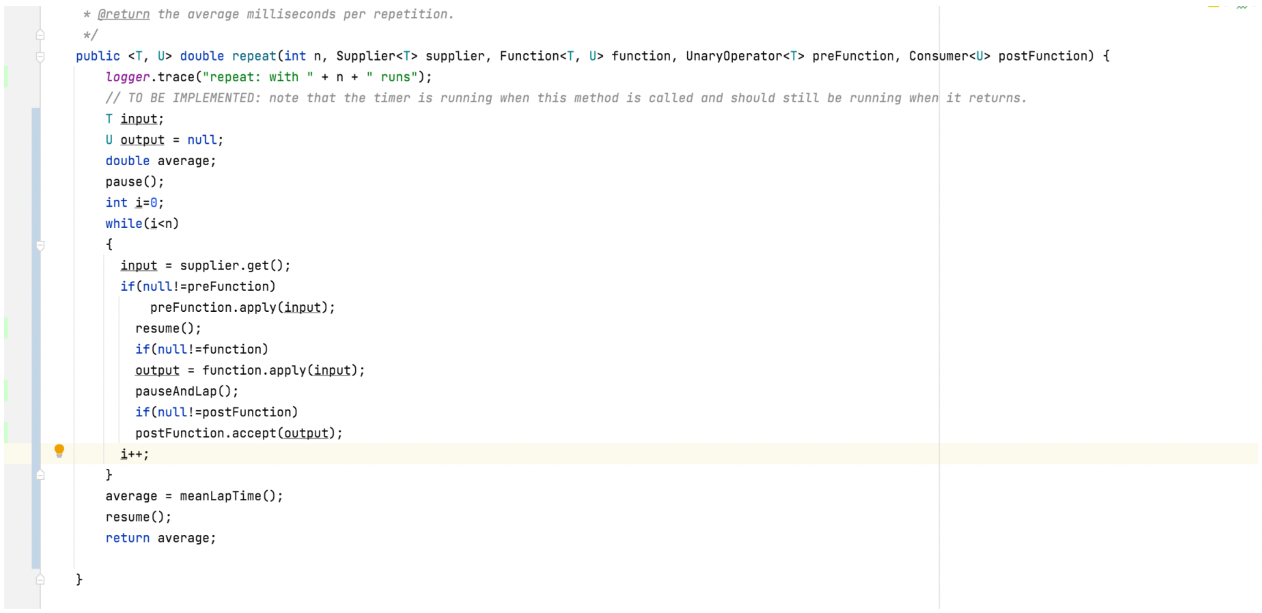
Run: TimerTest

Tests passed: 10 of 10 tests – 2 sec 406 ms

```
TimerTest (edu.neu.coe.i 2 sec 406 ms
  testPauseAndLapResume0  355 ms
  testPauseAndLapResume1  318 ms
  testLap                 202 ms
  testPause               211 ms
  testStop                105 ms
  testMillisecs           106 ms
  testRepeat1             131 ms
  testRepeat2             254 ms
  testRepeat3             622 ms
  testPauseAndLap         102 ms
```

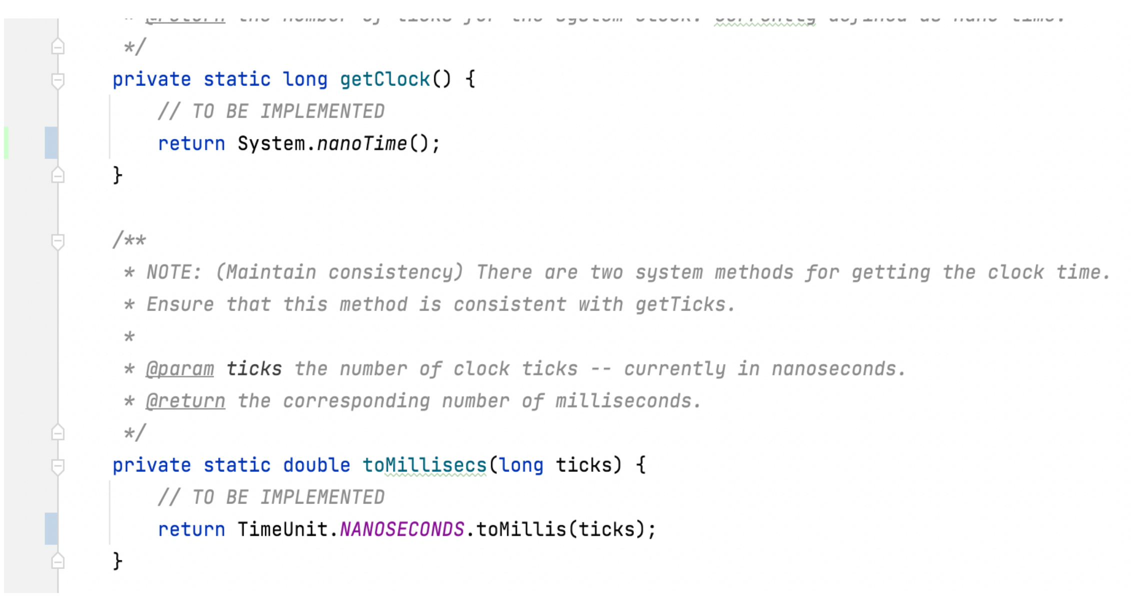
```
/Users/riddhibhatti/Library/Java/JavaVirtualMachines/openjdk-16.0.2/Contents/Home/bin/java ...

Process finished with exit code 0
```



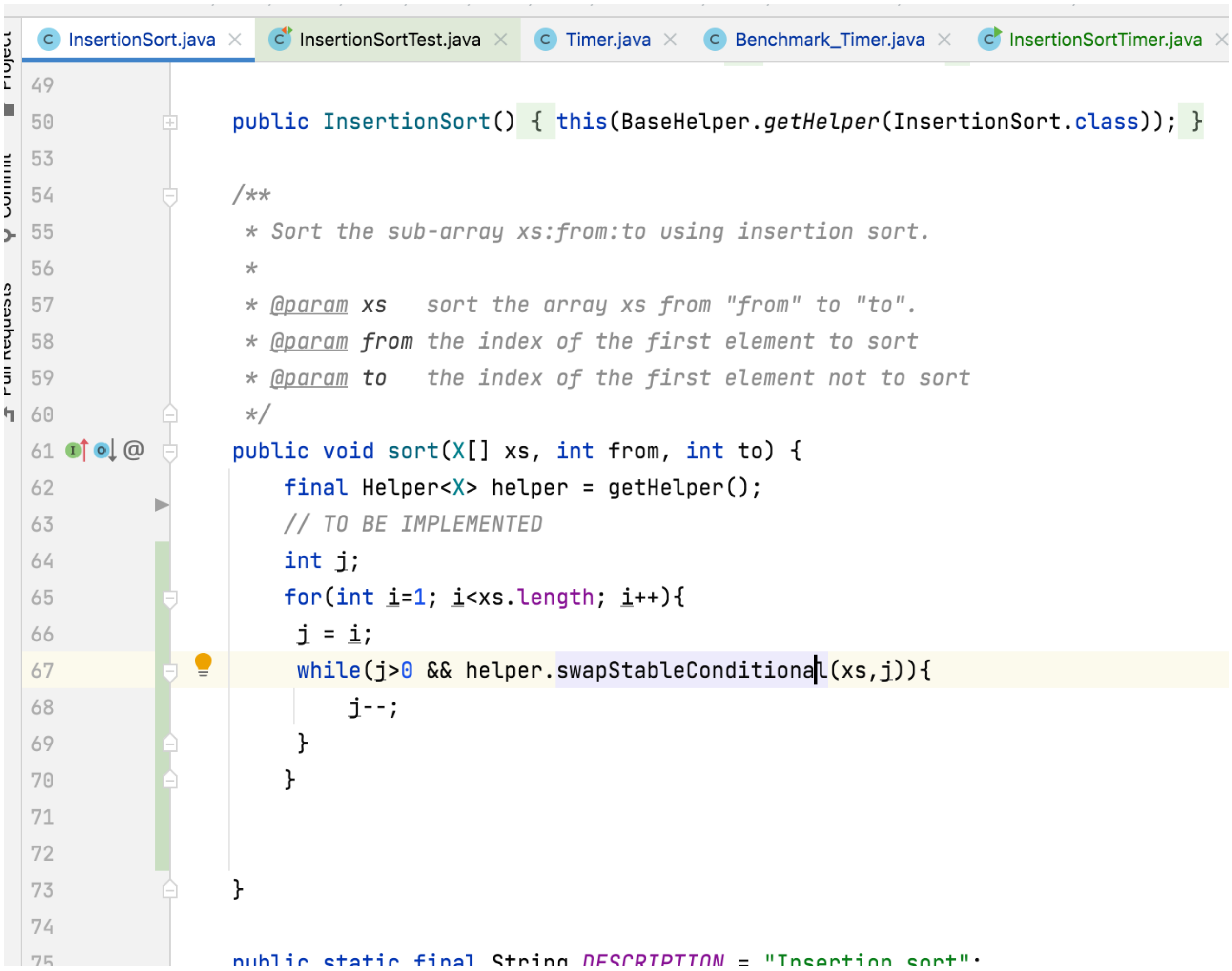
```
 * @return the average milliseconds per repetition.
 */
public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Consumer<U> postFunction) {
    logger.trace("repeat: with " + n + " runs");
    // TO BE IMPLEMENTED: note that the timer is running when this method is called and should still be running when it returns.
    T input;
    U output = null;
    double average;
    pause();
    int i=0;
    while(i<n)
    {
      input = supplier.get();
      if(null!=preFunction)
          preFunction.apply(input);
        resume();
        if(null!=function)
        output = function.apply(input);
        pauseAndLap();
        if(null!=postFunction)
        postFunction.accept(output);
      i++;
    }
    average = meanLapTime();
    resume();
    return average;

}
```



```
 */
private static long getClock() {
    // TO BE IMPLEMENTED
    return System.nanoTime();
}

/**
 * NOTE: (Maintain consistency) There are two system methods for getting the clock time.
 * Ensure that this method is consistent with getTicks.
 *
 * @param ticks the number of clock ticks -- currently in nanoseconds.
 * @return the corresponding number of milliseconds.
 */
private static double toMillisecs(long ticks) {
    // TO BE IMPLEMENTED
    return TimeUnit.NANOSECONDS.toMillis(ticks);
}
```



InsertionSort.java × InsertionSortTest.java × Timer.java × Benchmark_Timer.java × InsertionSortTimer.java ×

```java

    public InsertionSort() { this(BaseHelper.getHelper(InsertionSort.class)); }

    /**
     * Sort the sub-array xs:from:to using insertion sort.
     *
     * @param xs   sort the array xs from "from" to "to".
     * @param from the index of the first element to sort
     * @param to   the index of the first element not to sort
     */
    public void sort(X[] xs, int from, int to) {
        final Helper<X> helper = getHelper();
        // TO BE IMPLEMENTED
        int j;
        for(int i=1; i<xs.length; i++){
         j = i;
         while(j>0 && helper.swapStableConditional(xs,j)){
             j--;
         }
        }


    }

    public static final String DESCRIPTION = "Insertion sort";
```